

INTERNATIONAL
STANDARD

ISO/IEC
14394

First edition
1996-06-01

**Information technology — Directory
services C language interfaces — Binding
for Application Program Interface (API)**

*Technologies de l'information — Interfaces de langage C des services de
l'annuaire — Liant pour interface de programme d'application (API)*



Reference number
ISO/IEC 14394:1996(E)

AVA **Documents**
Document Sharing Hub

Contents

	PAGE
Section 1: General	1
1.1 Scope	1
1.2 Normative References	1
1.3 Conformance	2
1.4 Test Methods	3
Section 2: Terminology and General Requirements	5
2.1 Conventions	5
2.2 Definitions	11
Section 3: The Generic Interface	23
3.1 Introduction	23
3.2 Interface Functions	24
3.2.1 Overview	24
3.2.2 Common Datatypes	24
3.2.3 Abandon	24
3.2.4 Add Entry	25
3.2.5 Bind	25
3.2.6 Compare	25
3.2.7 Initialize	25
3.2.8 List	26
3.2.9 Modify Entry	26
3.2.10 Modify RDN	26
3.2.11 Read	26
3.2.12 Receive Result	27
3.2.13 Remove Entry	27
3.2.14 Search	27
3.2.15 Shutdown	27
3.2.16 Unbind	28
3.2.17 Version	28
3.3 Directory Service Package	28
3.3.1 Object Identifiers	28
3.3.2 Enumeration Types and Constants	30
3.3.3 OM Object Constants	32
3.3.4 Integer Constants	32
3.3.5 Class Constraints	33
3.4 The <xds.h> Header	34

© ISO/IEC 1996

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

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

Printed in Switzerland

Section 4: The X.500 Directory Service	35
4.1 Overview	35
4.2 Basic Directory Contents Package	35
4.2.1 Directory Object and Attribute Object Identifiers	35
4.2.2 OM Object and Attribute Type Identifiers	36
4.2.3 Directory Enumeration Types and Constants	37
4.2.4 Integer Constants	38
4.3 Strong Authentication Package	38
4.3.1 Directory Object and Attribute Object Identifiers	38
4.3.2 OM Object and Attribute Type Identifiers	39
4.3.3 Enumeration Types and Constants	40
4.3.4 Integer Constants	41
4.4 MHS Directory User Package	41
4.4.1 Directory Object and Attribute Object Identifiers	41
4.4.2 OM Object and Attribute Type Identifiers	41
4.4.3 Enumeration Types and Constants	42
4.5 Header Files	43
4.5.1 The <xdsbdc.p.h> Header	43
4.5.2 The <xdssap.h> Header	43
4.5.3 The <xdsmdup.h> Header	44
Section 5: Definitions of Constants	45
5.1 Introduction	45
5.2 Directory Service Package	45
5.2.1 Object and Attribute Type Identifiers	45
5.2.2 Enumeration Types and Constants	45
5.2.3 OM Object Constants	48
5.2.4 Integer Constants	48
5.3 Basic Directory Contents Package	49
5.4 Strong Authentication Package	49
5.5 MHS Directory User Package	49
Annex A (informative) Example Header Files	51
A.1 Introduction	51
A.2 Example <xds.h> Header	51
A.3 Example <xdsbdc.p.h> Header	58
A.4 Example <xdssap.h> Header	60
A.5 Example <xdsmdup.h> Header	62
Annex B (informative) Example Programs	65
B.1 Introduction	65
B.2 Synchronous Directory Example	65
B.3 Asynchronous Directory Example	69
B.4 Example Existing Practice Directory Service Implementation	72
B.5 Error Handling Module	79
Annex C (informative) Bibliography	81
Alphabetic Topical Index	83

TABLES

Table 2-1	– C Naming Conventions	8
Table 2-2	– Identifier Abbreviations	10
Table 4-1	– BDCP Directory Attribute Types	36
Table 4-2	– BDCP Directory Object Classes	37
Table 4-3	– Preferred Delivery Method Values	38
Table 4-4	– BDCP Upper Bounds	39
Table 4-5	– SAP Directory Attribute Types	39
Table 4-6	– SAP Directory Object Classes	39
Table 4-7	– SAP Upper Bounds	41
Table 4-8	– MDUP Directory Attribute Types	42
Table 4-9	– MDUP Directory Object Classes	42
Table 5-1	– OM Object Constant Names	48
Table 5-2	– Search Subset Constant Names	48
Table 5-3	– Receive Result Flag Constant Names	49
Table B-1	– OM Attributes of a Hostent	74

Foreword

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

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

International Standard ISO/IEC 14394 was prepared by IEEE (as IEEE Std 1327.2-1993) and was adopted, under a special “fast-track procedure”, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

Annexes A to C of this International Standard are for information only.

1 Introduction

(This introduction is not a normative part of ISO/IEC 14394, Information technology—Directory services C language interfaces—Binding for Application Program Interface (API), but is included for information only.)

The purpose of this International Standard is to define a C programming language binding for the Application Program Interface (API) to Directory Services that is defined in ISO/IEC 14392 {5}.

A directory is a distributed collection of information, which programs can access in order to make queries or updates. ISO/IEC 14392 {5} 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 1988 CCITT X.500 Series recommendations and ISO/IEC 9594: 1990. ISO/IEC 14392 {5} 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 ISO/IEC 14392 {5} or in this International 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.

28 Related Standards

The programming language independent specification on which the programming language binding defined in this International Standard is based is contained in ISO/IEC 14392 {5}.

This International Standard is intended to be used in conjunction with the C programming language binding defined in ISO/IEC 14364 {4} for the OSI abstract data Manipulation (OM) API defined in ISO/IEC 14360 {3}.

ISO/IEC 14395 {7} specifies a set of requirements to be satisfied by test methods for measuring conformance to this International Standard. In addition, such test methods must also satisfy the requirements specified in ISO/IEC 14393 {6} for test methods for measuring conformance to the programming language independent specification of the API.

ISO/IEC 9954: 1990 and the CCITT X.500 Series recommendations define the general model for the directory services to which the API defined in this International Standard provides access.

The API is intended to be used, if required, in conjunction with the C language binding defined in ISO/IEC 14365 (B14) to the API to X.400-based electronic messaging defined in ISO/IEC 14361 (B13).

Overview

The API defined in this International Standard contains three major parts. The first part is mandatory; it consists of the interface for accessing the services defined in the programming language independent specification, together with definitions for the information types required to access any directory service. The second part is optional; it consists of definitions for the information types required to access the particular directory service defined in X.500 directory standards, and is defined in Section 4. The third part is also optional; it contains definitions of symbolic constants that are required to ensure interworking with implementations of related services, such as the API to X.400-based electronic messaging defined in ISO/IEC 14365 (B14).

This International Standard is based on IEEE Std 1327.2-1993 (B16), which 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.

Information technology—Directory services C language interfaces—Binding for Application Program Interface (API)

Section 1: General

1.1 Scope

This International Standard defines an ISO/IEC 9899: 1990 {1} C language binding to the language-independent API defined in ISO/IEC 14392 {5}.¹⁾ It is intended to support application portability at the source-code level and to be used by application developers, system implementors, test method writers, and users.

1.2 Normative References

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

{1} ISO/IEC 9899:1990,²⁾ *Programming Languages—C*.

-
- 1) The numbers in curly brackets correspond to those of the references in 1.2. When preceded by a "B," the numbers correspond to those of the bibliography in Annex C.
 - 2) ISO/IEC documents can be obtained from the ISO Central Secretariat, 1 Rue de Varembe, Case Postale 56, CH-1211, Genève 20, Switzerland/Suisse.

- {2} ISO/IEC 9945-1: 1990³⁾ (IEEE Std 1003.1: 1990),⁴⁾ *Information technology—Portable Operating System Interface (POSIX)—Part 1: System Application Program Interface (API) [C Language]*.
- {3} ISO/IEC 14360: 1996, *Information technology—Open Systems Interconnection (OSI) abstract data manipulation—Application Program Interface (API) [Language independent]*.
- {4} ISO/IEC 14364: 1996, *Information technology—Open Systems Interconnection (OSI) abstract data manipulation C language interfaces—Binding for Application Program Interface (API)*.
- {5} ISO/IEC 14392: 1996, *Information technology—Directory services—Application Program Interface (API) [Language independent]*.
- {6} ISO/IEC 14393: 1996, *Information technology—Test methods for measuring conformance to directory services—Application Program Interface (API) [Language independent]*.
- {7} ISO/IEC 14395: 1996, *Information technology—Test methods for measuring conformance to directory services C language interfaces—Binding for Application Program Interface (API)*.

1.3 Conformance

1.3.1 Implementation Conformance

1.3.1.1 Requirements

A conforming implementation for this International Standard shall be a conforming implementation, as defined in ISO/IEC 14392 {5}, for the programming language binding specification that is contained in this International Standard.

1.3.1.2 Documentation

A conformance document shall be available for an implementation claiming conformance to this International Standard. It shall satisfy the requirements that are defined in ISO/IEC 14392 {5} for a conformance document for a programming language binding specification.

3) ISO/IEC 9945-1: 1990 is currently under revision.

4) 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.

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

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

1.3.2 Application Conformance

All applications claiming conformance to this International Standard shall fall within one of the categories in the following subclauses.

1.3.2.1 Strictly Conforming Application

A Strictly Conforming Application for this International Standard is a Strictly Conforming Application as defined in ISO/IEC 14392 {5} for the programming language binding specification contained in this International Standard.

1.3.2.2 Conforming Application

There is only one type of Conforming Application.

1.3.2.2.1 ISO/IEC Conforming Application

An ISO/IEC Conforming Application for this International Standard is an ISO/IEC Conforming Application as defined in ISO/IEC 14392 {5} for the programming language binding specification contained in this International Standard.

1.3.2.3 Conforming Application Using Extensions

A Conforming Application Using Extensions for this International Standard is a Conforming Application Using Extensions as defined in ISO/IEC 14392 {5} for the programming language binding specification contained in this International Standard.

1.4 Test Methods

Any measurement of conformance to this International Standard shall be performed using test methods that conform to ISO/IEC 14393 {6} ISO/IEC 14395 {7}.

Section 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 by both OSI abstract data manipulation and directory services descriptions. 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 International 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, one by the language-independent specification conventions and one 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**.)

Function synopses and other extended pieces of C code appear in constant width (courier) font while C language names embedded in ordinary text appear in italic font. Underscores are used to separate words in C language names. The C language names are derived mechanically from the corresponding language-independent names in a manner described in 2.1.4.

A C language function, datatype, or code fragment appears in *italic* font. A function name is indicated by following parentheses [e.g., *ds_abandon()*].

A C language symbolic constant, other than an error name, is surrounded by braces (e.g., {DS_MAX_OUTSTANDING_OPERATIONS}).

A C language symbolic constant for an error name is surrounded by brackets (e.g., [DS_E_ADMIN_LIMIT_EXCEEDED]).

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 International 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*
- C language names embedded in ordinary text
- 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 International Standard is based

— Function synopses and other extended pieces of C code.

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 specification for this International Standard is contained in ISO/IEC 14392 {5}. The language-independent specification conventions apply to that standard and to language specification terms used in this International Standard. These conventions are described in ISO/IEC 14360 {3}.

2.1.4 C Language Binding Conventions

2.1.4.1 Introduction

This International Standard specifies C identifiers for all the elements of the interface, so that application programs written in C can access the directory. These elements include function names, typedef names, and constants. All the C identifiers are mechanically derived from the language-independent names as explained below.

2.1.4.2 C Naming Conventions

The interface uses part of the C public namespace for its facilities. All identifiers start with the letters *ds*, *DS*, or *OMP*. More detail about the conventions used is given in Table 2-1. The interface reserves all identifiers starting with the letters *dsP* for private (i.e., internal) use by implementations of the interface. It also reserves *all* identifiers starting with the letters *dsX* or *DSX* for extensions of the interface. If the `<xds.h>` header is included, the application shall not use any identifier starting with these letters.

ISO/IEC 14360 {3} uses similar, though not identical, naming conventions. All OSI abstract data manipulation identifiers are prefixed by the letters *OM* or *om*.

No implementation of the interface shall use any public identifiers other than those defined in this International Standard and those beginning *dsP*, *dsX*, *DSX*, or *OMP*. A *public identifier* is any name except those reserved in the C Standard

Table 2-1 – C Naming Conventions

Item	Prefix
Reserved for implementors	<i>dsP</i>
Reserved for interface extensions	<i>dsX</i>
Reserved for interface extensions	<i>DSX</i>
Reserved for implementors	<i>OMP</i>
Functions	<i>ds_</i>
Error “problem” values	<i>DS_E_</i>
OM class names	<i>DS_C_</i>
OM value length limits	<i>DS_VL_</i>
OM value number limits	<i>DS_VN_</i>
Other constants	<i>DS_</i>
Attribute Type	<i>DS_A_</i>
Object Class	<i>DS_O_</i>

(ISO/IEC 9899 {1}), and the *public namespace* is the set of all possible public identifiers.

The C identifiers are derived from the language-independent names used throughout this International Standard by a purely mechanical process that depends on the kind of name.

- The C function names are identical to the language-independent operation names. Within text, they are italicized and followed by “()”.

Thus

— *ds_receive_result*

becomes

— *ds_receive_result()*

- C function input parameters are identical to the corresponding language-independent argument names. C function output parameters are derived from the argument names by adding “_return” as a suffix.

Thus, the output argument

— *operation_status*

becomes

— *operation_status_return*

(Within text, both language-independent argument names and C function parameter names are italicized.)

- The names of constants identifying OM classes are derived from the class names by converting lowercase to uppercase, converting hyphens to under-scores, and adding the prefix “DS_C_”.

Thus

— **Read-Result**

becomes

— {DS_C_READ_RESULT}

- The names of C language constants that denote language-independent specification choice values are derived from the choice value names by converting lowercase to uppercase and adding the prefix “DS_”.

Thus

— **Default-Context**

becomes

— {DS_DEFAULT_CONTEXT}

- Enumeration tags are derived from the name of the corresponding OM syntax by adding the prefix “DS_”. Hyphens are converted to underscores, but the case of letters is left unchanged.

Thus

— **Enum(Limit-Problem)**

becomes

— *DS_Limit_Problem*

- For enumeration constants, OM attributes, and all other constants except errors, lowercase is converted to uppercase, hyphens are converted to underscores, and the prefix “DS_” is added.

Thus

— **O-Residential-Person**

becomes

— {DS_O_RESIDENTIAL_PERSON}

- Errors are treated as a special case. Constants that are the possible values of the OM attribute **Problem** of a subclass of the OM class **Error** have hyphens converted to underscores, are made entirely uppercase, and are prefixed by “DS_E_”.

Thus

— alias-dereferencing-problem

becomes

— [DS_E_ALIAS_DEREFERENCING_PROBLEM]

- The constants in the “Value Length” and “Value Number” columns of the OM class definition tables are also assigned identifiers. (They have no names in ISO/IEC 14392 {5}.) When the upper limit in one of these columns is not “1” (one), it is given a name consisting of the OM attribute name prefixed by “DS_VL_” for value length or “DS_VN_” for value numbers.

- The sequence of octets for each object identifier is also assigned an identifier for internal use by certain OM macros. These identifiers are all uppercase and are prefixed by “OMP_O_”. See ISO/IEC 14360 {3} for further details on the use of object identifiers.
- In some cases, the transformations described above result in identifiers that are more than 32 characters in length. The ISO C Standard {1} only requires identifiers to be unique within the first 32 characters. The abbreviations listed in Table 2-2 are applied to substrings of identifiers to reduce their length.

Table 2-2 – Identifier Abbreviations

Identifier	Abbreviation	Identifier	Abbreviation
ADMINISTRATIVE	ADMIN	NUMBER	NBR
AGENT	AG	OFFICE	OFF
APPLICATION	APPLIC	OPTIONAL	OPT
AUTOMATIC	AUTO	ORGANIZATIONAL	ORG
CERTIFICATE	CERT	ORGANIZATION	ORG
CRITICAL	CRIT	PERMISSION	PERM
DELIVERY	DELIV	PHYSICAL	PHYS
DESTINATION	DEST	PREFERRED	PREF
EXTENSION	EXT	PROHIBITED	PROHIB
EXTENSIONS	EXT	PROVINCE	PROV
IDENTIFIER	IDENT	QUALIFIER	QUAL
IDENTIFIER	IDENT	REFERENCE	REF
INAPPROPRIATE	INAPPROP	REVOCATION	REVOC
INFORMATION	INFO	SUPPORTED	SUP
INTERNATIONAL	INTERNAT	SUPPORTED	SUPPORT
MODIFICATION	MOD	TERMINAL	TERM

Note that hyphens are translated to underscores.

2.1.4.3 Use and Implementation of Interfaces

If an argument to a function has an invalid value (such as a value outside the domain of the function, a pointer outside the address space of the program, or a null pointer), the behavior is undefined.

Any function declared in a header may be implemented as a macro defined in the header, so a library function should not be declared explicitly if its header is included. Any macro definition of a function can be suppressed locally by enclosing the name of the function in parentheses, because the name is not then followed by the left parenthesis that indicates expansion of a macro function name. For the same syntactic reason, it is permitted to take the address of a library function even if it is also defined as a macro. The use of *#undef* to remove any macro definition will also ensure that the identifier refers to an actual function. Any invocation of a library function that is implemented as a macro shall expand to code that evaluates each of its arguments exactly once, fully protected by parentheses where necessary, so it is generally safe to use arbitrary expressions as arguments. Likewise, any function-like macros can be invoked in an expression anywhere a function with a compatible return type could be called.

These concepts apply in this International Standard unless explicitly stated otherwise in the C language binding descriptions.

2.1.4.4 Function Return Values

Except for *ds_initialize()*, the return value of a C function is bound to the *Status* result of the language-independent description. Functions shall return a value of type *DS_status*, which is an error indication. If and only if the function succeeds, its value shall be **success**, expressed in the C language by the constant {DS_SUCCESS}. If a function returns a status other than this, then it shall not have updated the return parameters. The value of the status, in this case, shall be an error as described in ISO/IEC 14392 {5}.

Since the C language does not provide multiple return values, functions shall return all other results by writing them into storage passed by the application program. Any argument that is a pointer to such storage has a name ending with *_return*. For example, the C parameter declaration

— *om_uint * completion_flag_return*

indicates that the function shall return an unsigned integer *Completion-Flag* as a result, so the actual argument to the function shall be the address of a suitable variable. This notation allows the reader to distinguish between an input parameter that happens to be a pointer and an output parameter where the “*” is used to simulate the semantics of passing by reference.

2.2 Definitions

2.2.1 Terminology

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

2.2.1.1 conformance document: A document provided by an implementor that contains implementation details as described in 1.3.1.2. [ISO/IEC 9945-1 {2}]

2.2.1.2 implementation defined: An indication that the implementation shall define and document the requirements for correct program constructs and correct data of a value or behavior. [ISO/IEC 9945-1 {2}]

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. [ISO/IEC 14392 {5}]

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 International Standard, but that can be pro-

vided. With respect to Strictly Conforming Applications, the word *may* means that the optional feature shall not be used. [ISO/IEC 9945-1 {2}]

2.2.1.5 shall: An indication of a requirement on the implementation or on Strictly Conforming Applications, where appropriate. [ISO/IEC 9945-1 {2}]

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 {2}]

2.2.1.7 supported: A condition regarding optional functionality.

Certain functionality in this International Standard is optional, but the interfaces to that functionality are always required. If the functionality is *supported*, the interfaces work as specified in this International Standard (except that they do not return the error condition indicated for the unsupported case). If the functionality is not *supported*, the interface shall always return the indication specified for this situation. [ISO/IEC 9945-1 {2}]

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 {2}]

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

Implementations (or other standards) may specify the result of using that value or causing that behavior. An application using such behaviors is using extensions, as defined in 1.3.2.3. [ISO/IEC 9945-1 {2}]

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

Implementations (or other standards) may specify the result of using that value or causing that behavior. An application requiring a specific behavior, rather than tolerating any behavior when using that functionality, is using extensions, as defined in 1.3.2.3. [ISO/IEC 9945-1 {2}]

2.2.2 General Terms

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

2.2.2.1 abstract class: An OM class of OM objects of which instances are forbidden. [ISO/IEC 14392 {5}]

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 {B1}). [ISO/IEC 14392 {5}]

2.2.2.3 access point: The point at which an abstract service is obtained. [ISO/IEC 9594-2 {B5}]

2.2.2.4 address: An unambiguous name, label, or number that identifies the location of a particular entity or service. [ISO/IEC 14392 {5}]

See also *presentation address* (2.2.2.66).

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 {B5}]

2.2.2.6 alias entry: A Directory entry, of Object Class “alias,” containing information used to provide an alternative name for an object. [ISO/IEC 14392 {5}]

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. [ISO/IEC 14392 {5}]

2.2.2.9 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. [ISO/IEC 14392 {5}]

2.2.2.10 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. [ISO/IEC 14392 {5}]

2.2.2.11 attribute: Used alone, means *directory attribute* as in 2.2.2.28 (see also *OM attribute*, 2.2.2.57).

2.2.2.12 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. [ISO/IEC 14392 {5}]

2.2.2.13 attribute type: Used alone, means *directory attribute type* as in 2.2.2.29 (see also *OM attribute type*, 2.2.2.58).

2.2.2.14 attribute value: Used alone, means *directory attribute value* as in 2.2.2.30 (see also *OM attribute value*, 2.2.2.59).

2.2.2.15 attribute value assertion: A proposition, which may be true, false, or undefined, according to the specified matching rules for the type, concerning the presence in an entry of an attribute value (or a distinguished value) of a particular type. [ISO/IEC 9594-2 {B5}]

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.

2.2.2.16 attribute value syntax: See *OM syntax* in 2.2.2.62.

2.2.2.17 basic encoding rules: A specific set of rules used to encode ASN.1 values as strings of octets. [ISO/IEC 14392 {5}]

2.2.2.18 cache: See *copy* in 2.2.2.23.

2.2.2.19 certificate: A synonym for *user certificate*, 2.2.2.84.

2.2.2.20 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 {B7}]

2.2.2.21 class: See *object class* in 2.2.2.54 and *OM class* in 2.2.2.60.

2.2.2.22 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. [ISO/IEC 14392 {5}]

See also *referral* (2.2.2.71).

2.2.2.23 copy:

- (1) A copy of an entry stored in other DSA(s) through bilateral agreement,
- (2) A locally and dynamically stored copy of an entry resulting from a request (a cache copy). [ISO/IEC 14392 {5}]

2.2.2.24 concrete class: An OM class of which instances are permitted. [ISO/IEC 14392 {5}]

2.2.2.25 descriptor: The means by which the client and service exchange an OM attribute value, and the integers that denote its representation, type, and syntax. [ISO/IEC 14392 {5}]

2.2.2.26 descriptor list: An ordered sequence of descriptors that is used to represent several OM attribute types and values. [ISO/IEC 14392 {5}]

2.2.2.27 directory: A repository of information about objects that provides directory services to its users, which allows its users access to the information.

2.2.2.28 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 {B5}]

2.2.2.29 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 {B5}]

2.2.2.30 directory attribute value [attribute value]: A particular instance of the class of information indicated by a directory attribute type. [ISO/IEC 9594-2 {B5}]

2.2.2.31 directory class: See *object class* in 2.2.2.54.

2.2.2.32 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 {B5}]

2.2.2.33 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 {B5}]

2.2.2.34 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 {B5}]

2.2.2.35 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. [ISO/IEC 14392 {5}]

2.2.2.36 directory syntax: See *attribute syntax* in 2.2.2.12.

2.2.2.37 directory system agent: An OSI application process that is part of the Directory. [ISO/IEC 9594-2 {B5}]

2.2.2.38 directory user agent: An OSI application process that represents a user accessing the Directory. [ISO/IEC 9594-2 {B5}]

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).

2.2.2.39 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 {B10}. [ISO/IEC 14392 {5}]

2.2.2.40 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 {B5}]

2.2.2.41 distinguished value: An attribute value in an entry that has been designated to appear in the relative distinguished name of the entry. [ISO/IEC 14392 {5}]

2.2.2.42 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 {B5}]

2.2.2.43 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 {B6}]

2.2.2.44 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*, 2.2.2.63.) [ISO/IEC 14392 {5}]

2.2.2.45 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. [ISO/IEC 14392 {5}]

2.2.2.46 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. [ISO/IEC 14392 {5}]

2.2.2.47 interface operation: See *operation* in 2.2.2.63.

2.2.2.48 invoke ID: An identifier used to distinguish one directory operation from all other outstanding operations. [ISO/IEC 14392 {5}]

2.2.2.49 knowledge reference: Knowledge that associates, either directly or indirectly, a DIT entry with the DSA in which it is located. [ISO/IEC 9594-4 {B7}]

2.2.2.50 leaf entry: A Directory entry that has no subordinates.

It can be an alias entry or an object entry. [ISO/IEC 14392 {5}]

2.2.2.51 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 {B5}]

2.2.2.52 nonspecific subordinate reference: A knowledge reference that holds information about a DSA that holds one or more unspecified subordinate entries. [ISO/IEC 9594-4 {B7}]

2.2.2.53 object: See *directory object* in 2.2.2.34 and *OM object* in 2.2.2.61.

2.2.2.54 object class [directory class]: An identified family of objects (or conceivable objects) that share certain characteristics. [ISO/IEC 14392 {5}]

2.2.2.55 object entry: See *entry* in 2.2.2.42.

2.2.2.56 object identifier: A value (distinguishable from all other such values) that is associated with an information object (see ISO/IEC 8824 {B1}). [ISO/IEC 14392 {5}]

2.2.2.57 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.

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

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

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

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

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

2.2.2.63 operation: A program abstraction with formal input and output parameters that are bound to objects or values in the application that invokes it. [ISO/IEC 14360 {3}]

2.2.2.64 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_result` interface operation. [ISO/IEC 14392 {5}]

2.2.2.65 package: A group of related OM classes.

2.2.2.66 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. [ISO/IEC 14392 {5}]

2.2.2.67 private OM object: The internal representation of the language-independent model of an OM object in a service, and thus unspecified. [ISO/IEC 14360 {3}]

2.2.2.68 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 {2}]

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

2.2.2.69 public OM object: The representation by the client of an object in a particular programming language binding. [ISO/IEC 14360 {3}]

2.2.2.70 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 {B5}]

2.2.2.71 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 {B7}]

2.2.2.72 relative distinguished name: A set of AVAs, each of which is true, concerning the distinguished values of a particular entry. [ISO/IEC 9594-2 {B5}]

2.2.2.73 replication: The process by which copies of entries are created and maintained. [ISO/IEC 14392 {5}]

2.2.2.74 result: Information that is returned from an interface operation or a directory operation and that constitutes the outcome of the processing that was performed. [ISO/IEC 14392 {5}]

2.2.2.75 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 {B5}]

2.2.2.76 service controls: Parameters conveyed as part of a directory operation that constrain various aspects of its performance. [ISO/IEC 9594-3 {B6}]

2.2.2.77 session: A sequence of directory operations requested by a particular user of a particular DUA using the same session OM object. [ISO/IEC 14392 {5}]

2.2.2.78 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. [ISO/IEC 14392 {5}]

2.2.2.79 subordinate: The converse of superior. [ISO/IEC 9594-2 {B5}]

2.2.2.80 superior (applying to entry or directory object): Immediately superior, or superior to one that is immediately superior (recursively). [ISO/IEC 9594-2 {B5}]

2.2.2.81 syntax: See *attribute syntax* in 2.2.2.12 and *OM syntax* in 2.2.2.62.

2.2.2.82 thread: A single sequential flow of control within a process. [ISO/IEC 14392 {5}]

2.2.2.83 user: The entity or person that accesses the directory, used in this International Standard to refer to the application program that is using the interface. [ISO/IEC 14392 {5}]

2.2.2.84 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 {B10}]

2.2.2.85 value: See *attribute value* in 2.2.2.14 and *OM attribute value* in 2.2.2.59.

2.2.2.86 workspace: A repository for instances of OM classes in the closures of one or more packages associated with the workspace. [ISO/IEC 14392 {5}]

2.2.3 Abbreviations

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

2.2.3.1 API: Application Program Interface.

2.2.3.2 ASN.1: Abstract Syntax Notation One.

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.

This organization has now been renamed as the Telecommunications Standardization Sector of the International Telecommunications Union (ITU-T).

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 this International Standard).

2.2.3.16 DS Package: Directory Service Package.

2.2.3.17 DSA: Directory System Agent.

534 **2.2.3.18 DSP:** Directory System Protocol.

535 **2.2.3.19 DUA:** Directory User Agent.

536 **2.2.3.20 IA5:** International Alphabet No. 5.

537 **2.2.3.21 ID:** Identifier.

538 **2.2.3.22 the Interface:** The interface to directory services defined in this Interna-
539 tional Standard.

540 **2.2.3.23 ISDN:** Integrated Services Digital Network.

541 **2.2.3.24 ISO:** International Organisation for Standardization.

542 **2.2.3.25 MDUP:** MHS Directory User Package.

543 **2.2.3.26 MHS:** Message Handling System (as in ISO/IEC 10021).

544 **2.2.3.27 MS:** Message Store.

545 **2.2.3.28 NBP:** Nonbasic Parameter.

546 **2.2.3.29 O/R:** Originator/Recipient.

547 **2.2.3.30 OM:** OSI abstract data Manipulation.

548 **2.2.3.31 OSI:** Open Systems Interconnection.

549 **2.2.3.32 ISO/IEC 9945-1:** ISO/IEC 9945-1 {2}.

550 **2.2.3.33 POSIX.3:** IEEE Std 1003.3-1991 {B15}.

551 **2.2.3.34 RDN:** Relative Distinguished Name.

552 **2.2.3.35 X.500 directory standards:** ISO/IEC 9954 and the 1988 CCITT X.500
553 Series recommendations ({B4}–{B10}).

554 **2.2.3.36 SAP:** Strong Authentication Package.

Section 3: The Generic Interface

3.1 Introduction

NOTE: This section defines the C language binding corresponding to the interface operations, interface class definitions, and error class definitions of the language-independent specification in ISO/IEC 14392 {5}. The C language binding corresponding to the interface operations is in 3.2. The C language binding corresponding to the interface class definitions and error class definitions is in 3.3. These subclauses are followed by a subclause on the `<xds.h>` header.

Some of the C language datatypes appearing in this section are defined in ISO/IEC 14364 {4}. The types in question are those whose names begin with “OM_”. For the purpose of this International Standard, the definitions given in ISO/IEC 14364 {4} shall apply.

In the C binding, instances of classes defined in ISO/IEC 14392 {5} shall be represented by data structures of C language type *OM_object*, defined in ISO/IEC 14364 {4}.

NOTE: No additional type definitions for these classes are contained in this section.

All object identifiers shall be represented by constants in the C binding. These constants can be used with the macros defined in ISO/IEC 14364 {4}.

NOTE: Constants representing object identifiers required for the Directory Service package are defined in this section.

Every application program that makes use of a class or other object identifier name shall explicitly import it into every compilation unit (C source program) that uses it. Each such class or object identifier name shall be explicitly exported from just one compilation unit.

NOTE: The descriptions of the OM_IMPORT and OM_EXPORT macros in ISO/IEC 14364 {4} contain an explanation.

3.2 Interface Functions

3.2.1 Overview

For each operation of ISO/IEC 14392 {5}, there shall be a corresponding function of the C binding. Each C function shall have the same name as the operation to which it corresponds.

Except for *ds_initialize()* and *ds_version()*, the arguments of each C function shall correspond to the parameters of the corresponding language-independent specification operation. For input parameters, the function argument name shall be the same as the parameter name. For output parameters, the function argument name shall be derived from the parameter name by appending “*_return*” to it.

The processing performed by each C language function shall be exactly as described for the corresponding operation of ISO/IEC 14392 {5}.

Except for *ds_initialize()*, the C function shall return a result of C language type *DS_status*, corresponding to the status datatype *ds_status_type* of the equivalent operation of ISO/IEC 14392 {5}.

The error indications that each C function can return shall correspond to the errors described for the equivalent operation of ISO/IEC 14392 {5}.

NOTES:

- (1) Except for *ds_initialize()* and *ds_version()*, this section contains synopses of the C functions but no detailed descriptions of them.
- (2) Descriptions of the language-independent operations are contained in ISO/IEC 14392 {5}.

3.2.2 Common Datatypes

```
typedef OM_private_object DS_status;
```

The datatype of most of the C functions defined in this section, corresponding to the datatype *ds_status_type* of ISO/IEC 14392 {5}.

3.2.3 Abandon

```
DS_status ds_abandon(
    OM_private_object session,
    OM_sint          invoke_id
);
```

This function shall correspond to the operation *ds_abandon* of ISO/IEC 14392 {5} and shall abandon the result of a pending, asynchronously-executing directory operation.

3.2.4 Add Entry

```

58 DS_status ds_add_entry(
59     OM_private_object    session,
60     OM_private_object    context,
61     OM_object            name,
62     OM_object            entry,
63     OM_sint              * invoke_id_return
64 );

```

65 This function shall correspond to the operation *ds_add_entry* of ISO/IEC 14392 {5}
 66 and shall add a leaf entry to the directory information tree.

3.2.5 Bind

```

68 DS_status ds_bind(
69     OM_object            session,
70     OM_workspace        workspace,
71     OM_private_object    * bound_session_return
72 );

```

73 This function shall correspond to the operation *ds_bind* of ISO/IEC 14392 {5} and
 74 shall open a session with the directory.

3.2.6 Compare

```

76 DS_status ds_compare(
77     OM_private_object    session,
78     OM_private_object    context,
79     OM_object            name,
80     OM_object            ava,
81     OM_private_object    * result_return,
82     OM_sint              * invoke_id_return
83 );

```

84 This function shall correspond to the operation *ds_compare* of ISO/IEC 14392 {5}
 85 and shall compare a purported attribute value with the attribute value stored in
 86 the directory for a particular entry.

3.2.7 Initialize

```

88 OM_workspace ds_initialize(
89     void
90 );

```

91 This function shall correspond to the operation *ds_initialize* of ISO/IEC 14392 {5}
 92 and shall initialize the interface.

93 In the function *ds_initialize()* of the C language binding, the result of type
 94 *OM_workspace* returned when the function succeeds shall correspond to output
 95 parameter *workspace* of the operation *ds_initialize* of ISO/IEC 14392 {5}, which
 96 returns result **success**. When function *ds_initialize()* of the C language binding
 97 fails, it shall return the result {DS_NULL_RESULT}, corresponding to the result
 98 **no_workspace** of the operation *ds_initialize* of ISO/IEC 14392 {5}.