

INTERNATIONAL  
STANDARD

**ISO/IEC**  
**9594-2**

First edition  
1990-12-15

---

---

**Information technology — Open Systems  
Interconnection — The Directory —**

**Part 2:  
Models**

*Technologies de l'information — Interconnexion de systèmes ouverts — L'annuaire —*

*Partie 2: Modèles*



Reference number  
ISO/IEC 9594-2 : 1990 (E)

## Contents

	Page
Foreword .....	iii
Introduction .....	iv
<b>SECTION 1: GENERAL</b>	<b>1</b>
1 Scope .....	1
2 Normative references.....	1
3 Definitions .....	1
4 Abbreviations.....	2
<b>SECTION 2: DIRECTORY MODEL</b>	<b>2</b>
5 Directory Model.....	2
<b>SECTION 3: INFORMATION MODEL</b>	<b>3</b>
6 Directory Information Base .....	3
7 Directory Entries .....	5
8 Names .....	6
9 Directory Schema .....	8
<b>SECTION 4: SECURITY MODEL</b>	<b>13</b>
10 Security.....	13
Annex A — The Mathematics of Trees .....	14
Annex B — Object Identifier Usage .....	15
Annex C — Information Framework in ASN.1 .....	16
Annex D — Alphabetical Index of Definitions.....	19
Annex E — Name Design Criteria.....	20
Annex F — Access Control.....	21

© ISO/IEC 1990

All rights reserved. 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

## 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 9594-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

ISO/IEC 9594 consists of the following parts, under the general title *Information technology — Open Systems Interconnection — The Directory*:

- *Part 1: Overview of concepts, models and services*
- *Part 2: Models*
- *Part 3: Abstract service definition*
- *Part 4: Procedures for distributed operation*
- *Part 5: Protocol specifications*
- *Part 6: Selected attribute types*
- *Part 7: Selected object classes*
- *Part 8: Authentication framework*

Annexes B and C form an integral part of this part of ISO/IEC 9594. Annexes A, D, E and F are for information only.

## Introduction

**0.1** This part of ISO/IEC 9594, together with the other parts, has been produced to facilitate the interconnection of information processing systems to provide directory services. A set of such systems, together with the directory information which they hold, can be viewed as an integrated whole, called the *Directory*. The information held by the *Directory*, collectively known as the Directory Information Base (DIB), is typically used to facilitate communication between, with or about objects such as application entities, people, terminals and distribution lists.

**0.2** The *Directory* plays a significant role in Open Systems Interconnection, whose aim is to allow, with a minimum of technical agreement outside of the interconnection standards themselves, the interconnection of information processing systems:

- from different manufacturers;
- under different managements;
- of different levels of complexity; and
- of different ages.

**0.3** This part of ISO/IEC 9594 provides a number of different models for the *Directory* as a framework for the other parts of ISO/IEC 9594. The models are the overall (functional) model; the organizational model; the security model; and the information model. The latter describes the manner in which the *Directory* organizes the information it holds. It describes, for example, how information about objects is grouped to form directory entries for those objects and how that information provides names for objects.

**0.4** Annex A summarizes the mathematical terminology associated with tree structures.

**0.5** Annex B summarizes the usage of ASN.1 object identifiers in ISO/IEC 9594.

**0.6** Annex C provides the ASN.1 module which contains all of the definitions associated with the information framework.

**0.7** Annex D lists alphabetically the terms defined in this document.

**0.8** Annex E describes some criteria that can be considered in designing names.

**0.9** Annex F describes guidelines for access control.

**0.10** Annex G describes the derivation of a subclass.

# Information technology — Open Systems Interconnection — The Directory —

## Part 2: Models

### SECTION 1: GENERAL

#### 1 Scope

1.1 The models defined in this part of ISO/IEC 9594 provide a conceptual and terminological framework for the other parts of ISO/IEC 9594 which define various aspects of the Directory.

1.2 The functional and organizational models define ways in which the Directory can be distributed, both functionally and administratively.

1.3 The security model defines the framework within which security features, such as access control, are provided in the Directory.

1.4 The information model describes the logical structure of the DIB. From this viewpoint, the fact that the Directory is distributed, rather than centralized, is not visible. The other parts of ISO/IEC 9594 make use of the concepts of the information framework. Specifically:

- a) the service provided by the Directory is described (in ISO/IEC 9594-3) in terms of the concepts of the information framework: this allows the service provided to be somewhat independent of the physical distribution of the DIB;
- b) the distributed operation of the Directory is specified (in ISO/IEC 9594-4) so as to provide that service, and therefore maintain that logical information structure, given that the DIB is in fact highly distributed.

#### 2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9594. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 9594 are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 7498:1984, *Information Processing Systems — Open Systems Interconnection — Basic Reference Model.*

ISO/IEC 8824:1990, *Information Technology — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1).*

ISO/IEC 9594-1:1990, *Information Technology — Open Systems Interconnection — The Directory — Part 1: Overview of Concepts, Models and Services.*

ISO/IEC 9594-3:1990, *Information Technology — Open Systems Interconnection — The Directory — Part 3: Abstract Service Definition.*

ISO/IEC 9594-4:1990, *Information Technology — Open Systems Interconnection — The Directory — Part 4: Procedures for Distributed Operation.*

ISO/IEC 9594-5:1990, *Information Technology — Open Systems Interconnection — The Directory — Part 5: Protocol Specifications.*

ISO/IEC 9594-6:1990, *Information Technology — Open Systems Interconnection — The Directory — Part 6: Selected Attribute Types.*

ISO/IEC 9594-7:1990, *Information Technology — Open Systems Interconnection — The Directory — Part 7: Selected Object Classes.*

ISO/IEC 9594-8:1990, *Information Technology — Open Systems Interconnection — The Directory — Part 8: Authentication Framework.*

#### 3 Definitions

Definitions of terms are included at the beginning of individual clauses, as appropriate. An index of these terms is provided in annex D for easy reference.

## 4 Abbreviations

ADDMD	Administration Directory Management Domain
AVA	attribute value assertion
DIB	Directory Information Base

DIT	Directory Information Tree
DMD	Directory Management Domain
DSA	Directory System Agent
DUA	Directory User Agent
PRDMD	Private Directory Management Domain
RDN	Relative Distinguished Name

## SECTION 2: DIRECTORY MODEL

### 5 Directory Model

#### 5.1 Definitions

**5.1.1 Access Point** : the point at which an abstract service is obtained

**5.1.2 Administration Directory Management Domain (ADDMD)** : A DMD which is managed by an Administration.

**Note** — The term Administration denotes a public telecommunications administration or other organization offering public telecommunications services;

**5.1.3 Administrative Authority** : An entity which has administrative control over all entries stored within a single Directory System Agent.

**5.1.4 the Directory** : A repository of information about objects, and which provides directory services to its users which allow access to the information.

**5.1.5 Directory Management Domain (DMD)** : A collection of one or more DSAs and zero or more DUAs which is managed by a single organization.

**5.1.6 Directory System Agent (DSA)** : An OSI application process which is part of the Directory.

**5.1.7 (Directory) user** : The end user of the Directory, i.e., the entity or person which accesses the Directory.

**5.1.8 Directory User Agent (DUA)** : An OSI application process which represents a user in accessing the Directory.

**Note** — DUAs may also provide a range of local facilities to assist users compose queries and interpret the responses.

**5.1.9 Private Directory Management Domain (PRDMD)** : A DMD which is managed by an organization other than an Administration\*.

#### 5.2 The Directory and its Users

**5.2.1** A directory user (e.g., a person or an application-process) obtains directory services by accessing the Directory. More precisely, it is a *Directory User Agent*

(DUA), which actually accesses the Directory and interacts with it to obtain the service on behalf of a particular user. The Directory provides one or more *access points* at which such accesses can take place. These concepts are illustrated in figure 1.

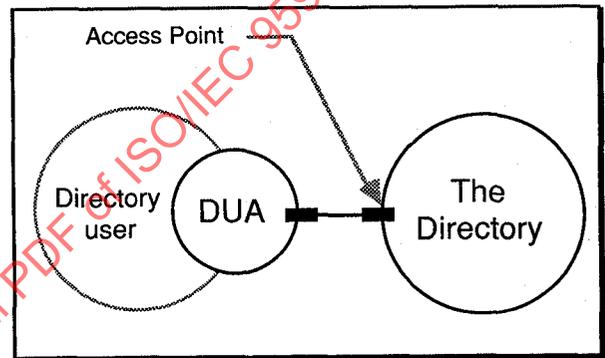


Figure 1 — Access to the Directory

**5.2.2** The services provided by the Directory are defined in ISO/IEC 9594-3.

**5.2.3** The Directory is a repository of information about objects, and the directory services it provides to its users are concerned with various kinds of access to this information. The information is collectively known as the *Directory Information Base (DIB)*. A model for the DIB is defined in section three of this part of ISO/IEC 9594.

**5.2.4** A DUA is manifested as an application-process. Each DUA represents precisely one directory user.

#### Notes

1. Some open systems may provide a centralized DUA function retrieving information for the actual users (application-processes, persons, etc.). This is transparent to the Directory.
2. The DUA functions and a DSA (see 5.3.1) can be within the same open system, and it is an implementation choice whether to make one or more DUAs visible within the OSI Environment as application-entities.
3. A DUA may exhibit local behavior and structure which is outside the scope of ISO/IEC 9594. For example, a DUA which represents a human directory user may provide a range of local facilities to assist its user to compose queries and interpret the responses.

### 5.3 Functional Model

**5.3.1** The Directory is manifested as a set of one or more application-processes known as *Directory System Agents (DSAs)*, each of which provides zero, one, or more of the access points. This is illustrated in figure 2. Where the Directory is composed of more than one DSA, it is said to be *distributed*. The procedures for the operation of the Directory when it is distributed are specified in ISO/IEC 9594-4.

**Note** — A DSA will likely exhibit local behavior and structure which is outside the scope of ISO/IEC 9594. For example, a DSA which is responsible for holding some or all of the information in the DIB will normally do so by means of a database, the interface to which is a local matter.

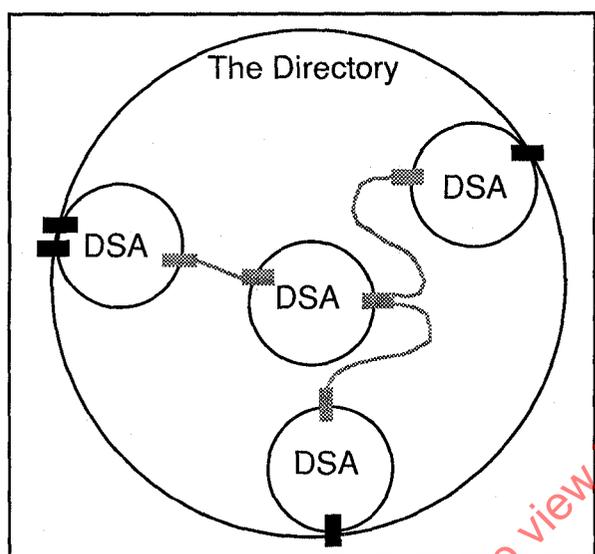


Figure 2 — The Directory Provided by Multiple DSAs

**5.3.2** A particular pair of application-processes which need to interact in the provision of directory services (either a DUA and a DSA, or two DSAs) may be located in

different open systems. Such an interaction is carried out by means of OSI Directory protocols, as specified in ISO/IEC 9594-5.

### 5.4 Organizational Model

**5.4.1** A set of one or more DSAs and zero or more DUAs managed by a single organization may form a *Directory Management Domain (DMD)*.

**Note** - The organization which manages a DMD may be an Administration\* (i.e., a public telecommunications administration or other organization offering public telecommunications services) in which case the DMD is said to be an Administration DMD (ADDMD); otherwise it is a Private DMD (PRDMD). It should be recognized that the provision of support for private directory systems by CCITT members falls within the framework of national regulations. Thus, the technical possibilities described may or may not be offered by an Administration\* which provides directory services. The internal operation and configuration of private DMDs is not within the scope of envisaged CCITT Recommendations.

**5.4.2** Management of a DUA by a DMD implies an ongoing responsibility for service to that DUA, e.g., maintenance, or in some cases ownership, by the DMD.

**5.4.3** The organization concerned may or may not elect to make use of ISO/IEC 9594 to govern any interactions among DUAs and DSAs which are wholly within the DMD.

**5.4.4** Each DSA is administered by an Administrative Authority. This entity has control over all object entries and alias entries stored by that DSA. This includes responsibilities for the Directory schema being used to guide the creation and modification of entries (see clause 9). The structure and allocation of names is the responsibility of a naming authority (see 8.1.6) and the role of the Administrative Authority is to implement these naming structures in the schema.

## SECTION 3: INFORMATION MODEL

### 6 Directory Information Base

#### 6.1 Definitions

**6.1.1** *alias entry* : an entry of the class "alias" containing information used to provide an alternative name for an object.

**6.1.2** *Directory Information Base (DIB)* : the complete set of information to which the Directory provides access, and which includes all of the pieces of information which can be read or manipulated using the operations of the Directory.

**6.1.3** *Directory Information Tree (DIT)*: the DIB considered as a tree, whose vertices (other than the root) are the Directory entries.

**Note** - The term DIT is used instead of DIB only in contexts where the tree structure of the information is relevant.

**6.1.4** *(Directory) entry* : a part of the DIB which contains information about an object.

**6.1.5** *immediate superior* (noun) : relative to a particular entry or object (it shall be clear from the context which is intended), the immediately superior entry or object.

### 6.1.6 *immediately superior*

(*entry*): relative to a particular entry - an entry which is at the initial vertex of an arc in the DIT whose final vertex is that of the particular entry;

(*object*): relative to a particular object - an object whose *object entry* is the immediate superior of *any* of the entries (object or alias) for the second object.

**6.1.7 *object (of interest)*** : anything in some 'world', generally the world of telecommunications and information processing or some part thereof, which is identifiable (can be named), and which it is of interest to hold information on in the DIB.

**6.1.8 *object class*** : an identified family of objects (or conceivable objects) which share certain characteristics.

**6.1.9 *object entry*** : an entry which is the primary collection of information in the DIB about an object, and which can therefore be said to represent that object in the DIB.

**6.1.10 *subclass***: relative to a superclass -- an object class derived from a superclass. The members of the subclass share all the characteristics of another object class (the superclass) and additional characteristics possessed by none of the members of that object class (the superclass).

**6.1.11 *subordinate/inferior***: the converse of superior.

**6.1.12 *superclass***: relative to a subclass -- an object class from which a subclass is derived.

**6.1.13 *superior*** : (applying to entry or object) immediately superior, or superior to one which is immediately superior (recursively).

## 6.2 Objects

**6.2.1** The purpose of the Directory is to hold, and provide access to, information about *objects of interest (objects)* which exist in some 'world'. An object can be anything in that world which is identifiable (can be named).

### Notes

1. The 'world' is generally that of telecommunications and information processing or some part thereof.
2. The objects known to the Directory may not correspond exactly with the set of 'real' things in the world. For example, a real-world person may be regarded as two different objects, a business person and a residential person, as far as the Directory is concerned. The mapping is not defined in ISO/IEC 9594, but is a matter for the users and providers of the Directory in the context of their applications.

**6.2.2** The complete set of information to which the Directory provides access is known as the *Directory Information Base (DIB)*. All of the pieces of information

which can be read or manipulated by the operations of the Directory are considered to be included in the DIB.

**6.2.3** An *object class* is an identified family of objects (or conceivable objects) which share certain characteristics. Every object belongs to at least one class. An *object class* may be a *subclass* of another object class, in which case the members of the former class (the subclass) are also considered to be members of the latter (the superclass). There may be subclasses of subclasses, etc., to an arbitrary depth.

## 6.3 Directory Entries

**6.3.1** The DIB is composed of *Directory entries (entries)* each containing information about (describing) a single object.

**6.3.2** For any particular object there is precisely one *object entry*, this being the primary collection of information in the DIB about that object. The object entry is said to represent the object.

**6.3.3** For any particular object there may, in addition to the object entry, be one or more *alias entries* for that object, which are used to provide alternative names (see 8.5).

**6.3.4** The structure of directory entries is depicted in figure 3 and described in 7.2.

**6.3.5** Each entry contains an indication of the object class and the superclasses of that object class with which the entry is associated. In the case of an object entry, this indicates the class(es) to which the object belongs. In the case of an alias entry, this indicates (by means of a special object class, "alias" (defined in 9.4.8.2)) that it is in fact an alias entry, and may also indicate to which subclass(es) of the alias object class the entry belongs.

## 6.4 The Directory Information Tree (DIT)

**6.4.1** In order to satisfy the requirements for the distribution and management of a potentially very large DIB, and to ensure that objects can be unambiguously named (see clause 8), and their entries found, a flat structure of entries is not likely to be feasible. Accordingly, the hierarchical relationship commonly found among objects (e.g., a person works for a department, which belongs to an organization, which is headquartered in a country) can be exploited, by the arrangement of the entries into a tree, known as the *Directory Information Tree (DIT)*.

**Note** — An introduction to the concepts and terminology of tree structures can be found in annex A.

**6.4.2** The component parts of the DIT have the following interpretations:

- a) the vertices are the entries. Object entries may be either leaf or non-leaf vertices, whereas alias entries

are always leaf vertices. The root is not an entry as such, but can, where convenient to do so (e.g., in the definitions of (b) and (c) below), be viewed as a null object entry (see (d) below);

- b) the arcs define the relationship between vertices (and hence entries). An arc from vertex A to vertex B means that that the entry at A is the *immediately superior entry* (*immediate superior*) of the entry at B, and conversely, that the entry at B is an *immediately subordinate entry* (*immediate subordinate*) of the entry at A. The *superior entries* (*superiors*) of a particular entry are its immediate superior together with its superiors (recursively). The *subordinate entries* (*subordinates*) of a particular entry are its immediate subordinates together with their subordinates (recursively);
- c) the object represented by an entry is, or is closely associated with, the naming authority (see clause 8) for its subordinates;
- d) the root represents the highest level of naming authority for the DIB.

**6.4.3** A superior/subordinate relationship between objects can be derived from that between entries. An object is an *immediately superior object* (*immediate superior*) of another object if and only if the object entry for the first object is the immediate superior of any of the entries for the second object. The terms *immediately subordinate object*, *immediate subordinate*, *superior* and *subordinate* (applied to objects) have their analogous meanings.

**6.4.4** Permitted superior/subordinate relationships among objects are governed by the DIT structure definitions (see 9.2).

## 7 Directory Entries

### 7.1 Definitions

**7.1.1** *attribute* : the information of a particular type concerning an object and appearing in an entry describing that object in the DIB.

**7.1.2** *attribute type* : that component of an attribute which indicates the class of information given by that attribute.

**7.1.3** *attribute value* : a particular instance of the class of information indicated by an attribute type.

**7.1.4** *attribute value assertion* : a proposition, which may be true, false, or undefined, concerning the values (or perhaps only the distinguished values) of an entry.

**Note** - In this document the notation "string1 = string2" is used to write down examples of attribute value assertions. In this notation, "string1" is an abbreviation for the 'name' of the attribute type, and "string2" is a textual representation of a

suitable value. Although the attribute types in the examples are often based upon real types, such as those defined in ISO/IEC 9594-7 (e.g., "C" stands for "Country", CN for "Common Name"), this is not strictly necessary for the purposes of this document, as the Directory is usually unaware of the meanings of the attribute types in use.

**7.1.5** *distinguished value* : an attribute value in an entry which has been designated to appear in the relative distinguished name of the entry.

### 7.2 Overall Structure

**7.2.1** As depicted in figure 3, an entry consists of a set of *attributes*.

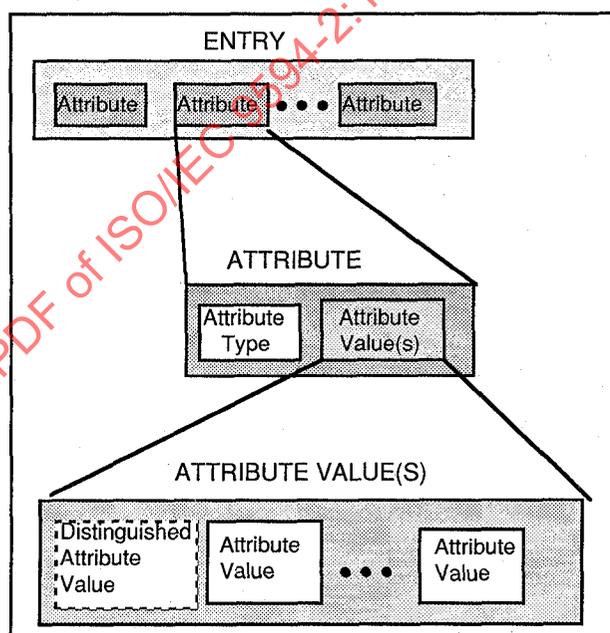


Figure 3 — Structure of an Entry

**7.2.2** Each attribute provides a piece of information about, or describes a particular characteristic of, the object to which the entry corresponds.

**Note** - Examples of attributes which might be present in an entry include naming information such as the object's personal name, and addressing information, such as its telephone number.

**7.2.3** An attribute consists of an *attribute type*, which identifies the class of information given by an attribute, and the corresponding *attribute value(s)*, which are the particular instances of that class appearing in the entry

```
Attribute ::=
SEQUENCE {
    type           AttributeType,
    values         SET OF AttributeValue
    -- at least one value is required --
}
```

### 7.3 Attribute Types

**7.3.1** Some attribute types will be internationally standardized. Other attribute types will be defined by

national administrative authorities and private organizations. This implies that a number of separate authorities will be responsible for assigning types in a way that ensures that each is distinct from all other assigned types. This is accomplished by identifying each attribute type with an object identifier when the type is defined (as described in 9.5):

**AttributeType ::= OBJECT IDENTIFIER**

**7.3.2** All attributes in an entry shall be of distinct attribute types.

**7.3.3** There are a number of attribute types which the Directory knows about and uses for its own purposes. They include:

- a) **ObjectClass** — An attribute of this type appears in every entry, and indicates the object class and superclass(es) to which the object belongs.
- b) **AliasedObjectName** — An attribute of this type appears in every alias entry, and holds the distinguished name (see 8.5) of the object which this alias entry describes.

These attributes are (partially) defined in 9.5.4.

**7.3.4** The types of attributes which shall or which may appear within an entry (other than as mentioned in 7.3.3) are governed by rules applying to the indicated object class(es).

## 7.4 Attribute Values

**7.4.1** Defining an attribute type (see 9.5) also involves specifying the syntax, and hence data type, to which every value in such attributes shall conform. This could be any data type:

**AttributeValue ::= ANY**

**7.4.2** At most one of the values of an attribute may be designated as a *distinguished value*, in which case the attribute value appears in the relative distinguished name (see 8.3) of the entry.

**7.4.3** An *attribute value assertion (AVA)* is a proposition, which may be true, false, or undefined, concerning the values (or perhaps only the distinguished values) of an entry. It involves an attribute type and an attribute value

**AttributeValueAssertion :=  
SEQUENCE {AttributeType, AttributeValue}**

and is:

- a) undefined, if any of the following holds:
  - 1) the attribute type is unknown,

- 2) the attribute syntax for the type has no equality matching rule,
- 3) the value does not conform to the data type of the attribute syntax;

**Note** — 2) and 3) normally indicate a faulty AVA; 1), however, may occur as a local situation (e.g., a particular DSA has not registered that particular attribute type).

- b) true, if the entry contains an attribute of that type, one of whose values matches that value (if the assertion is concerned only with distinguished values, then the matched value shall be the distinguished one);

**Note** — The matching of values is for equality, and involves the matching rule associated with the attribute syntax.

- c) false, otherwise.

## 8 Names

### 8.1 Definitions

**8.1.1** *alias, alias name*: a name for an object, provided by the use of one or more alias entries in the DIT.

**8.1.2** *dereferencing*: replacing the alias name for an object by the object's distinguished name.

**8.1.3** *distinguished name* (of an object): one of the names of the object, formed from the sequence of the RDNs of the object entry and each of its superior entries.

**8.1.4** (*directory*) *name*: a construct that singles out a particular 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 which unambiguously denotes the object).

**8.1.5** *purported name*: a construct which is syntactically a name, but which has not (yet) been shown to be a valid name.

**8.1.6** *naming authority*: an authority responsible for the allocation of names. Each object whose object entry is located at a non-leaf vertex in the DIT is, or is closely associated with, a naming-authority.

**8.1.7** *relative distinguished name (RDN)*: a set of attribute value assertions, each of which is true, concerning the distinguished values of a particular entry.

### 8.2 Names in General

**8.2.1** A (*directory*) *name* is a construct that identifies a particular object from among the set of all objects. A name shall be unambiguous, that is, denote just one object. However, a name need not be unique, that is be the only name that unambiguously denotes the object.

8.2.2 Syntactically, each name for an object is an ordered sequence of relative distinguished names (see 8.3).

```
Name ::=
    CHOICE (
        -- only one possibility for now --
        RDNSequence )
RDNSequence ::= SEQUENCE OF
    RelativeDistinguishedName
DistinguishedName ::= RDNSequence
```

Note- Names which are formed in other ways than as described herein are a possible future extension.

8.2.3 The null sequence is the name for the root of the tree.

8.2.4 Each initial subsequence of the name of an object is also the name of an object. The sequence of objects so identified, starting with the root and ending with the object being named, is such that each is the immediate superior of that which follows it in the sequence.

8.2.5 A *purported name* is a construct which is syntactically a name, but which has not (yet) been shown to be a valid name.

### 8.3 Relative Distinguished Names

8.3.1 Each entry has a unique *relative distinguished name (RDN)*. An RDN consists of a set of attribute value assertions, each of which is true, concerning the distinguished values of the entry.

```
RelativeDistinguishedName ::=
    SET OF AttributeValueAssertion
```

The set contains exactly one assertion about each distinguished value in the entry.

8.3.2 The RDNs of all of the entries with a particular immediate superior are distinct. It is the responsibility of the relevant naming authority for that entry to ensure that this is so by appropriately assigning distinguished attribute values.

Note - Frequently, an entry will contain a single distinguished value (and the RDN will thus comprise a single AVA); however, under certain circumstances (in order to differentiate), additional values (and hence AVAs) may be used.

8.3.3 The RDN for an entry is chosen when the entry is created. A single value instance of any attribute type may form part of the RDN, depending on the nature of the object class denoted. Allocation of RDNs is considered an administrative undertaking that may or may not require some negotiation between involved organizations or administrations. This part of ISO/IEC 9594 does not provide such a negotiation mechanism, and makes no assumption as to how it is performed. The RDN may be modified if necessary by complete replacement.

Note — RDNs are intended to be long-lived so that the users of the Directory can store the distinguished names of objects (e.g.,

in the Directory itself) without concerns for their obsolescence. Thus RDNs should be changed cautiously.

### 8.4 Distinguished Names

8.4.1 The *distinguished name* of a given object is defined as being the sequence of the RDNs of the entry which represents the object and those of all of its superior entries (in descending order). Because of the one to one correspondence between objects and object entries, the distinguished name of an object can be considered to also identify the object entry.

#### Notes

1 It is preferable that the distinguished names of objects which humans have to deal with be user-friendly.

2 ISO 7498-3 defines the concept of a primitive name. A distinguished name can be used as a primitive name for the object it identifies because: (a) it is unambiguous, (b) it is unique, and (c) the semantics of its internal structure (a sequence of RDNs) need not (but of course may) be understood by the user of the Directory.

3 Because only the object entry and its superiors are involved, distinguished names of objects can never involve alias entries.

8.4.2 It proves convenient to define the 'distinguished name' of the root and of an alias entry, although in neither case is the name also the distinguished name of an object. The distinguished name of the root is defined to be the null sequence. The distinguished name of an alias entry is defined to be the sequence of RDNs of the alias entry and those of all of its superior entries (in descending order).

8.4.3 An example which illustrates the concepts of RDN and distinguished name appears in figure 4.

	RDN	Distinguished Name
ROOT		{ }
Countries	C=GB	{C=GB}
Organizations	O=Telecom	{C=GB, O=Telecom}
Organizational Units	(OU=Sales, L=Ipswich)	{C=GB, O=Telecom, (OU=Sales, L=Ipswich)}
People	CN=Smith	{C=GB, O=Telecom, (OU=Sales, L=Ipswich), CN=Smith}

Figure 4 — Determination of Distinguished Names

### 8.5 Alias Names

8.5.1 An *alias*, or an *alias name*, for an object is a name at least one of whose RDNs is that of an alias entry. Aliases permit object entries to achieve the effect of having multiple immediate superiors. Therefore, aliases provide a basis for alternative names.

**8.5.2** Just as the distinguished name of an object expresses its principal relationship to some hierarchy of objects, so an alias expresses (in the general case) an alternative relationship to an different hierarchy of objects.

**8.5.3** An object with an entry in the DIT may have zero or more aliases. It, therefore, follows that several alias entries may point to the same object entry. An alias entry may point to an object entry that is not a leaf entry. Only object entries may have aliases. Thus aliases of aliases are not permitted.

**8.5.4** An alias entry shall have no subordinates, that is, an alias entry is a leaf entry.

**8.5.5** The Directory makes use of the aliased object name attribute in an alias entry to identify and to find the corresponding object entry.

## 9 Directory Schema

### 9.1 Definitions

**9.1.1** *Directory Schema*: The set of rules and constraints concerning DIT structure, object class definitions, attribute types and syntaxes which characterize the DIB.

**9.1.2** *DIT Structure Rule*: A rule, forming part of the Directory Schema, which relates an object class (the subordinate) to another object class (the superior) and which allows an entry of the former class to be immediately subordinate to one of the latter classes in the DIT. The rule also governs the attribute type(s) permitted to appear in the (subordinate) entry's RDN, and may impose additional conditions. The schema may contain many such rules.

### 9.2 Overview

**9.2.1** The Directory Schema is a set of definitions and constraints concerning the structure of the DIT and the possible ways entries are named, the information that can be held in an entry, and the attributes used to represent that information.

#### Notes

- 1 The schema enables the directory system to, for example:
  - prevent the creation of subordinate entries of the wrong object-class (e.g., a country as a subordinate of a person);
  - prevent the addition of attribute-types to an entry inappropriate to the object-class (e.g., a serial number to a person's entry);
  - prevent the addition of an attribute value of a syntax not matching that defined for the attribute-type (e.g., a printable string to a bit string).

- 2 Dynamic mechanisms for the management of the directory schema are not presently provided by ISO/IEC 9594.

**9.2.2** Formally, the Directory Schema comprises a set of

- a) *DIT Structure* definitions (rules) that define the distinguished names that entries may have and the ways in which they may be related to one another through the DIT.
- b) *Object Class* definitions that define the set of mandatory and optional attributes that shall be present, and may be present, respectively, in an entry of a given class (see 6.2.3 of ISO/IEC 9594-2).
- c) *Attribute Type* definitions that identify the object identifier by which an attribute is known, its syntax, and whether it is permitted to have multiple values.
- d) *Attribute Syntax* definitions that define for each attribute the underlying ASN.1 data type and matching rules.

Figure 5 summarizes the relationships between the schema definitions on the one side, and the DIT, directory entries, attributes, and attribute values on the other.

**9.2.3** The Directory Schema is distributed, like the DIB itself. Each Administrative Authority establishes the part of the schema that will apply for those portions of the DIB administered by the authority.

**Note** — Distribution of schema information across DSAs managed by different Administrative Authorities is not supported by this version of ISO/IEC 9594. Such distribution is handled administratively by bilateral agreements.

**9.2.4** The specification of what is involved in the definition of DIT structure, object classes, attribute types and attribute syntaxes can be found in 9.3 through 9.6 respectively.

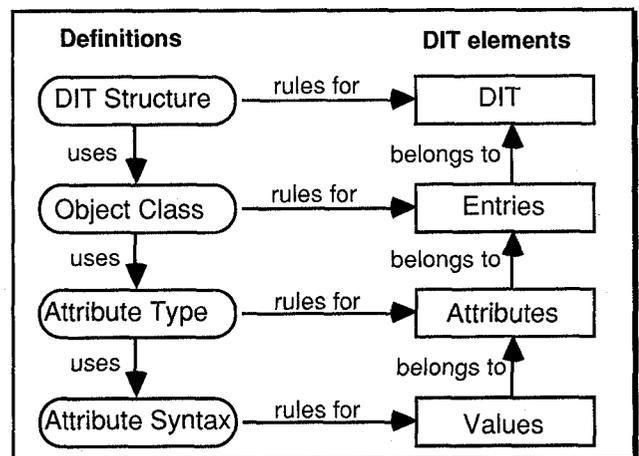


Figure 5 — Overview of Directory Schema

### 9.3 DIT Structure Definition

**9.3.1** A DIT Structure Rule defines the permitted hierarchical relationships between entries, and their

permitted RDNs. The definition of a DIT Structure Rule involves:

- identifying the subordinate and superior object classes;
- identifying the attribute types which may be involved in subordinate entries' RDNs; and
- (optionally) additional information.

**9.3.2** The Directory permits an entry to stand in the relationship of immediate subordinate to another (its immediate superior) only if there exists a DIT structure rule, contained in the schema (see 9.2.3) applicable to the portion of the DIB that would contain the entry, for which:

- the entry is of the subordinate object class;
- the immediate superior of the entry is of the superior object class;
- the attribute type(s) forming the entry's RDN is (are) among those permitted;
- any conditions imposed by the additional information are satisfied.

#### Notes

1. Techniques for documenting DIT Structure or for representing structure rules in the DIB are not presently provided by ISO/IEC 9594.
2. If a DIT Structure rule permits subordinates or superiors belonging to a particular class, it implicitly (unless explicitly overridden) also allows subordinates or superiors belonging to any object class derived from that class (see 9.4).

**9.3.3** The Directory enforces the defined structure rules at every entry in the DIT. Any attempt to modify the DIT in such a way as to violate the applicable structure rules fails.

**9.3.4** A DIT Structure Rule in which an object class is the subordinate is termed a name binding for that object class.

**9.3.5** For an object class to be represented by entries in a portion of the DIB, at least one name binding for that object class shall be contained in the applicable part of the schema. The schema contains additional name bindings as required.

## 9.4 Object Class Definition

**9.4.1** The definition of an object-class involves:

- a) optionally, assigning an object-identifier for the object-class;
- b) indicating which classes this is to be a subclass of;
- c) listing the *mandatory* attribute types that an entry of the object class shall contain in addition to the mandatory attribute types of all its superclasses;

- d) listing the *optional* attribute types that an entry of the object class may contain in addition to the optional attributes of all its superclasses.

**Note** -- An object class without an assigned object identifier is intended for local use as a means of conveniently adding new attribute types to a pre-defined superclass. This addition allows for a number of possibilities. For example, an Administrative authority may define an unregistered Object Class so as to permit a user to add any registered attribute to the entry. The Administrative authority may limit the attributes for an entry for a particular object class to those on a locally held list. It may also make particular attributes mandatory for a particular object class, over and above those required by the registered object class definition.

**9.4.2** There is one special object class, of which every other class is a subclass. This object class is called **Top** and is defined in 9.4.8.1.

**9.4.3** Every entry shall contain an attribute of type object class to identify the object class and superclasses to which the entry belongs. The definition of this attribute is given in 9.5.4. The attribute is multivalued. There shall be one value of the attribute for the object class and each of its superclasses for which an object identifier is defined, except that the value of **Top** need not be present so long as some other value is present.

#### Notes

1. The requirement that the **ObjectClass** attribute be present in every entry is reflected in the definition of **Top**.
2. Because an object class is considered to belong to all its superclasses, each member of the chain of superclasses up to Top is represented by a value in the object class attribute (and any value in the chain may be matched by a filter).

The **ObjectClass** attribute is managed by the Directory, i.e., it may not be modified by the user.

**9.4.4** The Directory enforces the defined object class for every entry in the DIB. Any attempt to modify an entry that would violate the entry's object class definition fails.

**Note** — In particular, the Directory will prevent:

- a) attribute types absent from the object class definition being added to an entry of that object class;
- b) an entry being created with one or more absent mandatory attribute types for the object class of the entry;
- c) a mandatory attribute type for the object class of the entry being deleted.

**9.4.5** The special object class **Alias** is defined in 9.4.8.2. Every alias entry shall have an object class which is a subclass of this class.

**Note**- The Directory's dereferencing of alias entries ensures that the values of the **ObjectClass** attribute of an alias entry are rarely seen. It is recommended that appropriate alias object classes be derived from **Alias** without assigning an object identifier.

**9.4.6** The following ASN.1 macro may (but need not) be used to define an object class. The empty production for **SubclassOf** is permitted only in defining **Top**:

```

OBJECT-CLASS MACRO ::=
BEGIN

TYPENOTATION ::= SubclassOf
                    MandatoryAttributes
                    OptionalAttributes

VALUENOTATION ::=
    value (VALUE OBJECT IDENTIFIER)

SubclassOf ::=
    "SUBCLASS OF" Subclasses1
    empty

Subclasses ::= Subclass1
                Subclass ", " Subclasses

Subclass ::= value (OBJECT-CLASS)

MandatoryAttributes ::=
    "MUST CONTAIN {"Attributes"}" | empty

OptionalAttributes ::=
    "MAY CONTAIN {"Attributes"}" | empty

Attributes ::= AttributeTerm1
                AttributeTerm ", " Attributes

AttributeTerm ::= Attribute | AttributeSet

Attribute ::= value (ATTRIBUTE)

AttributeSet ::= value (ATTRIBUTE-SET)

END
    
```

The correspondence between the parts of the definition as listed in 9.4.1, and the various pieces of the notation introduced by the macro, is as follows:

- the object identifier to the object class is the value supplied in the value assignment of the macro;
- the superclasses of which this object class is a subclass is that identified by the **SubclassOf** production, i.e., that following "SUBCLASS OF";
- the mandatory attributes are those identified by the list of object identifiers produced by the **MandatoryAttributes** production, i.e., those following **MUST CONTAIN**;
- the optional attributes are those identified by the list of object identifiers produced by the **OptionalAttributes** production, i.e., those following **MAY CONTAIN**.

**Notes**

1 The object identifiers in (c) and (d) identify both individual attributes and sets of attributes (see 9.4.7). The effective list in both cases is the set union of these. If an attribute appears in both the mandatory set and the optional set, it shall be considered mandatory.

2 The macro is used in defining selected object classes in ISO/IEC 9594-7. Should all the pieces of notation introduced by the macro and described in b), c) and d) above be empty, the resulting

notation ("OBJECT-CLASS") can be used to denote any possible object class.

**9.4.7** An *attribute set* is a set of attributes identified by an object identifier. The definition of an attribute set involves:

- assigning an object identifier to the set;
- listing the object identifiers of the attributes and other attribute sets whose members together form the set.

The following ASN.1 macro may (but need not) be used to define a set of attributes for use with the **OBJECT-CLASS** macro:

```

ATTRIBUTE-SET MACRO ::=
BEGIN

TYPE NOTATION ::= "CONTAINS" "{"Attributes"}" |
    empty

VALUE NOTATION ::= value (VALUE OBJECT
    IDENTIFIER)

Attributes ::= AttributeTerm1
                AttributeTerm ", " Attributes

AttributeTerm ::= Attribute | AttributeSet

Attribute ::= value (ATTRIBUTE)

AttributeSet ::= value (ATTRIBUTE-SET)

END
    
```

The correspondence between the parts of the definition of an attribute set and the notation introduced by the macro is as follows:

- the object identifier assigned to the attribute set is the value supplied in the value assignment of the macro
- the set of attributes comprising the attribute set is that formed by the set union of the attributes and sets of attributes identified by the **Attributes** production, i.e., following "CONTAINS".

Should the "empty" alternative of the notation be selected, the resulting notation ("ATTRIBUTE-SET") can be used to denote any possible attribute set.

**9.4.8** The object classes previously mentioned are defined in 9.4.8.1 and 9.4.8.2.

**Note** — These are partial definitions: the object identifiers are actually allocated for these object classes in ISO/IEC 9594-7 so as to provide a single point of allocation of these object identifiers in ISO/IEC 9594;

**9.4.8.1** The object class **Top** is defined as follows:

```

Top ::=
OBJECT-CLASS
MUST CONTAIN {objectClass}
    
```

**9.4.8.2** The object class **Alias** is defined as follows:

**Alias ::=**  
**OBJECT-CLASS**  
**SUBCLASS OF top**  
**MUST CONTAIN {aliasedObjectName}**

#### Notes

1. The object class **Alias** does not specify appropriate attribute types for the RDN of an alias entry. Administrative Authorities may specify subclasses of the class **Alias** which specify useful attribute types for RDNs of alias entries (see ISO/IEC 9594-7).
2. Entries of a subclass of the class **Alias** are alias entries.

## 9.5 Attribute Type Definition

**9.5.1** The definition of an attribute type involves:

- a) assigning an object identifier to the attribute type;
- b) indicating or defining the attribute syntax for the attribute type;
- c) indicating whether an attribute of this type may have only one or may have more than one value (recur).

**9.5.2** The Directory ensures that the indicated attribute syntax is used for every attribute of this type. The Directory also ensures that attributes of this type will have one and only one value in entries if attributes of this type are defined to have only one value.

**9.5.3** The following ASN.1 macro may (but need not) be used to define an attribute type:

**ATTRIBUTE MACRO ::=**  
**BEGIN**

**TYPENOTATION ::= AttributeSyntax Multivalued|empty**  
**VALUENOTATION ::= value (VALUE OBJECT IDENTIFIER)**

**AttributeSyntax ::=**  
**"WITH ATTRIBUTE-SYNTAX" SyntaxChoice**

**Multivalued ::= "SINGLE VALUE"**  
**"MULTI VALUE"|empty**

**SyntaxChoice ::= value(ATTRIBUTE-SYNTAX)**  
**Constraint|type MatchTypes**

**Constraint ::= ("ConstraintAlternative")|empty**

**ConstraintAlternative ::= StringConstraint|IntegerConstraint**

**StringConstraint ::= "SIZE" (" SizeConstraint ")**

**SizeConstraint ::= SingleValue|Range**

**SingleValue ::= value (INTEGER)**

**Range ::= value (INTEGER) "." value**  
**(INTEGER)**

**IntegerConstraint ::= Range**

**MatchTypes ::= "MATCHES FOR" Matches|empty**

**Matches ::= Match Matches|Match**

**Match ::= "EQUALITY"| "SUBSTRINGS"**  
**"ORDERING"**

**END**

The correspondence between the parts of the definition, as listed in 9.5.1, and the various pieces of the notation introduced by the macro, is as follows:

- (a) the object identifier assigned to the attribute type is the value supplied in the value assignment of the MACRO;
- (b) the attribute syntax for the attribute type is that identified by the **AttributeSyntax** production. This either points to a separately defined attribute syntax, or explicitly defines an attribute syntax by giving its ASN.1 type and matching rules (see 9.6). If a separately identified attribute syntax is employed, a size constraint for underlying string types or a value range for an underlying integer type may optionally be indicated;
- (c) the attribute is single valued if the **MultiValued** production is "SINGLE VALUE", and may have one or more values if it is "MULTI VALUE" or empty.

**Note** — The macro is used in defining selected attribute types in ISO/IEC 9594-6.

Should the "empty" alternative of the type notation be selected, the resulting notation ("ATTRIBUTE") can be used to denote any possible attribute type.

**9.5.4** The attribute types identified in 7.3.3 which are known to and used by the Directory for its own purposes are defined as follows:

**ObjectClass ::= ATTRIBUTE**  
**WITH ATTRIBUTE-SYNTAX objectIdentifierSyntax**

**AliasedObjectName ::= ATTRIBUTE**  
**WITH ATTRIBUTE-SYNTAX distinguishedNameSyntax**  
**SINGLE VALUE**

#### Notes

1. These are partial definitions: the object identifiers are actually allocated for these attribute types in ISO/IEC 9594-6 so as to provide a single point of allocation of these object identifiers in ISO/IEC 9594;
2. The attribute syntaxes referred to in these definitions are themselves defined in 9.6.5.

## 9.6 Attribute Syntax Definition

**9.6.1** The definition of an attribute syntax involves:

- a) optionally, assigning an object identifier to the attribute syntax;
- b) indicating the data type, in ASN.1, of the attribute syntax;
- c) defining appropriate rules for matching a presented value with a target attribute value held in the DIB.

None, some, or all of the following matching rules may be defined for a particular attribute syntax:

- 1) equality. Applicable to any attribute syntax. The presented value shall conform to the data type of the attribute syntax.
- 2) substrings. Applicable to any attribute syntax with a string data type. The presented value shall be a sequence ('SEQUENCE OF'), each of whose elements conforms to the data type
- 3) ordering. Applicable to any attribute syntax for which a rule can be defined that will allow a presented value to be described as less than, equal to, or greater than a target value. The presented value shall conform to the data type of the attribute syntax.

**9.6.2** If no equality matching rule is defined the Directory

- a) treats values as attributes of this attribute syntax as having type ANY, i.e., the Directory does not check that those values conform with the data type indicated for the attribute syntax.
- b) will not attempt to match presented values against target values of such an attribute type.

**Note** - It follows that the Directory will not permit such an attribute to be used in a distinguished name, nor allow for a specific value to be modified.

**9.6.3** If an equality matching rule is defined, the Directory

- a) treats values of attributes of this attribute syntax as having type ANY DEFINED BY the data type indicated for the attribute syntax.
- b) will only match according to the matching rules defined for that attribute syntax.
- c) will only match a presented value of a suitable data type as specified in 9.6.1 (c).

**9.6.4** The following ASN.1 macro may, but need not, be used to define attribute syntaxes:

```

ATTRIBUTE-SYNTAX MACRO ::=
BEGIN

TYPE NOTATION ::= Syntax MatchTypesempty
VALUE NOTATION ::= value (VALUE OBJECT IDENTIFIER)

Syntax ::= type
MatchTypes ::= "MATCHES FOR" Matchesempty
Matches ::= Match Matches!Match
Match ::= "EQUALITY"|"SUBSTRINGS"|"ORDERING"
END
    
```

The correspondence between the parts of the definition, as listed in 9.6.1, and the various pieces of the notation introduced by the macro, is as follows:

- a) the object identifier assigned to the attribute syntax is a value supplied in the value assignment of the macro.
- b) the data type of the attribute syntax is that identified by the Syntax production, i.e., that following macro name.
- c) the defined matching rules are equality, if "EQUALITY" appears in the **MatchTypes** production, substrings if "SUBSTRINGS" appears, and ordering if "ORDERING" appears. If the production is empty, then no matching rules are defined.

Should the "empty" alternative of the notation be selected, the resulting notation ("ATTRIBUTE-SYNTAX") can be used to denote any possible attribute syntax.

**Notes**

1 No support is provided in the macro for actually defining the matching rules themselves: this shall be done by natural language or by other means.

2 The macro is used in defining selected attribute syntaxes in ISO/IEC 9594-6.

**9.6.5** The attribute syntaxes used in 9.5.4 are defined in 9.6.5.1 and 9.6.5.2.

**Note** — These are partial definitions: the object identifiers are actually allocated for these attribute syntaxes in ISO/IEC 9594-6 so as to provide a single point of allocation of these object identifiers in ISO/IEC 9594;

**9.6.5.1** **ObjectIdentifierSyntax** is defined as follows:

```

ObjectIdentifierSyntax ::=
ATTRIBUTE-SYNTAX
OBJECT IDENTIFIER
MATCHES FOR EQUALITY
    
```

The matching rule for equality is inherent in the definition of the ASN.1 type object identifier.

**9.6.5.2** **DistinguishedNameSyntax** is defined as follows:

```

DistinguishedNameSyntax ::=
ATTRIBUTE-SYNTAX
DistinguishedName
MATCHES FOR EQUALITY
    
```

A presented distinguished name value is equal to a target distinguished name value if and only if all of the following are true:

- a) the number of RDNs in each is the same;
- b) corresponding RDNs have the same number of AVAs;

- c) corresponding AVAs (i.e., those with identical attribute types) have attribute values which match for equality (in such a match, the attribute values take the same roles — i.e., as presented or target

value — as the distinguished name which contains them does in the overall match).

## SECTION 4: SECURITY MODEL

### 10 Security

**10.1** The directory exists in an environment where various authorities provide access to their fragment of the DIB. Such access shall be in conformance to the security policy (see ISO/IEC 9594-8) of the security domain in which the fragment of the DIB exists.

**10.2** Two specific components of a security policy are addressed here:

- a) the definition of an authorization policy;
- b) the definition of an authentication policy.

**10.3** The definition of authorization in the context of the Directory includes the methods to:

- a) specify access rights;
- b) enforce access rights (access control);
- c) maintain access rights.

**10.4** The definition of authentication in the context of the Directory includes of the methods to verify:

- a) the identity of DSAs and directory users;
- b) the identity of the origin of received information at an access point.

The integrity of received information is a local matter and shall be in conformance to the security policy in force.

**10.5** This part of ISO/IEC 9594 does not define a Security Policy.

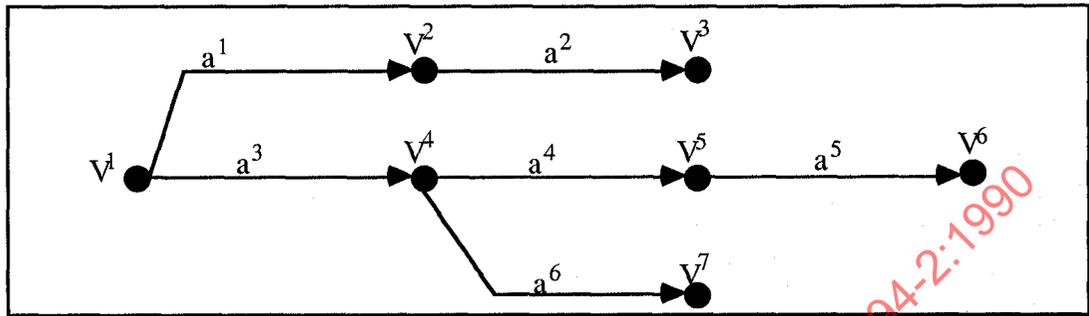
**10.6** Annex F describes guidelines for specifying access rights.

**10.7** ISO/IEC 9594-8 defines authentication procedures. The DAP and DSP may provide strong authentication of the initiator by the signing of the request, data integrity of the request by signing of the request, strong authentication of the responder, and data integrity of the result by signing the result. The DAP may provide simple authentication between a DUA and a DSA. The DSP may provide simple authentication between two DSAs.

**10.8** Administrative authorities of applications which make use the Directory can use their own security policy. The directory can support applications by holding authentication information (e.g., distinguished names, passwords, certificates) about communication entities. This is further described in ISO/IEC 9594-8.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9594-2:1990

## Annex A (Informative) The Mathematics of Trees



A tree is a set of points, called *vertices*, and a set of directed lines, called *arcs*; each arc  $a$  leads from a vertex  $V$  to a vertex  $V'$ . For example, the tree in the figure has seven vertices; labelled  $V^1$  through  $V^7$ , and six arcs, labelled  $a^1$  through  $a^6$ .

Two vertices  $V$  and  $V'$  are said to be the *initial* and *final* vertices, respectively, of an arc  $a$  from  $V$  to  $V'$ . For example,  $V^2$  and  $V^3$  are the initial and final vertices, respectively, of arc  $a^2$ . Several different arcs may have the same initial vertex, but not the same final vertex. For example, arcs  $a^1$  and  $a^3$  have the same initial vertex,  $V^1$ , but no two arcs in the figure have the same final vertex.

The vertex that is not the final vertex of any arc is often referred to as the *root* vertex, or even more informally as

the "root" of the tree. For example, in the figure,  $V^1$  is the root.

A vertex that is not the initial vertex of any arc is often referred to informally as a *leaf* vertex, or even more informally, as a "leaf" of the tree graph. For example, vertices  $V^3$ ,  $V^6$ , and  $V^7$  are leaves.

An *oriented path* from a vertex  $V$  to a vertex  $V'$  is a set of arcs  $(a^1, a^2, \dots, a^n)$  ( $n \geq 1$ ) such that  $V$  is the initial vertex of arc  $a^1$ ,  $V'$  is the final vertex of arc  $a^n$ , and the final vertex of arc  $a^k$  is also the initial vertex of arc  $a^{k+1}$  for  $1 \leq k < n$ . For example, the oriented path from vertex  $V^1$  to vertex  $V^6$  is the set of arcs  $(a^3, a^4, a^5)$ . The term "path" should be understood to denote an oriented path from the root to a vertex.

## Annex B ( Normative ) Object Identifier Usage

This Annex documents the upper reaches of the object identifier subtree in which all of the object identifiers assigned in ISO/IEC 9594 reside. It does so by providing

an ASN.1 module called "UsefulDefinitions" in which all non-leaf nodes in the subtree are assigned names.

---

```

UsefulDefinitions {joint-iso-ccitt ds(5) modules(1) usefulDefinitions(0)}
DEFINITIONS ::=
BEGIN

EXPORTS
  module, serviceElement, applicationContext, attributeType, attributeSyntax, objectClass, algorithm,
  abstractSyntax, attributeSet, usefulDefinitions, informationFramework, directoryAbstractService,
  directoryObjectIdentifiers, algorithmObjectIdentifiers, distributedOperations, protocolObjectIdentifiers,
  selectedAttributeTypes, selectedObjectClasses, authenticationFramework, upperBounds, dap, dsp,
  distributedDirectoryObjectIdentifiers, id-ac, id-ase, id-as, id-ot, id-pt;

      ds OBJECT IDENTIFIER ::= {joint-iso-ccitt ds(5)}

-- categories of information object --

      module OBJECT IDENTIFIER ::= {ds 1}
      serviceElement OBJECT IDENTIFIER ::= {ds 2}
      applicationContext OBJECT IDENTIFIER ::= {ds 3}
      attributeType OBJECT IDENTIFIER ::= {ds 4}
      attributeSyntax OBJECT IDENTIFIER ::= {ds 5}
      objectClass OBJECT IDENTIFIER ::= {ds 6}
      attributeSet OBJECT IDENTIFIER ::= {ds 7}
      algorithm OBJECT IDENTIFIER ::= {ds 8}
      abstractSyntax OBJECT IDENTIFIER ::= {ds 9}

-- modules --

      usefulDefinitions OBJECT IDENTIFIER ::= {module 0}
      informationFramework OBJECT IDENTIFIER ::= {module 1}
      directoryAbstractService OBJECT IDENTIFIER ::= {module 2}
      distributedOperations OBJECT IDENTIFIER ::= {module 3}
      protocolObjectIdentifiers OBJECT IDENTIFIER ::= {module 4}
      selectedAttributeTypes OBJECT IDENTIFIER ::= {module 5}
      selectedObjectClasses OBJECT IDENTIFIER ::= {module 6}
      authenticationFramework OBJECT IDENTIFIER ::= {module 7}
      algorithmObjectIdentifiers OBJECT IDENTIFIER ::= {module 8}
      directoryObjectIdentifiers OBJECT IDENTIFIER ::= {module 9}
      upperBounds OBJECT IDENTIFIER ::= {module 10}
      dap OBJECT IDENTIFIER ::= {module 11}
      dsp OBJECT IDENTIFIER ::= {module 12}
      distributedDirectoryObjectIdentifiers OBJECT IDENTIFIER ::= {module 13}

-- synonyms --

      id-ac OBJECT IDENTIFIER ::= applicationContext
      id-ase OBJECT IDENTIFIER ::= serviceElement
      id-as OBJECT IDENTIFIER ::= abstractSyntax
      id-ot OBJECT IDENTIFIER ::= object
      id-pt OBJECT IDENTIFIER ::= port

```

END

---

## Annex C ( Normative ) Information Framework in ASN.1

This Annex provides a summary of all of the ASN.1 type, value and macro definitions contained in this part of

ISO/IEC 9594. The definitions form the ASN.1 module **InformationFramework**.

---

```
InformationFramework {joint-iso-ccitt ds(5) modules(1) informationFramework(1)}
```

```
DEFINITIONS ::=
```

```
BEGIN
```

```
EXPORTS
```

```
Attribute, AttributeType, AttributeValue, AttributeValueAssertion,  
DistinguishedName, Name, RelativeDistinguishedName,  
OBJECT-CLASS, ATTRIBUTE, ATTRIBUTE-SET, ATTRIBUTE-SYNTAX,  
Top, Alias,  
ObjectClass, AliasedObjectName,  
ObjectIdentifierSyntax, DistinguishedNameSyntax;
```

```
IMPORTS
```

```
selectedAttributeTypes, selectedObjectClasses  
FROM UsefulDefinitions {joint-iso-ccitt ds(5) modules(1) usefulDefinitions(0)}  
top  
FROM SelectedObjectClasses selectedObjectClasses  
objectIdentifierSyntax, distinguishedNameSyntax, objectClass, aliasedObjectName  
FROM SelectedAttributeTypes selectedAttributesTypes;
```

```
-- attribute data types --
```

```
Attribute ::= SEQUENCE { type AttributeType,  
values SET OF AttributeValue -- at least one value is required - }
```

```
AttributeType ::= OBJECT IDENTIFIER
```

```
AttributeValue ::= ANY
```

```
AttributeValueAssertion ::= SEQUENCE {AttributeType, AttributeValue}
```

```
-- naming data types --
```

```
Name ::= CHOICE { RDNSequence -- only one possibility for now - }
```

```
RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
```

```
DistinguishedName ::= RDNSequence
```

```
RelativeDistinguishedName ::= SET OF AttributeValueAssertion
```

-- macros --

**OBJECT-CLASS MACRO ::=**  
**BEGIN**

<b>TYPENOTATION</b>	<b>::=</b>	<b>SubclassOf MandatoryAttributes OptionalAttributes</b>
<b>VALUENOTATION</b>	<b>::=</b>	<b>value (VALUE OBJECT IDENTIFIER)</b>
<b>SubclassOf</b>	<b>::=</b>	<b>"SUBCLASS OF" Subclassesempty</b>
<b>Subclasses</b>	<b>::=</b>	<b>Subclassl</b>
		<b>Subclass "," Subclasses</b>
<b>Subclass</b>	<b>::=</b>	<b>value(OBJECT-CLASS)</b>
<b>MandatoryAttributes</b>	<b>::=</b>	<b>"MUST CONTAIN {"Attributes"}"empty</b>
<b>OptionalAttributes</b>	<b>::=</b>	<b>"MAY CONTAIN {"Attributes"}"empty</b>
<b>Attributes</b>	<b>::=</b>	<b>AttributeTerm/AttributeTerm "," Attributes</b>
<b>AttributeTerm</b>	<b>::=</b>	<b>Attribute/AttributeSet</b>
<b>Attribute</b>	<b>::=</b>	<b>value (ATTRIBUTE)</b>
<b>AttributeSet</b>	<b>::=</b>	<b>value (ATTRIBUTE-SET)</b>

**END**

**ATTRIBUTE-SET MACRO ::=**

**BEGIN**

<b>TYPE NOTATION</b>	<b>::=</b>	<b>"CONTAINS"{"Attributes "}"empty</b>
<b>VALUE NOTATION</b>	<b>::=</b>	<b>value(VALUE OBJECT IDENTIFIER)</b>
<b>Attributes</b>	<b>::=</b>	<b>AttributeTerm/AttributeTerm "," Attributes</b>
<b>AttributeTerm</b>	<b>::=</b>	<b>Attribute/AttributeSet</b>
<b>Attribute</b>	<b>::=</b>	<b>value(ATTRIBUTE)</b>
<b>AttributeSet</b>	<b>::=</b>	<b>value(ATTRIBUTE-SET)</b>

**END**

**ATTRIBUTE MACRO ::=**

**BEGIN**

<b>TYPENOTATION</b>	<b>::=</b>	<b>AttributeSyntax Multivaluedempty</b>
<b>VALUENOTATION</b>	<b>::=</b>	<b>value (VALUE OBJECT IDENTIFIER)</b>
<b>AttributeSyntax</b>	<b>::=</b>	<b>"WITH ATTRIBUTE -SYNTAX" SyntaxChoice</b>
<b>Multivalued</b>	<b>::=</b>	<b>"SINGLE VALUE" "MULTI VALUE"empty</b>
<b>SyntaxChoice</b>	<b>::=</b>	<b>value(ATTRIBUTE-SYNTAX) ConstraintType MatchTypes</b>
<b>Constraint</b>	<b>::=</b>	<b>"("ConstraintAlternative")"empty</b>
<b>ConstraintAlternative</b>	<b>::=</b>	<b>StringConstraint IntegerConstraint</b>
<b>StringConstraint</b>	<b>::=</b>	<b>"SIZE" "(" SizeConstraint ")"</b>
<b>SizeConstraint</b>	<b>::=</b>	<b>SingleValue Range</b>
<b>SingleValue</b>	<b>::=</b>	<b>value (INTEGER)</b>
<b>Range</b>	<b>::=</b>	<b>value (INTEGER) ".." value (INTEGER)</b>
<b>IntegerConstraint</b>	<b>::=</b>	<b>Range</b>
<b>MatchTypes</b>	<b>::=</b>	<b>"MATCHES FOR" Matchesempty</b>
<b>Matches</b>	<b>::=</b>	<b>Match Matches Match</b>
<b>Match</b>	<b>::=</b>	<b>"EQUALITY" "SUBSTRINGS" "ORDERING"</b>

**END**

**ATTRIBUTE-SYNTAX MACRO ::=**  
**BEGIN**

**TYPENOTATION ::= Syntax MatchTypesempty**  
**VALUENOTATION ::= value (VALUE OBJECT IDENTIFIER)**  
**Syntax ::= type**  
**MatchTypes ::= "MATCHES FOR" Matchesempty**  
**Matches ::= Match Matches!Match**  
**Match ::= EQUALITY|SUBSTRINGS|ORDERING**

**END**

-- *object classes* --

**Top ::= OBJECT-CLASS**  
**MUST CONTAIN {objectClass }**  
**Alias ::= OBJECT-CLASS**  
**SUBCLASS OF top**  
**MUST CONTAIN {aliasedObjectName }**

-- *attribute types* --

**ObjectClass ::= ATTRIBUTE**  
**WITH ATTRIBUTE-SYNTAX objectIdentifierSyntax**  
**AliasedObjectName ::= ATTRIBUTE**  
**WITH ATTRIBUTE-SYNTAX distinguishedNameSyntax**  
**SINGLE VALUE**

-- *attribute syntaxes* --

**ObjectIdentifierSyntax ::=**  
**ATTRIBUTE-SYNTAX**  
**OBJECT IDENTIFIER**  
**MATCHES FOR EQUALITY**  
**DistinguishedNameSyntax ::=**  
**ATTRIBUTE-SYNTAX**  
**DistinguishedName**  
**MATCHES FOR EQUALITY**

**END**

IECNORM.COM : Click to view the full PDF of ISO/IEC 9594-2:1990