INTERNATIONAL STANDARD

ISO/IEC 9594-2

Third edition
1998-12-15

Information technology — Open Systems Interconnection — The Directory: Models

Technologies de l'information — Interconnexion de systèmes ouverts (OSI) — L'Annuaire: Les modèles

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Reference number
ISO/IEC 9594-2:1998(E)

© ISO/IEC 1998

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

© ISO/IEC 1998

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 either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.ch
Web www.iso.ch

Published by ISO in 2000

Printed in Switzerland

 © ISO/IEC 1998 – All rights reserved

Contents

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

© ISO/IEC 1998 – All rights reserved

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

© ISO/IEC 1998 – All rights reserved

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

© ISO/IEC 1998 – All rights reserved

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

© ISO/IEC 1998 – All rights reserved

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

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.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 9594 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 9594-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 6, *Telecommunications and information exchange between systems*, in collaboration with ITU-T. The identical text is published as ITU-T Recommendation X.501.

This third edition cancels and replaces the second edition (ISO/IEC 9594-2:1995), of which it constitutes a minor revision.

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*

— *Part 9: Replication*

— *Part 10: Use of systems management for administration of the Directory*

Annexes A to F and P form a normative part of this part of ISO/IEC 9594. Annexes G to O and Q are for information only.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

© ISO/IEC 1998 – All rights reserved

Introduction

This Recommendation | International Standard, together with the other Recommendations | International Standards, 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.

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.

This Recommendation | International Standard provides a number of different models for the Directory as a framework for the other Recommendations in the ITU-T X.500 series | parts of ISO/IEC 9594. The models are the overall (functional) model; the administrative authority model, generic Directory Information Models providing Directory User and Administrative User views on Directory information, generic DSA and DSA information models, an Operational Framework and a security model.

The generic Directory Information Models describe, for example, how information about objects is grouped to form Directory entries for those objects and how that information provides names for objects.

The generic DSA and DSA information models and the Operational Framework provide support for Directory distribution.

This Recommendation | International Standard provides a specialization of the generic Directory Information Models to support Directory Schema administration.

This third edition technically revises and enhances, but does not replace, the second edition of this Recommendation | International Standard. Implementations may still claim conformance to the second edition. However, at some point, the second edition will not be supported (i.e. reported defects will no longer be resolved). It is recommended that implementations conform to this third edition as soon as possible.

This third edition specifies version 1 and version 2 of the Directory protocols.

The first and second editions also specified version 1. Most of the services and protocols specified in this edition are designed to function under version 1. When version 1 has been negotiated, differences between the services and between the protocols defined in the three editions are accommodated using the rules of extensibility defined in this edition of ITU-T Rec. X.519 (1997) | ISO/IEC 9594-5:1998. However some enhanced services and protocols, e.g. signed errors, will not function unless all Directory entities involved in the operation have negotiated version 2.

Implementors should note that a defect resolution process exists and that corrections may be applied to this part of this International Standard in the form of technical corrigenda. The identical corrections will be applied to this Recommendation in the form of Corrigenda and/or an Implementor's Guide. A list of approved technical corrigenda for this part of this International Standard can be obtained from the subcommittee secretariat. Published technical corrigenda are available from your national standards organization. The ITU-T Corrigenda and Implementor's Guides may be obtained from the ITU Web site.

Annex A, which is an integral part of this Recommendation | International Standard, summarizes the usage of ASN.1 object identifiers in the ITU-T X.500-series Recommendations | parts of ISO/IEC 9594.

Annex B, which is an integral part of this Recommendation | International Standard, provides the ASN.1 module which contains all of the definitions associated with the information framework.

Annex C, which is an integral part of this Recommendation | International Standard, provides the subschema administration schema in ASN.1.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

© ISO/IEC 1998 – All rights reserved

Annex D, which is an integral part of this Recommendation | International Standard, provides the ASN.1 module for Basic Access Control.

Annex E, which is an integral part of this Recommendation | International Standard, provides the ASN.1 module which contains all the definitions associated with DSA operational attribute types.

Annex F, which is an integral part of this Recommendation | International Standard, provides the ASN.1 module which contains all the definitions associated with operational binding management operations.

Annex G, which is not an integral part of this Recommendation | International Standard, summarizes the mathematical terminology associated with tree structures.

Annex H, which is not an integral part of this Recommendation | International Standard, describes some criteria that can be considered in designing names.

Annex I, which is not an integral part of this Recommendation | International Standard, provides some examples of various aspects of Schema.

Annex J, which is not an integral part of this Recommendation | International Standard, provides an overview of the semantics associated with Basic Access Control permissions.

Annex K, which is not an integral part of this Recommendation | International Standard, provides an extended example of the use of Basic Access Control.

Annex L, which is not an integral part of this Recommendation | International Standard, describes some DSA-specific entry combinations.

Annex M, which is not an integral part of this Recommendation | International Standard, provides a framework for the modelling of knowledge.

Annex N, which is not an integral part of this Recommendation | International Standard, lists alphabetically the terms defined in this Recommendation | International Standard.

Annex O, which is not an integral part of this Recommendation | International Standard, describes criteria on whether a name can be an alternative distinguished name or the primary distinguished name, whether it can contain alternative values, and whether it can include context information.

Annex P, which is an integral part of this Recommendation | International Standard, provides the ASN.1 module which contains all the definitions associated with enhanced security.

Annex Q, which is not an integral part of this Recommendation | International Standard, lists the amendments and defect reports that have been incorporated to form this edition of this Recommendation | International Standard.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

© ISO/IEC 1998 – All rights reserved

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – OPEN SYSTEMS INTERCONNECTION – THE DIRECTORY: MODELS

SECTION 1 – GENERAL

1 Scope

The models defined in this Recommendation | International Standard provide a conceptual and terminological framework for the other ITU-T X.500-series Recommendations | parts of ISO/IEC 9594 which define various aspects of the Directory.

The functional and administrative authority models define ways in which the Directory can be distributed, both functionally and administratively. Generic DSA and DSA information models and an Operational Framework are also provided to support Directory distribution.

The generic Directory Information Models describe the logical structure of the DIB from the perspective of Directory and Administrative Users. In these models, the fact that the Directory is distributed, rather than centralized, is not visible.

This Recommendation | International Standard provides a specialization of the generic Directory Information Models to support Directory Schema administration.

The other ITU-T Recommendations in the X.500 series | parts of ISO/IEC 9594 make use of the concepts defined in this Recommendation | International Standard to define specializations of the generic information and DSA models to provide specific information, DSA and operational models supporting particular directory capabilities (e.g. Replication):

a) the service provided by the Directory is described (in ITU-T Rec. X.511 | 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 ITU-T Rec. X.518 | 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;

c) replication capabilities offered by the component parts of the Directory to improve overall Directory performance are specified (in ITU-T Rec. X.525 | ISO/IEC 9594-9).

The security model establishes a framework for the specification of access control mechanisms. It provides a mechanism for identifying the access control scheme in effect in a particular portion of the DIT, and it defines three flexible, specific access control schemes which are suitable for a wide variety of applications and styles of use. The security model also provides a framework for protecting the confidentiality and integrity of directory operations using mechanisms such as encryption and digital signatures. This makes use of the framework for authentication defined in ITU-T Rec. X.509 | ISO/IEC 9594-8 as well as generic upper layers security tools defined in ITU-T Rec. X.830 | ISO/IEC 11586-1.

DSA models establish a framework for the specification of the operation of the components of the Directory. Specifically:

a) the Directory functional model describes how the Directory is manifested as a set of one or more components, each being a DSA;

b) the Directory distribution model describes the principals according to which the DIB entries and entry-copies may be distributed among DSAs;

c) the DSA information model describes the structure of the Directory user and operational information held in a DSA;

d) the DSA operational framework describes the means by which the definition of specific forms of cooperation between DSAs to achieve particular objectives (e.g. shadowing) is structured.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

2.1 Identical Recommendations | International Standards

- ITU-T Recommendation X.200 (1994) | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model.*
- ITU-T Recommendation X.500 (1997) | ISO/IEC 9594-1:1998, *Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services*.
- ITU-T Recommendation X.509 (1997) | ISO/IEC 9594-8:1998, *Information technology – Open Systems Interconnection – The Directory: Authentication framework*.
- ITU-T Recommendation X.511 (1997) | ISO/IEC 9594-3:1998, *Information technology – Open Systems Interconnection – The Directory: Abstract service definition*.
- ITU-T Recommendation X.518 (1997) | ISO/IEC 9594-4:1998, *Information technology – Open Systems Interconnection – The Directory: Procedures for distributed operation*.
- ITU-T Recommendation X.519 (1997) | ISO/IEC 9594-5:1998, *Information technology – Open Systems Interconnection – The Directory: Protocol specifications*.
- ITU-T Recommendation X.520 (1997) | ISO/IEC 9594-6:1998, *Information technology – Open Systems Interconnection – The Directory: Selected attribute types*.
- ITU-T Recommendation X.521 (1997) | ISO/IEC 9594-7:1998, *Information technology – Open Systems Interconnection – The Directory: Selected object classes*.
- ITU-T Recommendation X.525 (1997) | ISO/IEC 9594-8:1999, *Information technology – Open Systems Interconnection – The Directory: Replication*.
- ITU-T Recommendation X.530 (1997) | ISO/IEC 9594-10:1998, *Information technology – Open Systems Interconnection – The Directory: Use of systems management for administration of the Directory*.
- CCITT Recommendation X.660 (1992) | ISO/IEC 9834-1:1993, *Information technology – Open Systems Interconnection – Procedures for the operation of OSI Registration Authorities: General procedures*.
- ITU-T Recommendation X.680 (1997) | ISO/IEC 8824-1:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*.
- ITU-T Recommendation X.681 (1997) | ISO/IEC 8824-2:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification*.
- ITU-T Recommendation X.682 (1997) | ISO/IEC 8824-3:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification*.
- ITU-T Recommendation X.683 (1997) | ISO/IEC 8824-4:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications*.
- ITU-T Recommendation X.803 (1994) | ISO/IEC 10745:1995, *Information technology – Open Systems Interconnection – Upper layers security model*.
- ITU-T Recommendation X.811 (1995) | ISO/IEC 10181-2:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Authentication framework*.
- ITU-T Recommendation X.812 (1995) | ISO/IEC 10181-3:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Access control framework.*
- ITU-T Recommendation X.813 (1996) | ISO/IEC 10181-4:1997, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Non-repudiation framework*.
- ITU-T Recommendation X.830 (1995) | ISO/IEC 11586-1:1996, *Information technology – Open Systems Interconnection – Generic upper layers security: Overview, models and notation.*
- ITU-T Recommendation X.833 (1995) | ISO/IEC 11586-4:1996, *Information technology – Open Systems Interconnection – Generic upper layers security: Protecting transfer syntax specification.*

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

2.2 Paired Recommendations | International Standards equivalent in technical content

– CCITT Recommendation X.800 (1991), *Security architecture for Open Systems Interconnection for CCITT applications*.

ISO 7498-2:1989, *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture*.

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply.

3.1 OSI Reference Model Definitions

The following terms are defined in ITU-T Rec. X.200 | ISO/IEC 7498-1:

a) *application-context;*

b) *application-entity;*

c) *application-process.*

3.2 Basic directory definitions

The following terms are defined in ITU-T Rec. X.500 | ISO/IEC 9594-1:

a) *Directory;*

b) *Directory Access Protocol;*

c) *Directory Information Base;*

d) *Directory Operational Binding Management Protocol;*

e) *Directory System Protocol;*

f) *(Directory) user.*

3.3 Distributed operation definitions

The following terms are defined in ITU-T Rec. X.518 | ISO/IEC 9594-4:

a) *access point;*

b) *hierarchical operational binding;*

c) *name resolution;*

d) *non-specific hierarchical operational binding;*

e) *relevant hierarchical operational binding*.

3.4 Replication definitions

The following terms are defined in ITU-T Rec. X.525 | ISO/IEC 9594-9:

a) *cache-copy;*

b) *consumer reference;*

c) *entry-copy;*

d) *master DSA;*

e) *primary shadowing;*

f) *replicated area;*

g) *replication;*

h) *secondary shadowing;*

i) *shadow consumer;*

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

j) *shadow supplier;*

k) *Shadowed DSA-Specific Entry;*

l) *shadowing;*

m) *supplier reference.*

The definitions of terms defined in this Recommendation | International Standard are included at the beginning of individual clauses, as appropriate. An index of these terms is provided in Annex N for easy reference.

4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply.

ACDF	Access Control Decision Function
ACI	Access Control Information
ACIA	Access Control Inner Area
ACSA	Access Control Specific Area
ADDMD	Administration Directory Management Domain
ASN.1	Abstract Syntax Notation One
AVA	attribute value assertion
BER	(ASN.1) Basic Encoding Rules
DACD	Directory Access Control Domain
DAP	Directory Access Protocol
DIB	Directory Information Base
DISP	Directory Information Shadowing Protocol
DIT	Directory Information Tree
DMD	Directory Management Domain
DMO	Domain Management Organization
DOP	Directory Operational Binding Management Protocol
DSA	Directory System Agent
DSE	DSA-Specific Entry
DSP	Directory System Protocol
DUA	Directory User Agent
HOB	Hierarchical Operational Binding
NHOB	Non-specific Hierarchical Operational Binding
NSSR	Non-Specific Subordinate Reference
PRDMD	Private Directory Management Domain
RHOB	Relevant Hierarchical Operational Binding (i.e. either a HOB or NHOB, as appropriate)
RDN	Relative Distinguished Name
SDSE	Shadowed DSE

5 Conventions

With minor exceptions, this Directory Specification has been prepared according to the "Presentation of ITU-T | ISO/IEC common text" guidelines in the Guide for ITU-T and ISO/IEC JTC 1 Cooperation, March 1997.

The term "Directory Specification" (as in "this Directory Specification") shall be taken to mean ITU-T Rec. X.501 | ISO/IEC 9594-2. The term "Directory Specifications" shall be taken to mean the X.500-series Recommendations | parts of ISO/IEC 9594.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

This Directory Specification uses the term "1988 edition systems" to refer to systems conforming to the first (1988) edition of the Directory Specifications, i.e. the 1988 edition of the series of CCITT X.500 Recommendations and the ISO/IEC 9594:1990 edition. This Directory Specification uses the term "1993 edition systems" to refer to systems conforming to the second (1993) edition of the Directory Specifications, i.e. the 1993 edition of the series of ITU-T X.500 Recommendations and the ISO/IEC 9594:1995 edition. Systems conforming to this third edition of the Directory Specifications are referred to as "1997 edition systems".

This Directory Specification presents ASN.1 notation in the bold Helvetica typeface. When ASN.1 types and values are referenced in normal text, they are differentiated from normal text by presenting them in the bold Helvetica typeface. The names of procedures, typically referenced when specifying the semantics of processing, are differentiated from normal text by displaying them in bold Times. Access control permissions are presented in italicized Times.

SECTION 2 – OVERVIEW OF THE DIRECTORY MODELS

6 Directory Models

6.1 Definitions

For the purposes of this Directory Specification, the following definitions apply:

6.1.1 administrative authority: An agent of the Domain Management Organization concerned with various aspects of Directory administration. The term *administrative authority* (in lower case) refers to the power vested in an Administrative Authority by the Domain Management Organization to execute policy.

6.1.2 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.

6.1.3 directory administrative and operational information: Information used by the Directory for administrative and operational purposes.

6.1.4 DIT domain: That part of the global DIT held by the DSAs forming a DMD.

6.1.5 directory management domain (DMD): A set of one or more DSAs and zero or more DUAs managed by a single organization.

6.1.6 domain management organization: An organization that manages a DMD (and the associated DIT Domain).

6.1.7 directory user information: Information of interest to users and their applications.

6.1.8 directory system agent (DSA): An OSI application process which is part of the Directory.

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

6.1.10 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.

6.1.11 private directory management domain (PRDMD): A DMD which is managed by an organization other than an Administration.

6.2 The Directory and its Users

The *Directory* is a repository of information. This repository is known as the Directory Information Base (DIB). Directory services provided to users are concerned with various kinds of access to this information.

The services provided by the Directory are defined in ITU-T Rec. X.511 | ISO/IEC 9594-3.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

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

A DUA is manifested as an application-process. In any instance of communication, each DUA represents precisely one directory user.

The Directory is manifested as a set of one or more application-processes known as *Directory System Agents (DSAs)*, each of which provides one or more of the access points. For a more detailed description of DSAs, see 19.2.

NOTE 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.

NOTE 2 – The DUA functions and a DSA 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.

NOTE 3 – A DUA may exhibit local behaviour and structure which is outside the scope of envisaged Directory Specifications. 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.

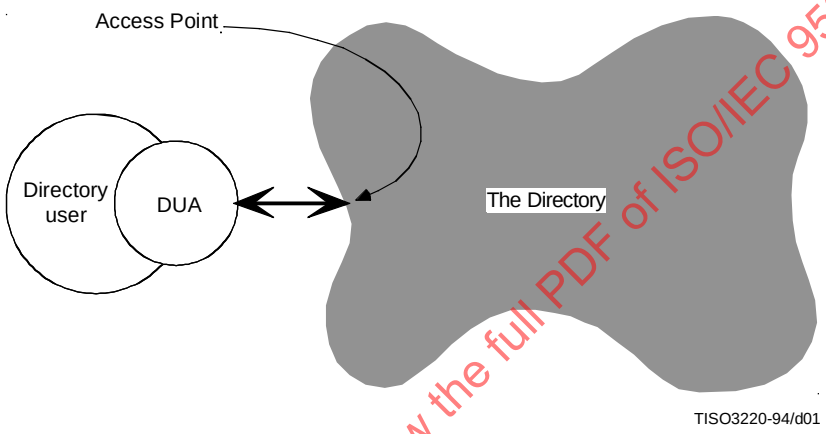


Figure 1 – Access to the Directory

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

6.3 Directory and DSA Information Models

6.3.1 Generic Models

Directory information may be classified as either:

- user information, placed in the Directory by, or on behalf of, users; and subsequently administered by, or on behalf of, users. Section 3 provides a model of this information; or
- administrative and operational information, held by the Directory to meet various administrative and operational requirements. Section 5 provides a model of this information. Also provided in Section 5 is a specification of the relationship between the user, administrative and operational information models.

These models, presenting views of the DIB from different perspectives, are referred to as the generic Directory Information Models.

Directory information models describe how the Directory as a whole represents information. The composition of the Directory as a set of potentially cooperating DSAs is abstracted from the model. The DSA information model, on the other hand, is especially concerned with DSAs and the information that must be held by DSAs in order that the set of DSAs comprising the Directory may together realize the Directory information model. The DSA Information Model is provided in clauses 20 through 21.

The DSA information model is a generic model describing the information held by DSAs and the relationship between this information and the DIB and DIT.

Some, but not all, of the information represented by the DSA information model is accessible via the Directory abstract service. Therefore, administration of all of the information described in these Directory Specifications is not possible via the Directory abstract service. It is envisioned that administration of DSA information will initially be a local matter, but that eventually some generic system management service will be employed to provide access to all of the information described in the DSA information model.

6.3.2 Specific Information Models

Subsequent to the development of generic models for the Directory as a whole and for its components, specific information models are required for the standardisation of particular aspects of the operation of the Directory and its components.

The generic Directory Information Models establish a framework for the following specific information models:

- an access control information model;
- a subschema information model;
- a collective attribute information model.

The generic DSA Information Model in turn establishes a framework for the following specific information models:

- a model for a DSA's distribution knowledge;
- a model for a DSA's replication knowledge.

6.4 Directory Administrative Authority Model

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

That part of the global DIT held by (the DSAs forming) a DMD is referred to as a *DIT Domain*. There is a one to one correspondence between DMDs and DIT Domains. The term DMD is used when referring to the management of the functional components of the Directory. The term DIT Domain is used when referring to the management of Directory Information. Two important points regarding this terminology are:

- A DIT Domain consists of one or more disjoint subtrees of the DIT (see 10.5). A DIT Domain shall not contain the root of the global DIT.
- The term DMD may also be used as a general term when both aspects of management are considered together.

An organization that manages a DMD (and the associated DIT Domain) is referred to as a *Domain Management Organization (DMO)*.

NOTE 1 – A DMO may be an Administration (i.e. a public telecommunications administration or other organization offering public telecommunications services) in which case the managed 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 ITU-T 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 Directory Specifications.

Figure 2 illustrates the relationship between a DMO, DMD and DIT Domain.

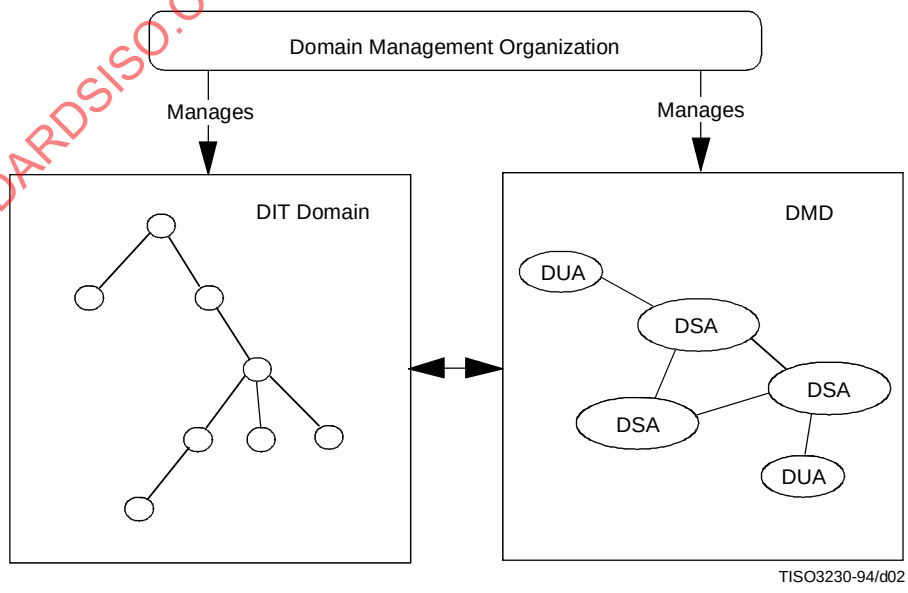


Figure 2 – Directory Management

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Management of a DUA by a DMO implies an ongoing responsibility for service to that DUA, e.g. maintenance, or in some cases ownership, by the DMO. The DMO may or may not elect to make use of the Directory Specifications to govern any interactions among DUAs and DSAs which are wholly within the DMD.

An agent of a DMO concerned with various aspects of Directory administration is referred to as an *Administrative Authority*. The term *administrative authority* (in lower case) refers to the power vested in an Administrative Authority by a DMO to execute policy.

NOTE 2 – A Directory Administrative Authority Model is specified in Section 4.

SECTION 3 – MODEL OF DIRECTORY USER INFORMATION

7 Directory Information Base

7.1 Definitions

For the purposes of this Directory Specification, the following definitions apply:

7.1.1 alias entry: An entry of the class "alias" containing information used to provide an alternative name for an object or alias entry.

7.1.2 direct superclass: Relative to a subclass – an object class from which the subclass is directly derived.

7.1.3 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.

7.1.4 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.

7.1.5 (directory) entry: A named collection of information within the DIB. The DIB is composed of entries.

7.1.6 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.

7.1.7 immediately superior

7.1.8 (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.

7.1.9 (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.

7.1.10 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.

7.1.11 object class: An identified family of objects (or conceivable objects) which share certain characteristics.

7.1.12 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.

7.1.13 subclass: Relative to one or more superclasses – an object class derived from one or more superclasses. The members of the subclass share all the characteristics of the superclasses and additional characteristics possessed by none of the members of those superclasses.

7.1.14 subordinate: The converse of superior.

7.1.15 superclass: Relative to a subclass – a direct superclass, or superclass to an object class that is a direct superclass (recursively).

7.1.16 superior: (Applying to entry or object) immediately superior, or superior to one which is immediately superior (recursively).

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

7.2 Objects

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).

NOTE 1 – The 'world' is generally that of telecommunications and information processing or some part thereof.

NOTE 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 this Directory Specification, but is a matter for the users and providers of the Directory in the context of their applications.

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 other object classes, in which case the members of the former class, the subclass, are also considered to be members of the latter classes, the superclasses. There may be subclasses of subclasses, etc., to an arbitrary depth.

7.3 Directory Entries

The DIB is composed of (*Directory) entries*. An entry is a named collection of information.

There are three kinds of entries:

- *Object entries*: Representing the primary collection of information in the DIB about a particular object. For any particular object there is precisely one object entry. The object entry is said to represent the object.
- *Alias entries*: Used to provide alternative names for object entries.
- *Subentries*: Representing a collection of information in the DIB used to meet administrative and operational requirements of the Directory. Subentries are discussed in Section 5.

A user view of the structure of directory entries is depicted in Figure 3 and described in 8.2.

Each entry contains an indication of the object classes, and their superclasses, to which the entry belongs.

Some object entries are specially designated for the purpose of Directory administration. These entries are termed administrative entries. The Directory user is not normally aware of this, and views these entries in the same way as other object entries.

7.4 The Directory Information Tree (DIT)

In order to satisfy requirements for the distribution and management of a very large DIB, and to ensure that entries can be unambiguously named and rapidly found, a flat structure 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 G.

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 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.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

A superior/subordinate relationship between objects can be derived from that between object 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 object entries for the second object. The terms *immediately subordinate object*, *immediate subordinate*, *superior* and *subordinate* (applied to objects) have their analogous meanings.

Permitted superior/subordinate relationships among objects are governed by the DIT structure definitions (see 12.3).

The Directory maintains, in addition to information concerning Directory entries, additional information regarding collections of Directory entries. Such collections may be *subtrees* (of the DIT) or *subtree refinements* (when not a true tree structure). See clause 11.

8 Directory Entries

8.1 Definitions

For the purposes of this Directory Specification, the following definitions apply:

8.1.1 attribute: Information of a particular type. Entries are composed of attributes.

8.1.2 user attribute: An attribute representing user information.

8.1.3 attribute hierarchy: The aspect of an attribute that permits a user attribute type to be derived from a more generic user attribute type. The relationship of the two attribute type definitions (which mandates certain behaviour of attributes corresponding to these attribute types) is thus hierarchical.

8.1.4 attribute subtype (subtype): An attribute type A is related to another attribute type B by the fact that either A has been derived from B, in which case A is a *direct* subtype of B, or A has been derived from an attribute type which is a subtype of B, in which case A is an *indirect* subtype of B.

8.1.5 attribute supertype (supertype): An attribute type B is related to another attribute type A by the fact that either A has been derived from B, in which case B is a *direct* supertype of A, or A has been derived from an attribute type which is a subtype of B, in which case B is an *indirect* supertype of A.

8.1.6 attribute type: That component of an attribute which indicates the class of information given by that attribute.

8.1.7 attribute value: A particular instance of the class of information indicated by an attribute type.

8.1.8 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 of a particular type.

8.1.9 auxiliary object class: An object class which is descriptive of entries or classes of entries and is not used for the structural specification of the DIT.

8.1.10 collective attribute: A user attribute whose values are the same for each member of an entry collection.

8.1.11 context: A property that can be associated with a user attribute value to specify information that can be used to determine the applicability of the value.

8.1.12 context assertion: A proposition, which may be true or false, regarding a context type and particular context values for that type, that determines the applicability of an attribute value.

8.1.13 context type: That component of a context which indicates its type or purpose.

8.1.14 context list: The set of contexts associated with an attribute value.

8.1.15 context value: A particular instance of the property indicated by a context type.

8.1.16 direct attribute reference: Reference (in the Directory and DSA abstract service) to one or more attribute values using the identifier of their attribute type.

8.1.17 distinguished value: An attribute value in an entry which may appear in the relative distinguished name of the entry.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

8.1.18 **entry collection**: A collection of entries belonging to an explicitly specified subtree or subtree refinement of the DIT.

8.1.19 **indirect attribute reference**: Reference (in the Directory and DSA abstract service) to one or more attribute values using the identifier of a supertype of their attribute type.

8.1.20 **matching rule**: A rule, forming part of the Directory Schema, which allows entries to be selected by making a particular statement (a matching rule assertion) concerning their attribute values.

8.1.21 **matching rule assertion**: A proposition, which may be true, false or undefined, concerning the presence in an entry of attribute values meeting the criteria defined by the matching rule.

8.1.22 **operational attribute**: An attribute representing operational and/or administrative information.

8.1.23 **structural object class**: An object class used for the structural specification of the DIT.

8.1.24 **structural object class of an entry**: With respect to a particular entry, the *single* structural object class used to determine the DIT Content Rule and DIT Structure Rule applying to the entry. This object class is indicated by the **structuralObjectClass** operational attribute. This object class is the most subordinate object class of the entry's structural object class superclass chain.

8.2 Overall Structure

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

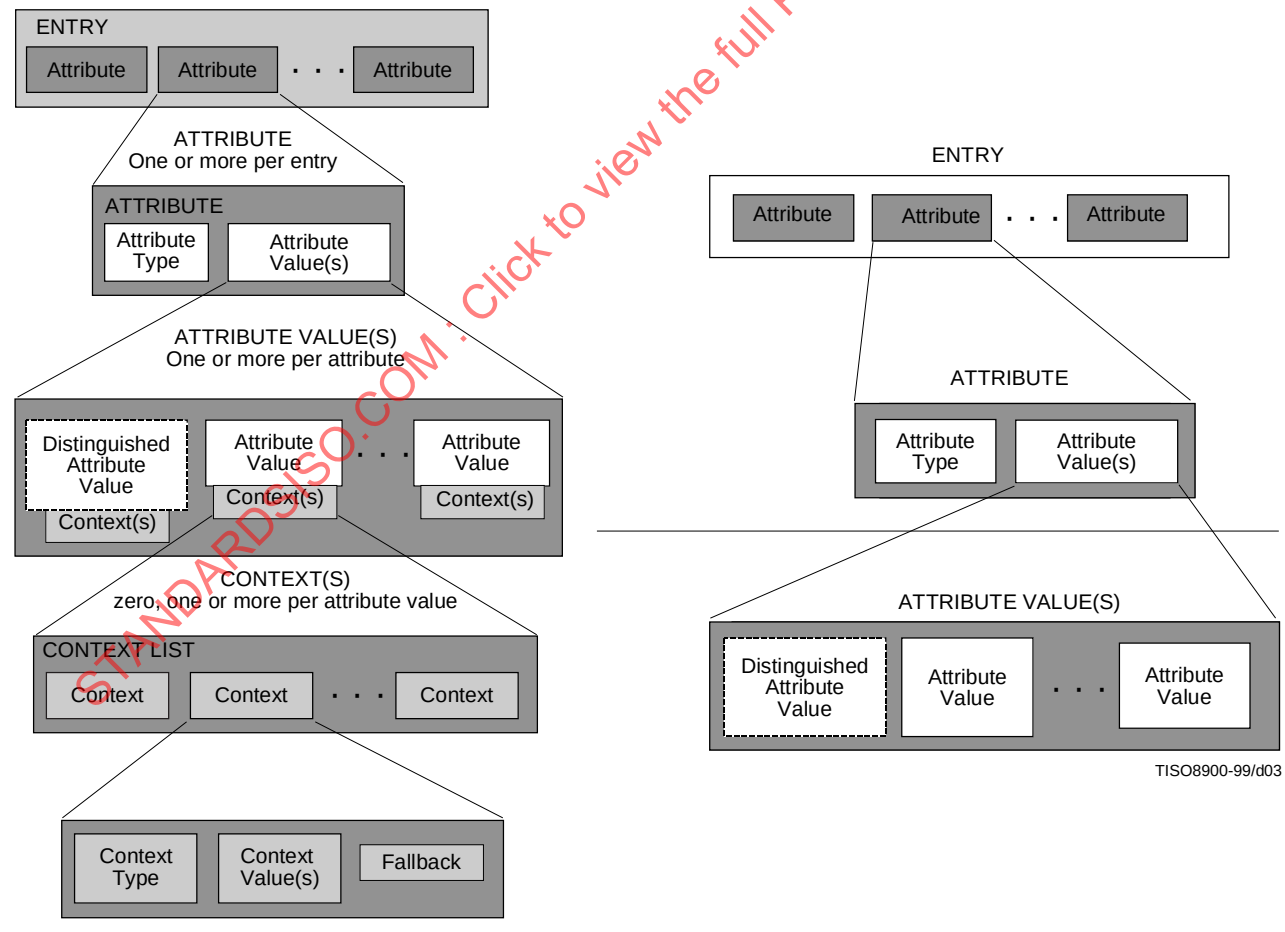


Figure 3 – Structure of an entry

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

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

NOTE 1 – 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.

An attribute consists of an *attribute type*, which identifies the class of information given by an attribute, and the corresponding *attribute values*, which are the particular instances of that class appearing in the entry. A user attribute value may have zero, one, or more contexts associated with it in its context list. Operational attribute values shall not have contexts.

NOTE 2 – Attribute types, attribute values, and contexts are described in 8.4, 8.5 and 8.7 respectively. Operational attributes are described in clause 11.

```
Attribute ::= SEQUENCE {
    type                ATTRIBUTE.&id ({ SupportedAttributes }),
    values              SET SIZE (0 .. MAX) OF ATTRIBUTE.&TYPE ({ SupportedAttributes}{@type}),
    valuesWithContext   SET SIZE (1 .. MAX) OF SEQUENCE {
    value               ATTRIBUTE.&Type ({SupportedAttributes}{@type}),
    contextList         SET SIZE (1 .. MAX) OF Context } OPTIONAL }
```

An attribute may be designated as single-valued or multi-valued. The Directory shall ensure that single-valued attributes have only a single value. This value may have a context list to associate properties with the attribute value. Attributes in storage shall have at least one value, but may at times appear to have zero values when transferred to or from storage (e.g. because values are hidden by access control).

8.3 Object Classes

Object classes are used in the Directory for a number of purposes:

- describing and categorizing objects and the entries that correspond to these objects;
- where appropriate, controlling the operation of the Directory;
- regulating, in conjunction with DIT structure rule specifications, the position of entries in the DIT;
- regulating, in conjunction with DIT content rule specifications, the attributes that are contained in entries;
- identifying classes of entry that are to be associated with a particular policy by the appropriate administrative authority.

Some object classes will be internationally standardized. Others will be defined by national administrative authorities and/or private organizations. This implies that a number of separate authorities will be responsible for defining object classes and unambiguously identifying them. This is accomplished by identifying each object class with an object identifier when the object class is defined. A notation for this purpose is provided in 12.3.3.

NOTE – An administrative authority may use object classes other than the useful object classes defined and registered in the Directory Specifications. An administrative authority may itself specify and register object classes, for example to supplement those defined in the Directory Specifications

An object class (a *subclass*) may be derived from an object class (its direct *superclass*) which is itself derived from an even more generic object class. For structural object classes, this process stops at the most generic object class, **top**. An ordered set of superclasses up to the most superior object class of an object class is its *superclass chain*.

An object class may be derived from two or more direct superclasses (superclasses not part of the same superclass chain). This feature of subclassing is termed *multiple inheritance*.

The specification of an object class identifies whether an attribute is mandatory or optional; this specification also applies to its subclasses. The subclass may be said to *inherit* the mandatory and optional attribute specification of its superclass. The specification of a subclass may indicate that an optional attribute of the superclass is mandatory in the subclass.

There are three kinds of object class:

- Abstract Object Classes;
- Structural Object Classes; and
- Auxiliary Object Classes.

Each object class is of precisely one of these kinds, and remains of this kind in whatever situation it is encountered within the Directory. The definition of each object class must specify what kind of object that it is.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

All entries shall be a member of the object class **top** and at least one other object class.

8.3.1 Abstract Object Classes

An abstract object class is used to derive other object classes, providing the common characteristics of such object classes. An entry shall not belong only to abstract object classes.

top is an abstract object class used as a superclass of all structural object classes.

8.3.2 Structural Object Classes

An object class defined for use in the structural specification of the DIT is termed a *structural object class*. Structural object classes are used in the definition of the structure of the names of the objects for compliant entries.

An object or alias entry is characterised by precisely one structural object class superclass chain which has a single structural object class as the most subordinate object class. This structural object class is referred to as the *structural object class of the entry*.

Structural object classes are related to associated entries:

- an entry conforming to a structural object class shall represent the real-world object constrained by the object class;
- DIT structure rules only refer to structural object classes; the structural object class of an entry is used to specify the position of the entry in the DIT;
- the structural object class of an entry is used, along with an associated DIT content rule, to control the content of an entry.

The structural object class of an entry shall not be changed.

8.3.3 Auxiliary Object Classes

Specific applications using the Directory will frequently find it useful to specify an *auxiliary object class* which may be used in the construction of entries of several types. For example, message handling systems make use of the auxiliary class MHS User (see ITU-T Rec. X.402 | ISO/IEC 10021-2) to specify a package of mandatory and optional message handling attributes for entry types whose structural object class is variable, e.g. Organizational Person or Residential Person.

In certain environments, there is a need to be able to add to or remove from the list of attributes permitted in an entry of a particular, perhaps standardized, class (or classes).

This requirement may be met by the definition and use of an auxiliary object class having semantics, known and maintained within a local community, which change from time to time as needed.

This requirement may also be met using the facilities of DIT content rule definitions to dynamically (i.e. without registration) allow the addition or exclusion of attributes from entries at particular points in the DIT (see 12.3.3).

Auxiliary object classes are descriptive of entries or classes of entries.

Therefore, besides being a member of the structural object class, an entry may be optionally a member of one or more auxiliary object classes.

An entry's auxiliary object classes may change over time.

NOTE – The unregistered object class facility, available in the 1988 edition of these Directory Specifications to support the requirements discussed in this clause, is now deprecated in favour of the use of DIT content rules.

8.3.4 Object Class Definition and the 1988 Edition of this Directory Specification

Object classes defined using the terminology of the 1988 edition of this Directory Specification will not be classified as one of structural, auxiliary or abstract.

Alias object classes specified using the terminology of the 1988 edition of this Directory Specification may be considered to be specified as either abstract, auxiliary or structural object classes and deployed in a subschema accordingly.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

8.4 Attribute Types

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 defining types and unambiguously identifying them. This is accomplished by identifying each attribute type with an object identifier when the type is defined. Using the notation of the **ATTRIBUTE** information object class defined in 12.4.6, an attribute type is defined as:

```
AttributeType ::= ATTRIBUTE.&id
```

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

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 classes and superclasses to which the object belongs.

b) **aliasedEntryName** – An attribute of this type appears in every alias entry, and holds the name (see 8.5) of the entry which the alias entry references.

These attributes are defined in 12.4.6.

The types of user attributes which shall or which may appear within an object or alias entry are governed by rules applying to the indicated object classes as well as by the DIT content rule for that entry (see 12.7). The types of attributes which may appear in a subentry are governed by the rules of the system schema.

Some Directory entries may contain special attributes not normally visible to the Directory User. These attributes are called operational attributes and are used to meet the administrative and operational requirements of the Directory. Operational attributes are discussed in more detail in Section 5.

8.5 Attribute Values

Defining an attribute also involves specifying the syntax, and hence data type, to which every value in such attributes shall conform. Using the notation of the **ATTRIBUTE** information object class defined in clause 12.4.6, an attribute value is defined as:

```
AttributeValue ::= ATTRIBUTE.&Type
```

An attribute value may be designated as a *distinguished value*, in which case the attribute value can form part of the relative distinguished name of the entry (see 9.3). It is possible to have multiple distinguished values differentiated by context, as described in 9.3.

8.6 Attribute Type Hierarchies

When defining an attribute type, the characteristics of some more generic attribute type may optionally be employed as the basis of the definition. The new attribute type is a *direct subtype* of the more generic attribute type, the *supertype*, from which it is derived.

Attribute hierarchies allow access to the DIB with varying degrees of granularity. This is achieved by allowing the value components of attributes to be accessed by using either their specific attribute type identifier (a direct reference to the attribute) or by the identifier of a more generic attribute type identifier (an indirect reference).

Semantically related attributes may be placed in a hierarchical relationship, the more specialized being placed subordinate to the more generalized. Searching for, or retrieving attributes and their values is made easier by quoting the more generalized attribute type; a filter item so specified is evaluated for the more specialized types as well as for the quoted type; a context assertion specified for the more generalized attribute type is also applied to the more specialized type.

Where subordinate specialized types are selected to be returned as part of a search result these types shall be returned if available. Where the more general types are selected to be returned as part of a search result both the general and the specialized types shall be returned, if available. An attribute value shall always be returned as a value of its own attribute type.

For an entry to contain a value of an attribute type belonging to an attribute hierarchy, that type must be explicitly included either in the definition of an object class to which the entry belongs, or because the DIT content rule applicable to that entry permits it.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

All of the attribute types in an attribute hierarchy are treated as distinct and unrelated types for the purpose of administration of the entry and for user modification of entry content.

An attribute value stored in a Directory object or alias entry is of precisely one attribute type. The type is indicated when the value is originally added to the entry.

8.7 Contexts

The information model may be refined by associating with attribute values properties called contexts. Associated with any user attribute value may be a list of contexts which provide additional information that can be used to determine the applicability of the attribute value.

NOTE 1 – For example, contexts can be used to associate a particular language, time, or locale with an attribute value.

Each context consists of a type field, a value field whose syntax is determined by the type, and a **fallback** flag. Using the notation of the **CONTEXT** information object class defined in 12.8, a Context is defined as:

```
Context ::= SEQUENCE {
    contextType     CONTEXT.&id ({SupportedContexts}),
    contextValues   SET SIZE (1..MAX) OF CONTEXT.&Type ({SupportedContexts}{@contextType}),
    fallback        BOOLEAN DEFAULT FALSE }
```

contextType is an **OBJECT IDENTIFIER**, and is specified using the **CONTEXT** information object class defined in 12.8. It specifies the particular property represented by the Context.

contextValues is the set of one or more values of the property specified by **contextType** that are associated with the particular attribute value.

fallback is used to designate one or more attribute values for specific behaviour in relation to a context type. In addition to having any specific contextValues of that context type associated with it, an attribute value for which fallback is **TRUE** for a given **contextType** is:

- considered as being associated with any value of the given **contextType** for which no other values of the same attribute are otherwise associated. Thus a context assertion of this context type that fails to match any values of the attribute based on the rules for matching **contextValues** shall match with any attribute value for which **fallback** is **TRUE** for this context type.

  NOTE 2 – For example, an attempt to select the attribute value associated with a particular language shall yield those values with **fallback** set to **TRUE** if none of the attribute values is otherwise associated with the chosen language.

- considered as a value to preserve during an operation which resets attribute values for a given attribute type. A Modify (reset value) removes all values of a chosen attribute type which have an associated context for which the **fallback** is set FALSE.

  NOTE 3 – Modify (reset value) is further described in 11.3.2 of ITU-T Rec. X.511 | ISO/IEC 9594-3.

An attribute value without contexts, or one whose context list does not contain a context of a specific type, is considered to be applicable under all context values of that specific type.

NOTE 4 – For example, a selection based on the French context value of a language context shall select an attribute value that does not have any language context specifically associated with it (as well as those attribute values having the French language context associated with them specifically).

All contexts in an attribute value's context list shall be of distinct context types.

Context information associated with attribute values may be retrieved along with the attribute values (e.g. to differentiate between those attribute values). A user of the Directory may also make use of contexts to refine selection and retrieval of information during Directory operations.

8.8 Matching Rules

8.8.1 Overview

Of paramount importance to the Directory is the ability to be able to select a set of entries from the DIB based on assertions concerning attribute values held by these entries.

A matching rule allows entries to be selected by making a particular assertion concerning their attribute values.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The most primitive type of assertion is the attribute value assertion. More complex assertions may be supported using matching rule assertions. A matching rule assertion is a proposition, which may be true, false or undefined, concerning the presence in an entry of attribute values meeting the criteria defined by the matching rule.

An attribute value or matching rule assertion is evaluated based on the matching rule associated with the assertion.

A matching rule is defined through the specification of:

- the range of attribute syntaxes supported by the rule;
- the specific types of matches supported by the rule;
- the syntax required to express an assertion of each specific type of match;
- rules for deriving a value of the assertion syntax from a value of the attribute syntax, if required.

NOTE – No restrictions are placed on the matching rules that may be defined to support a particular application. However, rules defined to support one particular application may not be widely supported by DUAs and DSAs. Wherever possible, the matching rules defined in ITU-T Rec. X.520 | ISO/IEC 9594-6 should be used in preference to the specification of new ones.

Sometimes there will be a one to one correspondence between a matching rule and the types of matches supported. For example, the Directory Abstract Service supports a presence matching rule to detect the presence of an attribute in an entry.

Sometimes there will be a many to many correspondence between a rule and the types of matches supported. For example, the Directory Abstract Service supports a generic ordering rule allowing greater than or equal and less than or equal types of matches.

8.8.2 Attribute Value Assertions

An attribute value assertion (AVA) is 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 of a particular type. It involves an attribute type, an asserted attribute value, and optionally an assertion about contexts associated with the attribute value:

```
AttributeValueAssertion  ::=  SEQUENCE {
      type                    ATTRIBUTE.&id ({SupportedAttributes}),
      assertion               ATTRIBUTE.&equality-match.&AssertionType ({SupportedAttributes}{@type}),
      assertedContexts        CHOICE {
            allContexts             [0]  NULL,
            selectedContexts        [1]  SET OF ContextAssertion } OPTIONAL }

ContextAssertion  ::=  SEQUENCE {
      contextType             CONTEXT.&id({SupportedContexts}),
      contextValues           SET SIZE (1..MAX) OF
                  CONTEXT.&Assertion ({SupportedContexts}{@contextType})}
```

The syntax of the **assertion** component of an AVA is determined by the equality matching rule defined for the attribute type, and may be different from the syntax of the attribute itself.

8.8.2.1 Evaluation of an AVA

An AVA is:

a) undefined, if any of the following holds:

1) the attribute type is unknown;

2) the attribute type has no equality matching rule,

3) the value does not conform to the data type indicated by the syntax of the assertion of the attribute's equality matching rule;

NOTE – 2) and 3) normally indicate a faulty AVA; 1) however, may occur as a local situation (e.g. a particular DSA has not been configured with support for that particular attribute type).

b) true, if the entry contains an attribute of that type, and the attribute contains a value of that value, and the value contains a context that matches the **assertedContexts** as described in 8.8.2.2;

c) false, otherwise.

8.8.2.2 Use of assertedContexts or context assertion defaults

The inclusion of **assertedContexts** within an **AttributeValueAssertion** is optional. If **assertedContexts** is specified, then the **assertion** shall be evaluated only against those values of the attribute for which the **assertedContexts** is true, as defined in 8.8.2.3.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

If **assertedContexts** is not provided within an **AttributeValueAssertion**, then a default context assertion may be applied in the same manner; that is, the **assertion** shall be evaluated only against those values of the attribute for which, as defined in 8.8.2.3, the default context assertion is true. There are three potential sources for a default context assertion: that specified for the operation as a whole, that available within subentries in the DIT, and that available locally in the DSA. They are applied as follows:

1) If **assertedContexts** is not provided within an **AttributeValueAssertion**, then any context assertion for the given attribute type which has been supplied for the operation as a whole, as part of **operationContexts** as described in 7.3 of ITU-T Rec. X.511 | ISO/IEC 9594-3, shall be applied.

2) If the user has not provided **assertedContexts** for the AVA and there is no context assertion for the given attribute type which has been supplied for the operation as a whole, then the default context assertion for the given attribute type in the context assertion subentries (if any) controlling the entry shall be applied, as described in 13.7.

3) If there is no context assertion through steps 1) and 2) above, the DSA may apply a locally-defined default context assertion for the given attribute type. Such a default shall typically reflect local parameters, such as the language or location of the place of deployment of the DSA, or the current time of day, but may be tailored differently by the DSA for each DUA to which it responds.

4) If no context assertion is available from any of these sources, then the **assertion** shall be evaluated against all values of the attribute.

8.8.2.3 Evaluation of assertedContexts

assertedContexts is true if:

a) **allContexts** is specified (this permits a context assertion to override any default context assertion that might otherwise be applied if **assertedContexts** were omitted from the **AttributeValueAssertion**); or

b) each **ContextAssertion** in **selectedContexts** is true as described in 8.8.2.4.

assertedContexts is false otherwise.

8.8.2.4 Evaluation of a ContextAssertion

A **ContextAssertion** is true for a particular attribute value if:

a) the attribute value has a context of the same **contextType** of the **ContextAssertion** and any of the stored **contextValues** of that context matches with any of the asserted **contextValues** according to the definition of how a match is determined for that **contextType**; or

b) the attribute value contains no contexts of the asserted **contextType**; or

c) none of the other attribute values for the attribute satisfies the **ContextAssertion** according to 1) or 2) in 8.8.2.2 above, but the attribute value does contain a context of the asserted **contextType** with the **fallback** set to **TRUE**.

A **ContextAssertion** is false otherwise.

8.8.3 Built-in Matching Rule Assertions

A number of categories of related matching rules, whose semantics are generally understood and applicable to values of many different types of attributes, are understood by the Directory:

– present;
– equality;
– substrings;
– ordering;
– approximate match.

Syntax for asserting certain types of matches associated with these categories of matching rules has been built into the Directory Abstract Service:

– a **present** syntax for the present rule;
– an **equality** syntax for equality rules;
– **greaterOrEqual** and **lessOrEqual** syntaxes for ordering rules;
– **initial**, **any** and **final** syntaxes for substrings rules;
– an **approximateMatch** syntax for approximate matching rules.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The **present** syntax may be used for any attribute of any type. The present match tests for the presence of any value of a particular type.

Specific equality, substrings and ordering matching rules may be associated with an attribute type when it is defined. These specific rules are used when evaluating assertions of the equality, ordering and substrings rules made using the syntax built-in to the Directory Abstract Service. If specific rules are not provided, then assertions made concerning these attributes are undefined.

The **approximateMatch** syntax supports an approximate matching rule whose definition is a local matter to a DSA.

8.8.4 Matching Rule Requirements

In order for the Directory to behave in a consistent and well-defined manner, it is necessary that certain restrictions be placed upon the matching rules that shall be used in conjunction with the syntax that has been built into the Directory Abstract Service.

For an equality matching rule in which the syntax of the assertion is different from the attribute syntax to which the matching rule applies, rules for deriving a value of the syntax of the assertion from a value of the attribute syntax shall be supplied.

Equality matching rules for attributes used for naming shall be transitive, commutative and have an assertion syntax identical to the attribute syntax.

A transitive matching rule is characterized by the fact that if a value **a** matches a value **b**, and if that value **b** matches a third value **c**; then value **a** must necessarily match value **c** using the rule.

A commutative matching rule is characterized by the fact that if a value **a** matches a value **b** then that value **b** must necessarily match the value **a**. The attribute **presentationAddress** is an example of an attribute supporting an attribute syntax whose matching rule is not commutative.

With respect to a specific attribute type, the equality and ordering rules (if both present) must always be related in at least the following respect: two values are equal using the equality relation if and only if they are equal using the ordering relation. In addition, the ordering relation must be well-ordered; that is, for all x, y and z for which x precedes y and y precedes z according to the relation, then x must precede z.

NOTE – These requirements imply that when ordering is defined, it also defines equality.

With respect to a specific attribute type, the equality and substrings rule (if both present) must always be related in at least the following respect: for all x and y that match according to the equality relation, then for all values z of the substring relation, the result of evaluating the assertion against the value x must equal the result of evaluating the assertion against the value y. That is, two values that are indistinguishable using the equality relation must also be indistinguishable using the substrings relation.

8.8.5 Object Identifier and Distinguished Name equality matching rules

There are a number of equality matching rules used to evaluate attribute value assertions which the Directory knows about and uses for its own purposes. They include:

- **objectIdentifierMatch**: This rule is used to match attributes with **ObjectIdentifier** syntax.
- **distinguishedNameMatch**: This rule is used to match attributes with **DistinguishedName** syntax.

8.9 Entry Collections

8.9.1 Overview

A collection of object and alias entries may have certain common characteristics (e.g. certain attributes that have the same value for each entry of the collection) because of some common characteristic or shared relationship of the corresponding objects. Such a grouping of entries is termed an entry collection.

Entry collections may contain object and alias entries that are related by their position in the DIT. These collections are specified as subtrees or subtree refinements as described in Section 5.

An entry may belong to several entry collections subject to administrative limitations imposed in Section 5.

8.9.2 Collective Attributes

When user attributes are shared by the entries of an entry collection, they are termed *collective attributes.*

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

It is also permissible that the same collective attribute be independently associated with two or more of these collections. In such cases the entry's collective attribute has multiple values. Collective attributes shall, therefore, always be specified as multi-valued.

Although they appear to users of the Directory interrogation operations as entry attributes, collective attributes are treated differently from entry attributes in the Directory information model. This difference is manifested to users of the Directory modification operations in that collective attributes cannot be administered (i.e. modified) via the entries in which they appear but must be administered via their associated subentries.

NOTE – The independent sources of these values is not manifested to the users of the Directory interrogation operations.

For a collective attribute to appear in an entry, the presence of that attribute type must be permitted according to the DIT content rule governing the entry.

Entries may specifically exclude a particular collective attribute. This is achieved through the use of the **collectiveExclusions** attribute, described in 11.7 and defined in 13.6.

9 Names

9.1 Definitions

For the purposes of this Directory Specification, the following definitions apply:

9.1.1 alias, alias name: An alternative name for an object, provided by the use of alias entries.

9.1.2 (alias) dereferencing: The process of converting an object's alias name to its distinguished name.

9.1.3 distinguished name (of an entry): Every object entry, alias entry, and subentry has at least one distinguished name. If any RDN for the entry or any superior entry includes an attribute for which there exist multiple distinguished values differentiated by context (as described in 9.3), then the entry shall have multiple distinguished names differentiated by context. The *primary distinguished name* is that distinguished name in which each RDN has the primary distinguished value of each contributing attribute as the main value in the RDN construct.

9.1.4 (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).

9.1.5 (entry) name: A construct that singles out a particular entry from all other entries.

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

9.1.7 naming authority: An authority responsible for the allocation of names in some region of the DIT.

9.1.8 relative distinguished name (RDN): A set of one or more attribute type and value pairs, each of which matches a distinct distinguished attribute value of the entry.

9.2 Names in General

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, denotes just one object. However, a name need not be unique, that is, be the only name that unambiguously denotes the object. A (directory) name also identifies an entry. This entry is either an object entry that represents the object or an alias entry which contains information that helps the Directory to locate the entry that represents the object.

NOTE 1 – The set of names of an object thus comprises the set of alias names for the object, together with the distinguished names of the object.

An object can be assigned a distinguished name without being represented by an entry in the Directory, but this name is then the name its object entry would have had were it represented in the Directory.

Syntactically, each name for an object or entry is an ordered sequence of relative distinguished names (see 9.3).

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

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

RDNSequence  ::=  SEQUENCE OF  RelativeDistinguishedName

DistinguishedName  ::=  RDNSequence
```

NOTE 2 – Names which are formed in other ways than as described herein are a possible future extension.

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.

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

9.3 Relative Distinguished Names

Each object and entry has at least one *relative distinguished name (RDN)*. An RDN of an object or alias entry consists of a set of attribute type and value (with optional context list) pairs, each of which matches, using the equality matching rule and the applicable context matching rule, a distinct distinguished attribute value of the entry.

Any attribute contributing to an RDN may have more than one distinguished value, differentiated by context, as described below. This provides alternative RDNs for the same object. Within an attribute's set of distinguished values (differentiated by context), precisely one of them is designated the primary distinguished value. The *primary relative distinguished name* of an object comprises the set of primary distinguished values from the set of attributes that comprise the RDN. When conveyed in protocol, each attribute in an RDN signals the primary distinguished value (if it is present) and may optionally include a context for the value and additional alternative attribute values with context. In this case, each attribute value with its context matches a distinct distinguished attribute value of the entry for the attribute type according to the applicable equality matching rule and context matching rules.

NOTE 1 – The equality matching rule can be used because for naming attributes, the attribute syntax and the assertion syntax of the equality matching rule are the same. Similarly, for contexts that may be used to differentiate distinguished values in a naming attribute, the context syntax and the context assertion syntax are the same.

The RDNs of all of the entries with a particular immediate superior are distinct irrespective of any associated context lists. It is the responsibility of the relevant naming authority for an entry to ensure that this is so by appropriately assigning distinguished attribute values. Allocation of RDNs is considered an administrative undertaking that may or may not require some negotiation between involved organizations or administrations. This Directory Specification does not provide such a negotiation mechanism, and makes no assumption as to how it is performed.

```
RelativeDistinguishedName  ::=  SET SIZE (1..MAX) OF AttributeTypeAndDistinguishedValue

AttributeTypeAndDistinguishedValue ::=  SEQUENCE  {
        type                    ATTRIBUTE.&id ({SupportedAttributes}),
        value                   ATTRIBUTE.&Type({SupportedAttributes}{@type}),
        primaryDistinguished    BOOLEAN DEFAULT TRUE,
        valuesWithContext       SET SIZE (1 .. MAX) OF SEQUENCE {
                distingAttrValue        ATTRIBUTE.&Type ({SupportedAttributes}{@type}) OPTIONAL,
                contextList             SET SIZE (1 .. MAX) OF Context } OPTIONAL }
```

The set that forms an RDN contains exactly one **AttributeTypeAndDistinguishedValue** for each attribute which contains distinguished values in the entry; that is, a given attribute type cannot appear twice in the same RDN.

An attribute value that has been designated to appear in an RDN is called a distinguished value. There may be other values of the same attribute that are not distinguished values and thus may not be used in an RDN. An attribute may have multiple distinguished values only if they are differentiated by associated context. This allows an object to have alternative names differentiated by contexts. This is the only case where an attribute may have more than one distinguished value. In that case, the distinguished value shall have context lists containing the same context type(s), the context values of which shall provide that only one of the distinguished values is applicable given any specific context.

An RDN for a given entry is formed by using one distinguished value from each attribute that has distinguished values. The simplest case is an entry that has one distinguished value; it thus has one RDN, formed by using that distinguished value. More than one attribute in an entry may contribute to the RDN. If each contributing attribute has only one distinguished value, then the entry has a single RDN, formed by using the distinguished value for each attribute. If any of the contributing attributes has multiple distinguished values differentiated by context, then the entry has multiple RDNs, each formed by using one of the possible combinations in which one distinguished value is chosen for each attribute type forming the RDN.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Each RDN for an entry shall contain a **type** and **value** pair for each given attribute type forming part of the RDN. **primaryDistinguished** is used to indicate that the **value** is the primary distinguished value of that attribute type. **valuesWithContext** is used to convey the context list for the distinguished attribute value in **value** when necessary to do so. It is also used to convey in a single RDN, some or all of the other distinguished values of the same attribute type. Each **distingAttrValue** is accompanied by its **contextList**. The **distingAttrValue** is only omitted for the distinguished value that appears in **value**; this is how the context list for that value is made present in the RDN.

One and only one of the distinguished values for a given attribute type in an entry shall be considered the *primary distinguished value* for that attribute type. This value shall be used as the **value** in the **AttributeTypeAndDistinguishedValue** when forming the primary relative distinguished name of the object (see 9.8 and 9.6). The *primary relative distinguished name* is an RDN in which the primary distinguished values for each attribute in the RDN appear in the **value** components of each **AttributeTypeAndDistinguishedValue** in the RDN. Context and alternative distinguished values may appear in the valuesWithContext component of each **AttributeTypeAndDistinguishedValue.**

The RDN may be modified if necessary by complete replacement of all the distinguished values of all contributing attributes.

NOTE 2 – 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.

NOTE 3 – Changing the RDN of a non-leaf entry automatically changes the name of subordinate entries.

NOTE 4 – The context in which a particular attribute type and value forming part of an RDN is applicable is independent of the contexts associated with any other part of that RDN or other RDNs in a distinguished name.

NOTE 5 – For example, a valid distinguished name for an entry can be formed by combining an RDN designated as the Language = French variant of that entry's RDN with the Language = English DN of its superior entry).

9.4 Name Matching

It is often necessary in the operation of the Directory to determine if two names match. This requires that corresponding RDNs be matched. The general approach to name matching is described here; specific approaches for particular uses for name matches are described where appropriate.

A purported RDN is said to match a target RDN if each **AttributeTypeAndDistinguishedValue** in the purported RDN matches with the **AttributeTypeAndDistinguishedValue** for the same attribute type in the target RDN. There is a match if the purported **value** or any **distingAttrValue** of the purported **AttributeTypeAndDistinguishedValue** matches either the target **value** or any **distingAttrValue** in the target **AttributeTypeAndDistinguishedValue**. **primaryDistinguished**, if present in either the purported or target **AttributeTypeAndDistinguishedValue**, is ignored for matching.

NOTE 1 – The equality matching rule can be used because for naming attributes, the attribute syntax and the assertion syntax of the equality matching rule are the same.

NOTE 2 – Note that there is no guarantee that every distinguished value for a given naming attribute is present in the **AttributeTypeAndDistinguishedValue** for that attribute type in a given RDN. Two RDNs for the same object could be formed using different distinguished values (differentiated by context) for the same attribute type. If there is no overlap in the sets of distinguished values for a given attribute that each uses, then they will fail to match, even though the purported RDN and target RDN are alternative RDNs for the same object. How this could occur, and the impact of this, depends on the reason for name matching (e.g. name resolution, access control, filtering).

If matching attribute values are not found as a result of the above, then the RDNs do not match. If matching attribute values are found, then there shall also be a match between associated contexts for those values, if present, before the attribute type and value pairs are considered to match. Each context in the purported attribute value's context list is considered a context assertion against the matching target attribute value's context list, and shall evaluate to true as described in 8.8.2.4 in order for the contexts to be considered a match. **fallback** in the purported contexts is ignored when forming the context assertions.

NOTE 3 – The purported contexts can be used as context assertions in this way because the context assertion syntax is the same as the context syntax for context types that may be used with distinguished values.

If **valuesWithContext** is not present in a purported RDN, then context assertions supplied as part of the operation, or defaults that are set up to be applied to an operation shall also be applied as described in 8.8.2.2. The exception to this is for the case of name matching during name resolution during a Directory operation; in that case, no context assertions are applied if none is available in **valuesWithContext.**

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

9.5 Names returned during operations

Many Directory operations return the name of an entry. When an operation returns a name for an entry, or names for multiple entries, it shall return the primary distinguished name for each entry and may return in addition alternative distinguished name information and context information (see 7.7 of ITU-T Rec. X.511 | ISO/IEC 9594-3).

9.6 Names held as attribute values or used as parameters

Where a name is held as an attribute value within some other attribute, or passed as an attribute value in some exchange (e.g. an alias pointer), there is always the question of whether the name held can be an alternative distinguished name or must be the primary distinguished name, whether it can contain alternative distinguished values, and whether it can include context information. Specific restrictions are mentioned where necessary throughout these Directory Specifications.

NOTE – Annex O includes suggestions for improving interoperability with pre-1997 edition systems and ensuring predictable behaviour in regards to using contexts with names.

9.7 Distinguished Names

The *distinguished name* of a given object is defined as that name which consists of 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 is the distinguished name of the object entry.

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

NOTE 2 – ISO/IEC 7498-3 defines the concept of a primitive name. A distinguished name can be used as a primitive name for the object it identifies.

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

Alias entries also have distinguished names; however, this name cannot be the distinguished name of an object. When this distinction needs to be made, the complete term "distinguished name of an alias entry" is used. The distinguished name of an alias entry is defined, as for the distinguished name of an object entry, to be the sequence of RDNs of the alias entry and those of all of its superior entries (in descending order).

It also proves convenient to define the 'distinguished name' of the root, although this can never be the distinguished name of an object. The distinguished name of the root is defined to be an empty sequence.

If any attribute contributing to an RDN within the distinguished name for an object has multiple distinguished values differentiated by contexts, then that object has multiple distinguished names. Each unambiguously identifies the object. The primary distinguished name is that distinguished name for which every RDN is a primary RDN. When conveyed in protocol, the primary distinguished name is formed by using the primary distinguished value as **value** in the **AttributeTypeAndDistinguishedValue** for each attribute in each RDN forming the name. Alternative distinguished names are formed by using alternative distinguished values for attributes in one or more RDNs. In some uses for a name, the primary distinguished name shall be used. In other cases, alternative distinguished names may be used. Since the **AttributeTypeAndDistinguishedValue** in RDNs may include alternate distinguished values in the **valuesWithContext** component, any distinguished name may include alternative values within its RDNs.

NOTE 4 – The distinguished name is said to include alternative names when an RDN includes multiple distinguished values for any contributing attribute.

Context information may be included with a distinguished name in the **valuesWithContext** component within any RDN. Wherever names are used throughout these Directory Specifications, it is specified if the name shall be the primary distinguished name, if the name may include alternative values, and if context information may be included. Where there is no explicit statement, alternative distinguished names may be used, and the name may include alternative values and/or context information.

NOTE 5 – Any requirement to use a primary distinguished name in protocol instead of an alternative distinguished name need not be reflected to the end user.

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

9.8 Alias Names

An *alias*, or *an alias name*, for an object is a an alternative name for an object or object entry which is provided by the use of alias entries.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

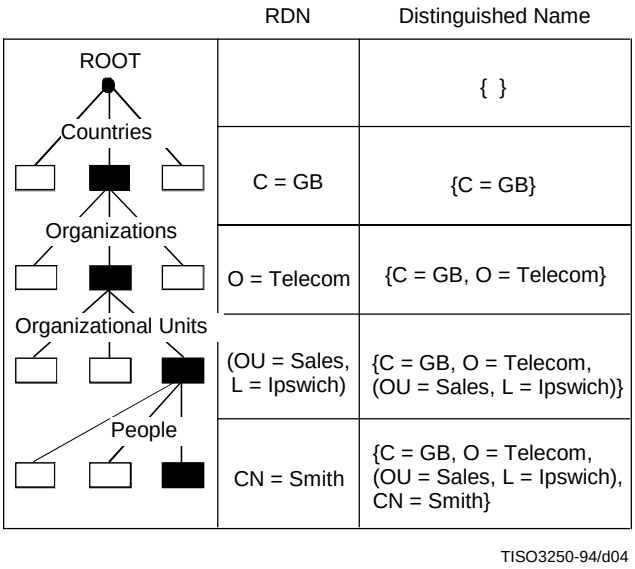


Figure 4 – Determination of distinguished names

Each alias entry contains, within the **aliasedEntryName** attribute, a name of some object. The distinguished name of the alias entry is thus also a name for this object.

NOTE 1 – The name within the **aliasedEntryName** is said to be pointed to by the alias. It does not have to be the distinguished name of any entry.

NOTE 2 – The **AliasedEntryName** attribute value may be the primary distinguished name or any alternative distinguished name if such exist. Consistency and interworking with pre-1997 DSAs may be affected if the primary distinguished name is not used.

The conversion of an alias name to an object name is termed (alias) dereferencing and comprises the systematic replacement of alias names, where found within a purported name, by the value of the corresponding **aliasedEntryName** attribute. The process may require the examination of more than one alias entry.

Any particular entry in the DIT may have zero or more alias names. It therefore follows that several alias entries may point to the same entry. An alias entry may point to an entry that is not a leaf entry and may point to another alias entry.

An alias entry shall have no subordinates, so that an alias entry is always a leaf entry.

Every alias entry shall belong to the **alias** object class which is defined in 12.3.3.

SECTION 4 – DIRECTORY ADMINISTRATIVE MODEL

10 Directory Administrative Authority model

10.1 Definitions

For the purposes of this Directory Specification, the following definitions apply:

10.1.1 administrative area: A subtree of the DIT considered from the perspective of administration.

10.1.2 administrative entry: An entry located at an administrative point.

10.1.3 administrative point: The root vertex of an administrative area.

10.1.4 administrative user: A representative of an Administrative Authority. The full definition of the administrative user concept is outside the scope of this Directory Specification.

10.1.5 autonomous administrative area: A subtree of the DIT whose entries are all administered by the same Administrative Authority. Autonomous administrative areas are non-overlapping.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

10.1.6 DIT domain administrative authority: An Administrative Authority in its role as the entity having responsibility for the administration of a part of the DIT.

10.1.7 DIT domain policy: An expression of the general goals and acceptable procedures for a DIT Domain.

10.1.8 DMD administrative authority: An Administrative Authority in its role as the entity responsible for the administration of a DMD.

10.1.9 DMD policy: A policy governing the operation of the DSAs in a DMD.

10.1.10 DMO policy: A policy defined by a DMO, expressed in terms of DMD and DIT Domain policies.

10.1.11 inner administrative area: A specific administrative area whose scope is wholly contained within the scope of another specific administrative area of the same type.

10.1.12 policy: An expression by an Administrative Authority of general goals and acceptable procedures.

10.1.13 policy attribute: A generic term for any Directory operational attribute which expresses policy.

10.1.14 policy object: An entity with which a policy is concerned.

10.1.15 policy procedure: A rule defining how a set of policy objects should be considered and what actions should be taken as a result of this consideration.

10.1.16 policy parameter: A policy procedure is characterized by certain *policy parameters* which are subject to configuration (i.e. choice) by an Administrative Authority.

10.1.17 specific administrative area: A subset (in the form of a subtree) of an autonomous administrative area defined for a particular aspect of administration: access control, subschema or entry collection administration. When defined, specific administrative areas of *a particular kind* partition an autonomous administrative area.

10.1.18 specific administrative point: The root vertex of a specific administrative area.

10.2 Overview

A fundamental objective of the Directory information model is to consider well-defined collections of entries so that they may be administered consistently as a unit. This clause clarifies the nature and scope of the authorities responsible for that administration and the means by which their authority is exercised.

The concept of policy, defined in 10.3, provides the mechanism by which Administrative Authorities exercise control of the Directory.

Some aspects of the Directory Administrative Model are supported by the Model of Directory Administrative and Operational Information (see clause 11). This is to allow the modelling of information required for the regulation of Directory user information and for other administrative purposes.

Other aspects of the Directory Administrative Model require support for the distribution of administrative and operational information among the component parts of the Directory, i.e. DSAs. Clauses 20 through 22 describe a DSA Information Model to support these requirements.

10.3 Policy

A *policy* is an expression by an Administrative Authority, acting as an agent of the DMO, of general goals and acceptable procedures. A policy is defined in terms of rules that are to be enforced (by the Directory, if appropriate) and in terms of aspects within which an administrative user has some degree of freedom of action and specific responsibilities.

An Administrative Authority expresses DMO policy in terms of:

- DIT Domain Policy;
- DMD Policy.

These policies may be expressed as policy attributes. A model of DIT policies is defined in 10.6.

NOTE – Clause 13 defines the system schema necessary to support the administration of collective attributes. Clause 14 defines a framework for supporting subschema administration policies. Clause 16 defines a framework supporting access control policies.

DMD policies relate specifically to DSAs as components of the distributed Directory. These DMD policies are described in 10.7 which defines a model for DSA administration.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Finally, there are policies which relate to external matters (such as bilateral agreements between DMOs) and are therefore not further described here.

A *policy object* is an entity with which a policy is concerned (e.g. a subschema administrative area is a policy object).

A *policy procedure* is a rule defining how a set of policy objects should be considered and what actions should be taken (and under what circumstances) as a result of this consideration (e.g. clause 14 defines subschema administration policy procedures).

A policy procedure is characterized by certain *policy parameters* which are subject to configuration (i.e. choice) by an Administrative Authority.

Operational attributes are used to represent policy parameters. The values of such an attribute form an expression of some or all of the policy parameter it represents.

10.4 Specific administrative authorities

The administration of a DIT Domain involves the execution of four functions related to different aspects of administration:

- naming administration;
- subschema administration;
- security administration;
- collective attribute administration.

A *specific Administrative Authority* is an Administrative Authority in its role as the entity responsible for one of these specific aspects of DIT Domain policy.

The term *Naming Authority* (see clause 9) identifies the role of the Administrative Authority as it pertains to the allocation of names and administration of the structure of these names. A role of the Subschema Authority is to implement these naming structures in the subschema.

The term *Subschema Authority* identifies the role of the Administrative Authority as it pertains to the establishment, administration and execution of the subschema policy controlling the naming and content of entries in a DIT Domain. Clause 14 describes Directory support of Subschema Administration.

The term *Security Authority* (see ITU-T Rec. X.509 | ISO/IEC 9594-8) identifies the role of the Administrative Authority as it pertains to the establishment, administration and execution of a security policy governing the behaviour of the Directory with respect to entries in a DIT Domain.

The term *Collective Attribute Authority* identifies the role of the Administrative Authority as it pertains to the establishment and administration of collective attributes (see 11.7) in a DIT Domain.

10.5 Administrative areas and administrative points

10.5.1 Autonomous administrative areas

Each entry in the DIT is administered by precisely one Administrative Authority (which may operate in different roles). An *autonomous administrative area* is a subtree of the DIT whose entries are all administered by the same Administrative Authority.

The DIT Domain may be partitioned into one or more non-overlapping autonomous administrative areas.

The set of one or more autonomous administrative areas for which a DMO has administrative authority is its DIT Domain. This is represented in Figure 5.

10.5.2 Specific administrative areas

In the same way that an Administrative Authority may operate in a specific role, entries in an administrative area may be considered in terms of a specific administrative function. When viewed in this context, an administrative area is termed a *specific administrative area*. There are four kinds of specific administrative area:

- subschema administrative areas;
- access control administrative areas;
- collective-attribute administrative areas;
- context default administrative areas.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

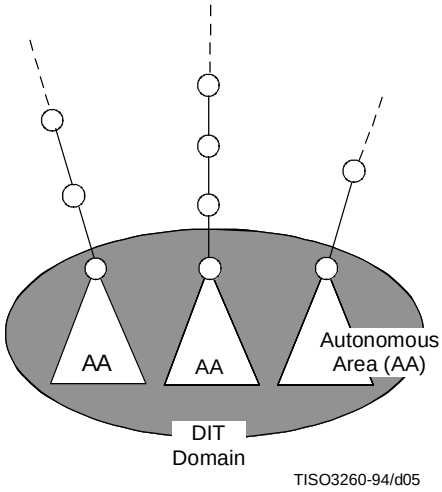


Figure 5 – A DIT Domain

An autonomous administrative area may be considered as implicitly defining a single specific administrative area for each specific aspect of administration. In this case, there is a precise correspondence between each such specific administrative area and the autonomous administrative area.

Alternatively, for each specific aspect of administration, the autonomous administrative area may be partitioned into non-overlapping specific administrative areas.

If so partitioned for a particular aspect of administration, each entry of the autonomous administrative area is contained in one and only one specific administrative area of that aspect.

A specific Administrative Authority is responsible for each specific administrative area. If, for a particular administrative aspect, an autonomous administrative area is not partitioned, a specific Administrative Authority is responsible for that administrative aspect for the entire autonomous administrative area.

10.5.3 Inner administrative areas

For the purpose of security or collective attribute administration, *inner (administrative) areas* within these kinds of specific administrative areas may be defined:

a) to represent a limited form of delegation; or

b) for administrative or operational convenience (e.g. where the administrative point of a subtree is in a DSA other than the one holding the entries within the subtree, that subtree may be designated as an inner area to allow administration via the local DSA).

An inner administrative area may be nested within another inner administrative area.

Inner areas represent areas of limited autonomy. Entries in inner areas are administered by the specific Administrative Authorities of the specific administrative areas within which they are contained, and also by the Administrative Authorities of the inner areas within which they are contained. The former authorities have overall control of the policies regulating these entries while the latter authorities have (limited) control over those aspects of policy delegated to them by the former.

The rules for nested inner areas, should they be permitted, must be defined as part of the definition of the specific administrative aspect within which they are contained.

10.5.4 Administrative points

The specification of the extent of an autonomous administrative area is implicit and consists of the identification of a point in the DIT (the root of the autonomous administrative area's subtree), an *autonomous administrative point*, from which the administrative area proceeds downwards until another autonomous administrative point is encountered, at which another autonomous area begins.

NOTE 1 – The immediate subordinates of the root of the DIT are autonomous administrative points.

Where an autonomous administrative area *is not* partitioned for a specific aspect of administration, then the administrative area for that aspect coincides with the autonomous administrative area. In this case, the autonomous administrative point is also the *specific administrative point* for this aspect of administration.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Where an autonomous administrative area *is* partitioned for a specific aspect of administration, then the specification of the extent of each specific administrative area consists of the identification of the root of the specific administrative area's subtree, a *specific administrative point*, from which the specific administrative area proceeds downwards until another specific administrative point (of the same administrative aspect) is encountered, at which another specific administrative area begins.

Specific administrative areas are always bounded by the autonomous administrative area they partition.

A particular administrative point may be the root of an autonomous administrative area and may be the root of one or more specific administrative areas.

The specification of the extent of an inner administrative area (within a specific administrative area) consists of the identification of the root of the inner administrative area's subtree, an *inner administrative point*. An inner administrative area is bounded by the specific administrative area within which it is defined.

An administrative point corresponding to the root of an autonomous administrative area represents a DIT Domain (and DSA) boundary. That is, its immediate superior in the DIT must be under the administrative authority of another DMD.

NOTE 2 – This implies that a DMO cannot arbitrarily partition a DIT Domain into autonomous administrative areas.

An administrative point is represented in the Directory information model by an entry holding an **administrativeRole** attribute. The values of this attribute identify the type of administrative point. This attribute is defined in 13.3.

Clauses 20 through 22 describe how administrative areas are mapped onto DSAs and the DSA information model.

Figure 6 depicts an autonomous administrative area which has been partitioned into two specific administrative areas for a specific aspect of administration (e.g. access control). In one specific administrative area, a nested inner administrative area has been created (e.g. because the subtree is to be held in a different DSA from the remainder of the specific administrative area).

Figure 6 uses the abbreviations AAP (Autonomous Administrative Point), SAP (Specific Administrative Point) and IAP (Inner Administrative point).

10.5.5 Administrative entries

An entry located at an administrative point is an *administrative entry*. Administrative entries may have special entries, called *subentries*, as immediate subordinates. The administrative entry and its associated subentries are used to control the entries encompassed by the associated administrative area.

Where inner administrative areas are used, the scopes of these areas may overlap.

Therefore, for each specific aspect of administrative authority, a definition is required of the method of combination of administrative information when it is possible for entries to be included in more than one subtree or subtree refinement associated with an inner area defined for that aspect.

NOTE – It is not necessary for an administrative point to represent each specific aspect of administrative authority. For example, there might be an administrative point, subordinate to the root of the autonomous administrative area, which is used for access control purposes only.

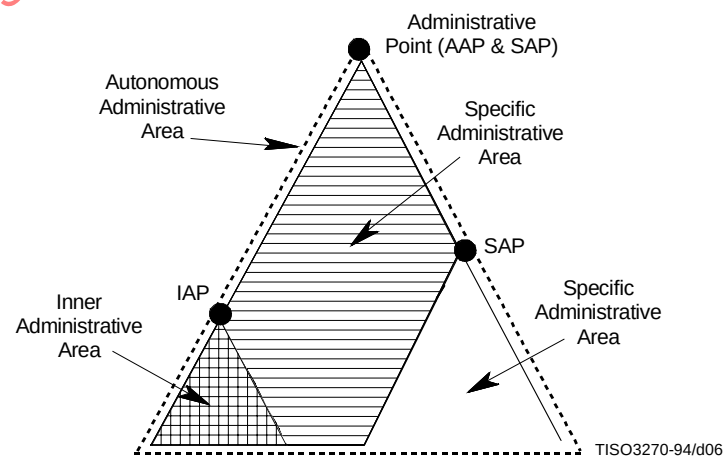


Figure 6 – Administrative Points and Areas

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

10.6 DIT Domain policies

A DIT Domain policy has the following components: DIT policy objects, DIT policy procedures, and DIT policy parameters.

An operational attribute that represents a DIT policy parameter is termed a *DIT policy attribute* (e.g. subschema administration operational attributes defined in clause 13 are DIT Domain policy attributes).

For a particular DSA, the possible values of a policy parameter may not correspond to distinct, realisable courses of action for that component. This may be the case, for example, when the DSA lacks the technical capability to perform all aspects of the policy procedure (e.g. implement a particular access control scheme). To be well-defined, a policy procedure must take such circumstances into account as part of its definition.

Specific DIT Domain policy objects and attributes are defined in clause 14 to support subschema administration.

10.7 DMD policies

A *DMD policy* is a policy that pertains to the operation of one or more of the DSAs in the DMD. A DMD policy may apply either to all the DSAs in the DMD in a uniform manner, to a subset of the DSAs in the DMD, or it may apply to one specific DSA.

One sort of DMD policy is to restrict or otherwise control the Directory and DSA abstract service provided by one or more DSAs.

Examples of such restrictions are:

a) Limiting the basic service provided to Directory (i.e. non-administrative) users to interrogation operations only, in accordance with CCITT Recommendation F.500.

b) Limiting the service provided to users accessing the DSA indirectly, via chaining, including distinctions based on whether the user request traversed a trusted path.

c) Limitations on requests accepted from users accessing the DSA directly when chaining is required to DSAs in the DMD known to be subject to limitations of the kind indicated in the previous point.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

SECTION 5 – MODEL OF DIRECTORY ADMINISTRATIVE AND OPERATIONAL INFORMATION

11 Model of Directory Administrative and Operational Information

11.1 Definitions

For the purposes of this Directory Specification, the following definitions apply:

11.1.2 base: The root vertex of the subtree or subtree refinement produced by the evaluation of a subtree specification.

11.1.3 chop: A set of assertions concerning the names of the subordinates of a base.

11.1.4 directory operational attribute: An operational attribute defined and visible in the Directory Administrative and Operational information model.

11.1.5 directory system schema: The set of rules and constraints concerning operational attributes and subentries.

11.1.6 entry: A Directory entry or extended Directory entry, depending on the context (either users and their applications or administration and operation of the Directory) in which the term is used.

11.1.7 subentry: A special sort of entry, known by the Directory, used to hold information associated with a subtree or subtree refinement.

11.1.8 subtree: A collection of object and alias entries situated at the vertices of a tree. The prefix "sub" emphasizes that the base (or root) vertex of this tree is usually subordinate to the root of the DIT.

11.1.9 subtree refinement: An explicitly specified subset of the entries in a subtree, where the entries are not located at the vertices of a single subtree.

11.1.10 subtree specification: The *explicit* specification of a subtree or subtree refinement. A subtree specification consists of zero or more of the specification elements base, chop and specification filter. The definition is termed explicit (in contrast to that of an administrative area) because the portion of the DIT subordinate to the base that is included in the subtree or subtree refinement is explicitly specified.

11.2 Overview

From an administrative perspective, user information held in the DIB is supplemented by administrative and operational information represented by:

- *operational attributes*, which represent information used to control the operation of the Directory (e.g. access control information) or used by the Directory to represent some aspect of its operation (e.g. time stamp information); and
- *subentries*, which associate the values of a set of attributes (e.g. collective attributes) with entries within the scope of the subentry. The scope of a subentry is a subtree or subtree refinement.

This information, illustrated in Figure 7, may be placed in the Directory by administrative authorities or by DSAs, and is used by the Directory in the course of its operation.

Two mechanisms in the Directory abstract service that relate to this view of Directory information are:

- **EntryInformationSelection** has been extended to permit the selection of operational attributes in an entry; and
- the **subentries** service control has been added to permit the list and search operations to apply either to object and alias entries or to subentries.

Access to operational information, as for user information, may be limited by an access control mechanism.

Entries are made visible to Directory users via the Directory abstract service, but their relationships to the DSAs that ultimately hold them are not. The DSA information model, described in clauses 21 through 24, expresses the mapping of these entries onto the information repositories of DSAs.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

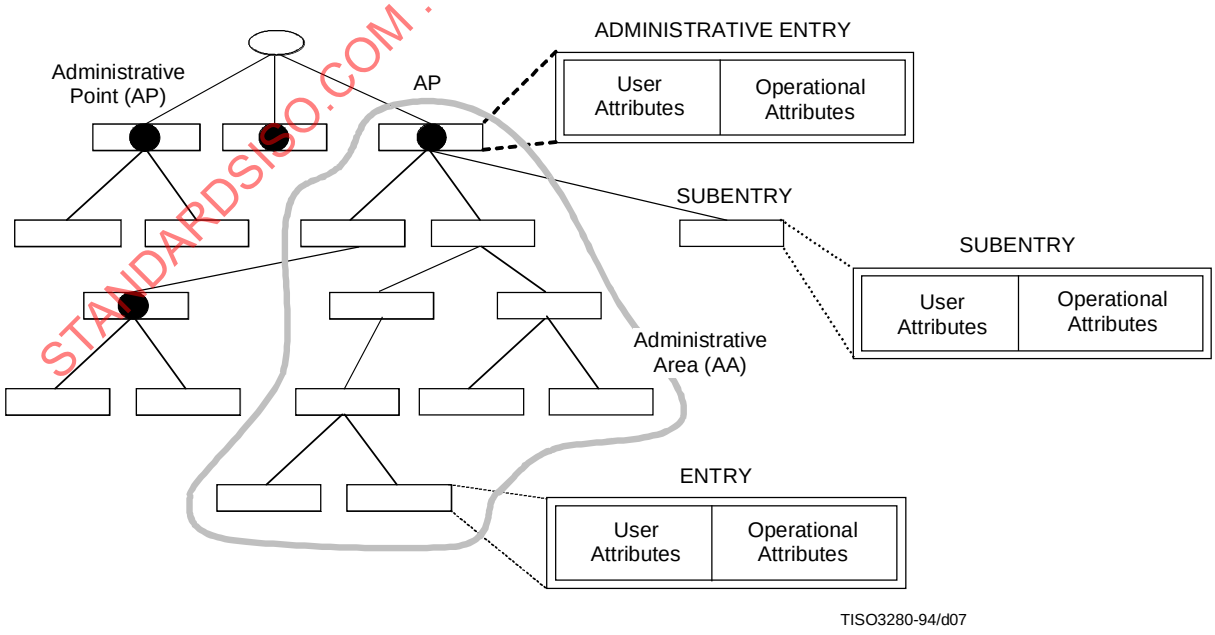


Figure 7 – Model of Directory Administrative and Operational Information

11.3 Subtrees

11.3.1 Overview

A subtree is a collection of object and alias entries situated at the vertices of a tree. Subtrees do not contain subentries. The prefix "sub", in subtree, emphasizes that the base (or root) vertex of this tree is usually subordinate to the root of the DIT.

A subtree begins at some vertex and extends to some identifiable lower boundary, possibly extending to leaves. A subtree is always defined within a context which implicitly bounds the subtree. For example, the vertex and lower boundaries of a subtree defining a replicated area are bounded by a naming context. Similarly, the scope of a subtree defining a specific administrative area is limited to the context of an enclosing autonomous administrative area.

11.3.2 Subtree specification

Subtree specification is the definition of a subset of the entries below a specified vertex which forms the base of the subtree or subtree refinement.

The vertex and/or the lower boundary of the subtree may be implicitly specified, in which case they are determined by the context within which the subtree is used.

The vertex and/or the lower boundary may be explicitly specified using the mechanism specified in this clause. This mechanism may also be used to specify subtree refinements which are not true tree structures.

NOTE – The topological concept of a (sub)tree is useful in considering such specifications, although a particular specification may determine a collection of entries that are *not* located at the vertices of a single (sub)tree. The term *subtree refinement* is preferred when the entries of the collection are not so located.

Specification of a subtree consists of three optional elements of specification which identify the base of the subtree, and then reduce the collection of subordinate entries. These elements of specification are:

a) *Base* – The root vertex of the subtree or subtree refinement produced by the evaluation of a subtree specification;

b) *Chop* – A set of assertions concerning the names of the subordinate entries; and

c) *Specification filter* – A proper subset of the assertive capability of a filter applied to the subordinates.

The specification of a subtree or subtree refinement may be represented by the following ASN.1 type:

```
SubtreeSpecification ::= SEQUENCE {
    base                [0]  LocalName DEFAULT { },
                             COMPONENTS OF  ChopSpecification,
    specificationFilter [4]  Refinement OPTIONAL }
    -- empty sequence specifies whole administrative area
```

The three components of this sequence correspond to the three specification elements identified above.

Where a value of **SubtreeSpecification** identifies a collection of entries that are located at the vertices of a single subtree, the collection is termed a subtree, otherwise the collection is termed a subtree refinement.

The **SubtreeSpecification** type provides a general purpose mechanism for the specification of subtrees and subtree refinements. Any particular use of this mechanism defines the specific semantics of precisely what is specified and may impose limitations or constraints on the components of **SubtreeSpecification**.

When each of the components of **SubtreeSpecification** is absent (i.e. a value of type **SubtreeSpecification** which is an empty sequence, **{}**), the subtree so specified is implicitly determined by the context within which the **SubtreeSpecification** is used.

These terms are illustrated in Figure 8, for the case where subtrees are deployed within the context of administrative areas.

11.3.3 Base

The **base** component of **SubtreeSpecification** represents the root vertex of the subtree or subtree refinement. This may be an entry which is subordinate to the root vertex of the identified scope or may be the root vertex of the identified scope itself (the default).

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

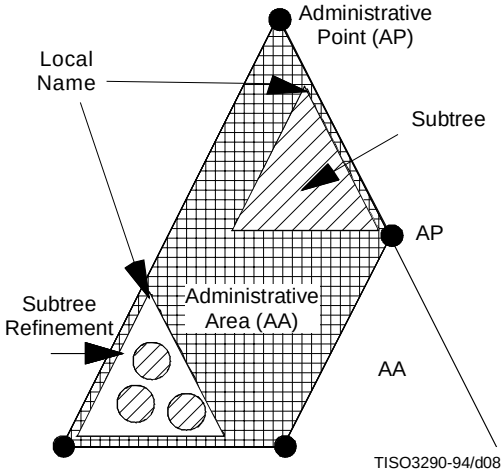


Figure 8 – Specification of Subtrees and Subtree Refinements within the context of Administrative Areas

The relative name of the root vertex of the subtree with respect to the root vertex of the identified scope is a value of type **LocalName**:

LocalName ::= RDNSequence

Note that the root vertex of the identified scope and the root vertex of the subtree coincide when **LocalName** is omitted from **SubtreeSpecification**.

RDNs used to name the root vertex of the subtree shall be primary RDNs.

11.3.4 Chop Specification

The **ChopSpecification** component consists of a set of assertions concerning the names of the subordinates of a base. It consists of a value of type **ChopSpecification**:

```
ChopSpecification ::= SEQUENCE {
        specificExclusions      [1]   SET OF CHOICE {
                chopBefore              [0]   LocalName,
                chopAfter               [1]   LocalName } OPTIONAL,
        minimum                 [2]   BaseDistance DEFAULT 0,
        maximum                 [3]   BaseDistance OPTIONAL }
```

This type is intended to permit the specification of a tree structure (or subset thereof) starting at the base by two methods, specific exclusions and base distance.

Where any attribute in an RDN in **chopBefore** or **chopAfter** has multiple distinguished values differentiated by context, the primary distinguished value shall be used as the **value** in the RDN in **LocalName**.

11.3.4.1 Specific Exclusions

The **specificExclusions** component has two forms, **chopBefore** and **chopAfter,** which may be used individually or in combination.

The **chopBefore** component defines a list of exclusions, each in terms of some limit point which is to be excluded, along with its subordinates, from the subtree or subtree refinement. The limit points are the entries identified by a **LocalName**, relative to the base.

The **chopAfter** component defines a list of exclusions, each in terms of some limit point whose subordinates are to be excluded from the subtree or subtree refinement. The limit points are the entries identified by a **LocalName**, relative to the base.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

11.3.4.2 Minimum and Maximum

These components allow exclusion of all entries that are superior to entries that are **minimum** RDN arcs below the base as well as entries which are subordinate to entries that are **maximum** RDN arcs below the base. These distances are expressed by values of the type **BaseDistance**:

```
BaseDistance ::= INTEGER (0 .. MAX)
```

A value of **minimum** equal to zero (the default), corresponds to the base. An absent **maximum** component indicates that no lower limit should be imposed on the subtree or subtree refinement.

11.3.5 Specification Filter

The **specificationFilter** component consists of a proper subset of the assertive capability of a filter (see ITU-T Rec. X.511 | ISO/IEC 9594-3) applied to the subordinates of a base. Only entries for which the filter evaluates to true are included in the resulting subtree refinement. It consists of a value of type **Refinement**:

```
Refinement ::= CHOICE {
        item        [0]    OBJECT-CLASS.&id,
        and         [1]    SET OF Refinement,
        or          [2]    SET OF Refinement,
        not         [3]    Refinement }
```

A **Refinement** evaluates to TRUE as if it were a filter making an **equality** assertion regarding values of the attribute type **objectClass** only.

11.4 Operational attributes

There are three varieties of operational attribute: Directory operational attributes, DSA-shared operational attributes, and DSA-specific operational attributes.

Directory operational attributes occur in the Directory information model and are used to represent control information (e.g. access control information) or other information provided by the Directory (e.g. an indication of whether an entry is a leaf or non-leaf entry).

DSA-shared operational attributes occur only in the DSA Information Model, and are not visible at all in the Directory Information Models.

DSA-specific operational attributes occur only in the DSA Information Model, and are not visible at all in the Directory Information Models.

NOTE – These are described in clauses 19 through 20.

The definition and use of each operational attribute is a matter for specification in the appropriate Directory Specification.

11.5 Entries

11.5.1 Overview

From an administrative perspective, user information held in an entry may be supplemented by administrative and operational information represented by operational attributes.

The Directory uses the object class attribute and DIT content rules applicable to an entry to control the user attributes required and permitted in the entry. The operational attributes of an entry are governed by the Directory system schema (see clause 13) applicable to the entry.

11.5.2 Access to operational attributes

Although not normally visible, the directory operational attributes within entries may be made visible to authorized (e.g. administrative) users of the directory abstract service. Certain operational attributes (e.g. **entryACI**, or **modifyTimestamp**) might also be available to ordinary users.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

11.6 Subentries

11.6.1 Overview

A *subentry* is a special kind of entry immediately subordinate to an administrative point. It contains attributes that pertain to a subtree (or subtree refinement) associated with its administrative point. The subentries and their administrative point are part of the same naming context (see clause 17).

A single subentry may serve all or several aspects of administrative authority. Alternatively, a specific aspect of administrative authority may be handled through one or more of its own subentries. At most one subentry is permitted for a subschema administrative authority. Access control and collective attribute authorities may have several subentries.

A subentry is not considered in **list** and **search** operations unless the **subentries** service control is included in the request.

A subentry shall not have subordinates.

The structure of a subentry corresponding to an administrative point is depicted in Figure 9.

A subentry consists of:

- a **commonName** attribute, specified in ITU-T Rec. X.520 | ISO/IEC 9594-6 which contains the RDN of the subentry;
- a **subtreeSpecification** attribute, specified in clause 13;
- an **objectClass** attribute, specified in clause 12, which indicates the purpose(s) of the subentry in the operation of the Directory;
- other attributes, depending on the values of the **objectClass** attribute.

Subentries may also contain operational attributes with appropriate semantics (see 11.6.4).

11.6.2 Subentry RDN attribute

The **commonName** attribute used as the subtree identifier serves to distinguish the various subentries that may be defined as immediate subordinates of a specific administrative entry.

NOTE – The value of this attribute might be selected to serve as a mnemonic to representatives of the Administrative Authority.

The **commonName** attribute for a subentry may not contain multiple distinguished values differentiated by context; only a single distinguished value is permitted.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

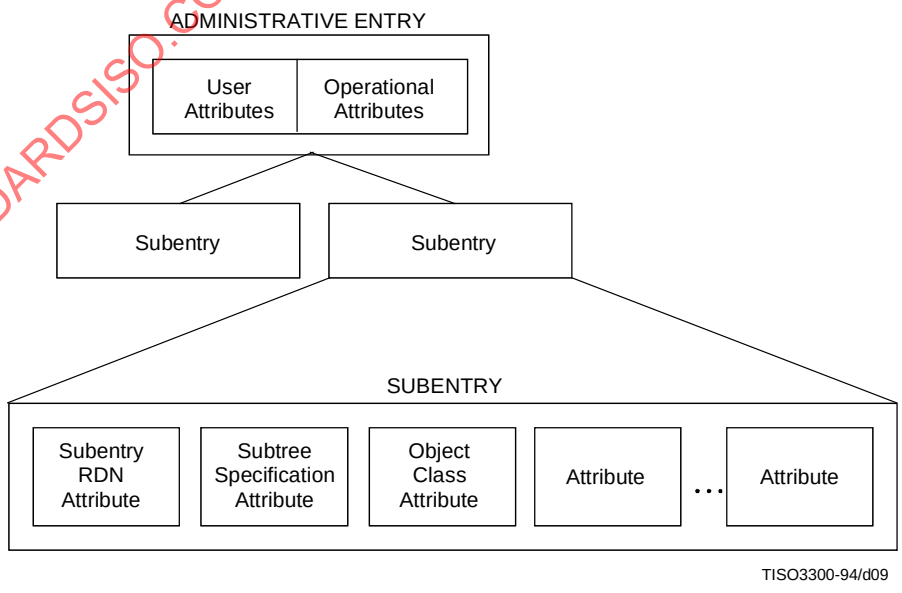


Figure 9 – Structure of a Subentry

11.6.3 Subtree Specification attribute

The **subtreeSpecification** attribute defines the collection of entries within the administrative area with which the subtree is concerned.

11.6.4 Use of the Object Class attribute

The content of a subentry is regulated by the values of the subentry's **objectClass** attribute.

The **objectClass** attribute of all subentries shall contain the value **subentry**. The **subentry** object class is a structural object class, defined in clause 13, used to include the **commonName**, **subtreeSpecification** and **objectClass** attributes in all subentries.

In order to regulate the remaining attributes, the other values of the **objectClass** attribute, representing the auxiliary object classes allowed for the subentry, shall be used.

The definition of the semantics of one of these values includes an identification and specification of zero or more attribute types that must or may appear in the subentry when the **objectClass** attribute assumes the value. The definition of the semantics of a value for the **objectClass** attribute shall include:

- an indication of whether an entry may be included in more than one subtree or subtree refinement associated with the particular purpose (e.g. it may not be permitted in the case of **subschema,** but may be permitted for access control); and if so
- the effects of the combination of associated subentry attributes, if any.

A subentry of a particular object class may only be subordinate to an administrative entry if the **administrativeRole** attribute permits that class of subentry as a subordinate.

As for object and alias entries, information held in a subentry may be supplemented by administrative and operational information represented by operational attributes. For example, a subentry is permitted to contain entry ACI, provided only that this ACI is permitted by and consistent with the value of the **accessControlScheme** attribute of the corresponding access control specific point. Similarly, a subentry may contain a **modifyTimestamp**.

11.6.5 Other subentry attributes

The remaining attributes within a subentry depend on the values of the **objectClass** attribute. For example, a subschema attribute may only be placed in a subentry if its **objectClass** attribute has **subschema** as one of its values.

11.7 Information model for collective attributes

An autonomous administrative area may be designated as a collective attribute specific administrative area in order to deploy and administer collective attributes. This shall be indicated by the presence of the value **id-ar-collectiveAttributeSpecificArea** in the associated administrative entry's **administrativeRole** attribute (in addition to the presence of the value **autonomousArea**, and possibly other values).

Such an autonomous administrative area may be partitioned in order to deploy and administer collective attributes in the specific partitions. In this case, the administrative entries for each of the collective attribute specific administrative areas are indicated by the presence of the value **id-at-collectiveAttributeSpecificArea** in these entries' **administrativeRole** attributes.

If such an autonomous administrative area is not partitioned, there is a single specific administrative area for collective attributes encompassing the entire autonomous administrative area.

Additionally, a specific administrative area defined for the purpose of collective attribute administration may be further divided into nested inner areas for the same purpose. The **administrativeRole** attribute of the administrative entries for each such inner administrative area shall indicate this by the presence of the value **id-ar-collectiveAttributeInnerArea.**

An entry collection and its associated collective attributes are represented in the Directory information model by a subentry, termed a *collective attribute subentry*, whose **objectClass** attribute has the value **id-sc-collectiveAttributeSubentry**, as defined in clause 13. A subentry of this class may be the immediate subordinate of an administrative entry whose **administrativeRole** attribute contains the value **id-ar-collectiveAttributeSpecificArea** or **id-ar-collectiveAttributeInnerArea**.

Where there are different entry collections within a given collective attribute area, each must have its own subentry.

The entry collection itself is defined by the value of the **subtreeSpecification** operational attribute of the subentry. This value defines the *scope* of the collective attribute subentry. The user attributes of the subentry are the collective attributes of the entry collection.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

NOTE 1 – Because subtree refinement is based on object class, the association of collective attributes with object entries can be done in a manner that naturally extends the schema for these entries. For example, the **organizationalPerson** entries of an organization might be extended with a set of collective attributes appropriate for all persons affiliated with the organization by the creation of a subentry whose associated subtree is refined to include only **organizationalPerson** entries and which contains the organization's set of collective attributes. Additionally, a DIT Content Rule for such entries would need to be defined to allow collective attributes to become visible in the entries.

Collective attribute types and non-collective attribute types differ semantically. An attribute type capable of expressing collective semantics must be designated as a collective attribute type at the time of its definition.

NOTE 2 – Merging procedures employed by the Directory in the case of independent sources of values of a collective attribute type are described in ITU-T Rec. X.511 | ISO/IEC 9594-3.

Collective attributes may be excluded from appearing in a particular entry through use of the **collectiveExclusions** attribute defined in clause 13.

11.8 Information model for context defaults

An autonomous administrative area may be designated as a context default specific administrative area in order to deploy and administer context defaults. This shall be indicated by the presence of the value **id-ar-contextDefaultSpecificArea** in the associated administrative entry's **administrativeRole** attribute (in addition to the presence of the value **id-ar-autonomousArea**, and possibly other values).

Such an autonomous administrative area may be partitioned in order to deploy and administer context defaults in the specific partitions. In this case, the administrative entries for each of the context default specific areas are indicated by the presence of the value **id-ar-contextDefaultSpecificArea** in these entries' **administrativeRole** attribute.

If an autonomous administrative area is not partitioned, there is a single specific administrative area for context defaults encompassing the entire autonomous administrative area.

Context defaults are represented in the Directory Information model by a subentry, termed a *context default subentry*, whose **objectClass** attribute has the value **id-sc-contextAssertionSubentry** as defined in 13.7. A subentry of this class may be the immediate subordinate of an administrative entry whose **administrativeRole** attribute contains the value **id-ar-contextDefaultSpecificArea**.

The context default subentry defines a set of context assertions, any one of which is applied whenever there is no context assertion applicable to a given attribute type specified by the user when accessing the portion of the DIT defined by the **subtreeSpecification** operational attribute of the subentry. Application of default context assertions is described in 8.8.2.2, and in 7.6.1 of ITU-T Rec. X.511 | ISO/IEC 9594-3.

SECTION 6 – THE DIRECTORY SCHEMA

12 Directory Schema

12.1 Definitions

For the purposes of this Directory Specification, the following definitions apply:

12.1.1 attribute syntax: The ASN.1 data type used to represent values of an attribute.

12.1.2 directory schema: The set of rules and constraints concerning DIT structure, DIT content, DIT context use, object classes, and attribute types, syntaxes and matching rules which characterize the DIB. The Directory Schema is manifested as a set of non-overlapping subschemas each governing entries of an autonomous administrative area (or a subschema specific partition thereof). The Directory schema is concerned only with Directory User Information.

12.1.3 (directory) subschema: The set of rules and constraints concerning DIT structure, DIT content, object classes and attribute types, syntaxes and matching rules which characterize the DIB entries within an autonomous administrative area (or a subschema specific partition thereof).

12.1.4 DIT content rule: A rule governing the content of entries of a particular structural object class. It specifies the auxiliary object classes and additional attribute types permitted to appear, or excluded from appearing, in entries of the indicated structural object class.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

12.1.5 DIT context use: A rule governing the context types that may be associated with attribute values of particular attribute types. It specifies the permitted and the mandatory context types for the attribute type.

12.1.6 DIT structure rule: A rule governing the structure of the DIT by specifying a permitted superior to subordinate entry relationship. A structure rule relates a name form, and therefore a structural object class, to superior structure rules. This permits entries of the structural object class identified by the name form to exist in the DIT as subordinates to entries governed by the indicated superior structure rules.

12.1.7 governing structure rule (of an entry): With respect to a particular entry, the *single* DIT structure rule that applies to the entry. This rule is indicated by the **governingStructureRule** operational attribute.

12.1.8 name form: A name form specifies a permissible RDN for entries of a particular structural object class. A name form identifies a named object class and one or more attribute types to be used for naming (i.e. for the RDN). Name forms are primitive pieces of specification used in the definition of DIT structure rules.

NOTE – Name forms are registered and have global scope. DIT structure rules are not registered and have the scope of the administrative area with which they are associated.

12.1.9 superior structure rule: With respect to a particular entry, the DIT structure rule governing the entry's superior.

12.2 Overview

The Directory Schema is a set of definitions and constraints concerning the structure of the DIT, the possible ways entries are named, the information that can be held in an entry, the attributes used to represent that information and their organization into hierarchies to facilitate search and retrieval of the information and the ways in which values of attributes may be matched in attribute value and matching rule assertions.

NOTE 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).

Formally, the Directory Schema comprises a set of:

a) *Name Form* definitions that define primitive naming relations for structural object classes;

b) *DIT Structure Rule* definitions that define the names that entries may have and the ways in which the entries may be related to one another in the DIT;

c) *DIT Content Rule* definitions that extend the specification of allowable attributes for entries beyond those indicated by the structural object classes of the entries;

d) *Object Class* definitions that define the basic set of mandatory and optional attributes that shall be present, and may be present, respectively, in an entry of a given class, and which indicate the kind of object class that is being defined (see 7.3);

e) *Attribute Type* definitions that identify the object identifier by which an attribute is known, its syntax, associated matching rules, whether it is an operational attribute and if so its type, whether it is a collective attribute, whether it is permitted to have multiple values and whether or not it is derived from another attribute type;

f) *Matching Rule* definitions that define matching rules;

g) *DIT Context Use* definitions that govern the context types that may be associated with attribute values of any particular attribute type.

Figure 10 illustrates the relationships between schema and subschema definitions on the one side, and the DIT, directory entries, attributes, and attribute values on the other.

Figure 10 is interpreted as follows:

- the items listed vertically on the left represent elements of schema;
- the items listed vertically on the right represent instances of corresponding schema items, instantiated according to the rules defined by these schema items;
- the relationship between items of schema is illustrated by the "uses" relationship;
- the relationship between instances of different aspects of schema is illustrated using the "belong to" relationship.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

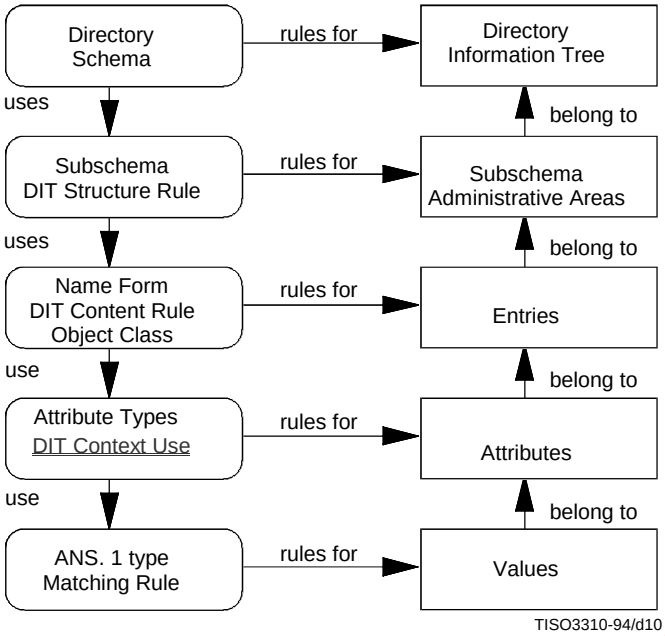


Figure 10 – Overview of Directory

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The Directory Schema is distributed, like the DIB itself. It is manifested as a set of non-overlapping subschemas each governing entries of an autonomous administrative area (or a subschema specific partition thereof). A subschema administrative authority establishes the rules and constraints constituting the subschema.

The subschema administrative authority may elect to use individual elements of the Directory Schema having global scope which are defined in these Directory Specifications: name forms, object classes and attributes (types and matching rules). It may also choose to define alternatives to these elements more appropriate to its own environment or it may choose some intermediate approach, using both standardised and proprietary schema elements.

The subschema administrative authority defines those schema elements whose scope is limited to the subschema: DIT structure rules, DIT content rules, and DIT context use. In addition, the subschema administrative authority may also specify which matching rules are applicable to which attribute types.

The Directory Schema is concerned only with directory user information. Although some support for the specification of operational information is provided in the notation defined in this clause, the regulation of Directory Administrative and Operational Information is the concern of the *Directory System Schema*.

NOTE 2 – The Directory System Schema is described in clause 13.

12.3 Object class definition

The definition of an object class involves:

a) indicating which classes this object class is to be a subclass of;

b) indicating what kind of object class is being defined;

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;

e) assigning an object identifier for the object class.

NOTE – Collective attributes shall not appear in the attribute types of an object class definition.

12.3.1 Subclassing

There are restrictions on subclassing, namely:

– only abstract object classes shall be superclasses of other abstract object classes.

There is one special object class, of which every structural object class is a subclass. This object class is called **top**. **top** is an abstract object class.

12.3.2 The object class attribute

Every entry shall contain an attribute of type **objectClass** to identify the object classes and superclasses to which the entry belongs. The definition of this attribute is given in 12.4.6. This attribute is multi-valued.

There shall be one value of the **objectClass** attribute for the entry's structural object class and a value for each of its superclasses. **top** may be omitted.

An entry's structural object classes shall not be changed. The initial values of the **objectClass** attribute are provided by the user when the entry is created.

Where auxiliary object classes are used, an entry may contain values of the **objectClass** attribute for the auxiliary object classes and their superclasses allowed by a DIT content rule. If a value for an allowed auxiliary object class is present, then values for the superclasses of the auxiliary object class shall also be present.

Where the **objectClass** attribute contains an object identifier value for an auxiliary object class, then the entry shall contain the mandatory attributes indicated by that object class.

NOTE 1 – The requirement that the **objectClass** attribute be present in every entry is reflected in the definition of **top**.

NOTE 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 **objectClass** attribute (and any value in the chain may be matched by a filter).

NOTE 3 – Access Control restrictions may be placed on modification of the **objectClass** attribute.

In conjunction with the applicable DIT content rules, 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 that is not explicitly allowed by the entry's DIT content rule shall fail.

NOTE 4 – In particular, the Directory will ordinarily prevent:

a) attribute types absent from an entry's structural object class definition and not permitted by the entry's DIT content rule being added to an entry of that object class;

b) an entry being created with one or more absent mandatory attribute types for an object class of the entry;

c) a mandatory attribute type for the object class of the entry being deleted.

12.3.3 Object class specification

Object classes may be defined as values of the **OBJECT-CLASS** information object class:

```
OBJECT-CLASS  ::=    CLASS {
      &Superclasses          OBJECT-CLASS OPTIONAL,
      &kind                  ObjectClassKind DEFAULT structural,
      &MandatoryAttributes   ATTRIBUTE OPTIONAL,
      &OptionalAttributes    ATTRIBUTE OPTIONAL,
      &id                    OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
      [ SUBCLASS OF          &Superclasses ]
      [ KIND                 &kind ]
      [ MUST CONTAIN         &MandatoryAttributes ]
      [ MAY CONTAIN          &OptionalAttributes ]
      ID                     &id }

ObjectClassKind  ::=  ENUMERATED {
      abstract     (0),
      structural   (1),
      auxiliary    (2) }
```

For an object class which is defined using this information object class:

a) **&Superclasses** is the set of object classes which are its direct superclasses;

b) **&kind** is its kind;

c) **&MandatoryAttributes** is the set of attributes which entries of that class must contain;

d) **&OptionalAttributes** is the set of attributes which entries of that class may contain, except that if an attribute appears in both the mandatory and optional sets, it shall be considered mandatory;

e) **&id** is the object identifier assigned to it.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The object classes previously mentioned (**top** and **alias**) are defined below:

```
top  OBJECT-CLASS  ::= {
       KIND              abstract
       MUST CONTAIN      { objectClass }
       ID                id-oc-top }

alias  OBJECT-CLASS  ::= {
       SUBCLASS OF       { top }
       MUST CONTAIN      { aliasedEntryName }
       ID                id-oc-alias }
```

NOTE – 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.

12.4 Attribute type definition

The definition of an attribute type involves:

a) optionally indicating that the attribute type is a subtype of a previously defined attribute type, its direct supertype;

b) specifying the attribute syntax for the attribute type;

c) optionally indicating the equality, ordering and/or substring matching rule(s) for the attribute type;

d) indicating whether an attribute of this type shall have only one or may have more than one value;

e) indicating whether the attribute type is operational or user;

f) optionally indicating that a user attribute type is collective;

g) optionally indicating that an operational attribute is not user modifiable;

h) for operational attributes, indicating the application;

i) assigning an object identifier to the attribute type.

12.4.1 Operational attributes

Some operational attributes are under direct user control. In other cases the operational attribute's values are controlled by the Directory. In the latter case, the definition of the operational attribute shall indicate that no user modifications to the attribute values are permitted.

The specification of an operational attribute type shall indicate its application, which shall be one of the following:

– Directory operational attribute (e.g. access control attributes);

– DSA-shared operational attribute (e.g. a master-access-point attribute);

– DSA-specific operational attribute (e.g. a copy-status attribute).

12.4.2 Attribute hierarchies

An attribute hierarchy shall contain either user attributes or operational attributes but not both. It follows that a user attribute shall not be derived from an operational attribute and that an operational attribute shall not be derived from a user attribute.

An operational attribute that is a subtype of another operational attribute shall have the same application as its supertype.

If an attribute type is not a subtype of another attribute type, the attribute syntax and matching rules (if applicable) shall be specified in the attribute type definition. Specifying an attribute syntax shall be done by directly specifying the ASN.1 data type.

If an attribute type is a subtype of an indicated type, the definition need not specify an attribute syntax, in which case its attribute syntax is that of its direct supertype. If the attribute syntax is indicated and the attribute has a direct supertype, the indicated syntax must be compatible with the supertype's syntax, i.e. every possible value satisfying the attribute's syntax must also satisfy the supertype's syntax.

If an attribute type is a subtype of another attribute type, the matching rules applicable to the supertype are applicable to the subtype, unless extended or modified in the definition of the subtype. A matching rule defined for a supertype may not be removed when defining a subtype.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

12.4.3 Collective attributes

An operational attribute shall not be defined to be collective.

A user attribute may be defined to be collective. This indicates that the same attribute values will appear in the entries of an entry collection subject to the use of the **collectiveExclusions** attribute.

Collective attributes shall be multi-valued.

12.4.4 Attribute syntax

If an equality matching rule is specified for the attribute type, the Directory shall ensure that the correct attribute syntax is used for every value of this attribute type.

12.4.5 Matching rules

Equality, ordering and substrings matching rules may be indicated in the attribute type definition. The same matching rule may be used for one or more of these types of matches if the semantics of the rule allows for more than one of these different types of matches.

NOTE 1 – This fact must be reflected in the definition of the indicated matching rule.

If no equality matching rule is indicated, the Directory:

a) treats values of this attribute as having type **ANY**, i.e. the Directory may not check that those values conform with the data type or any other rule indicated for the attribute;

b) does not permit the attribute to be used for naming;

c) does not allow individual values of multi-valued attributes to be added or removed;

d) does not perform comparisons of values of the attribute;

e) will not attempt to evaluate **AVAs** using values of such an attribute type.

If an equality matching rule is indicated, the Directory:

a) treats values of this attribute as having the type defined in the **&Type** field in the attribute's definition (or that of the attribute from which the attribute is derived);

b) will use the indicated equality matching rule for the purpose of evaluating attribute value assertions concerning the attribute;

c) will only match a presented value of a suitable data type as specified in the attribute type definition.

NOTE 2 – This subclause applies equally to an attribute whose equality matching rule uses an assertion syntax different from the syntax of the attribute type.

If no ordering matching rule is indicated, the Directory shall treat any assertion of an ordering match using the syntax provided by the Directory Abstract Service as undefined.

If no substrings matching rule is indicated, the Directory shall treat any assertion of a substring match using the syntax provided by the Directory Abstract Service as undefined.

An attribute type shall only specify matching rules whose definition apply to the attribute's attribute syntax.

12.4.6 Attribute definition

Attributes may be defined as values of the **ATTRIBUTE** information object class:

```
ATTRIBUTE ::= CLASS {
    &derivation                 ATTRIBUTE OPTIONAL,
    &Type                       OPTIONAL,       -- either &Type or &derivation required --
    &equality-match             MATCHING-RULE OPTIONAL,
    &ordering-match             MATCHING-RULE OPTIONAL,
    &substrings-match           MATCHING-RULE OPTIONAL,
    &single-valued              BOOLEAN DEFAULT FALSE,
    &collective                 BOOLEAN DEFAULT FALSE,
    -- operational extensions --
    &no-user-modification       BOOLEAN DEFAULT FALSE,
    &usage                      AttributeUsage DEFAULT userApplications,
    &id                         OBJECT IDENTIFIER UNIQUE }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
WITH SYNTAX {
        [ SUBTYPE OF                    &derivation ]
        [ WITH SYNTAX                   &Type ]
        [ EQUALITY MATCHING RULE        &equality-match ]
        [ ORDERING MATCHING RULE        &ordering-match ]
        [ SUBSTRINGS MATCHING RULE      &substrings-match ]
        [ SINGLE VALUE                  &single-valued ]
        [ COLLECTIVE                    &collective ]
        [ NO USER MODIFICATION          &no-user-modification ]
        [ USAGE                         &usage ]
        ID                              &id }

AttributeUsage  ::=  ENUMERATED {
        userApplications        (0),
        directoryOperation      (1),
        distributedOperation    (2),
        dSAOperation            (3) }
```

For an attribute which is defined using this information object class:

a) **&derivation** is the attribute, if any, of which it is a subtype;

b) **&Type** is its attribute syntax. This shall be an ASN.1 type, but not a type that contains an **EmbeddedPDV**;

c) **&equality-match** is its equality matching rule (if any);

d) **&ordering-match** is its ordering matching rule (if any);

e) **&substrings-match** is its substrings matching rule (if any);

f) **&single-valued** is TRUE if it is single valued, and false otherwise;

g) **&collective** is TRUE if it is a collective attribute, and false otherwise;

h) **&no-user-modification** is TRUE if it is an operational attribute which cannot be modified by the user;

i) **&usage** indicates the operational usage of the attribute. **userApplications** means it is a user attribute, **directoryOperation**, **distributedOperation**, and **dSAOperation** mean it is a directory, distributed, or DSA-operational attribute respectively;

j) **&id** is the object identifier assigned to it.

The attribute types defined in the 1988 edition of this Directory Specification which are known to and used by the Directory for its own purposes are defined as follows:

```
objectClass  ATTRIBUTE  ::=  {
        WITH SYNTAX                     OBJECT IDENTIFIER
        EQUALITY MATCHING RULE          objectIdentifierMatch
        ID                              id-at-objectClass }

aliasedEntryName  ATTRIBUTE ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        SINGLE VALUE                    TRUE
        ID                              id-at-aliasedEntryName }
```

NOTE – The matching rules referred to in these definitions are themselves defined in 12.5.2.

The **objectClass** and **aliasedEntryName** attributes are defined as user attributes even though they are used for Directory operations and semantically should be defined as operational. This is because these attributes were defined as user attributes before the operational attribute concept and must remain as user attributes to facilitate interworking between systems implementing different editions of this Directory Specification.

12.5 Matching rule definition

12.5.1 Overview

The definition of a matching rule involves:

a) defining the syntax of an assertion of the matching rule;

b) specifying the different types of matches supported by the rule;

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

c) defining the appropriate rules for evaluating a presented assertion with respect to target attribute values held in the DIB;

d) assigning an object identifier to the matching rule.

A matching rule shall be used to evaluate attribute value assertions of attributes indicating the rule as their equality matching rule. The syntax used in the attribute value assertion (i.e. the **assertion** component of the attribute value assertion) is the matching rule's assertion syntax.

A matching rule may apply to many different types of attributes with different attribute syntaxes.

The definition of a matching rule shall include a specification of the syntax of an assertion of the matching rule and the way in which values of this syntax are used to perform a match. This does not require a full specification of the attribute syntax to which the matching rule may apply. A definition of a matching rule for use with attributes with different ASN.1 syntaxes shall specify how matches shall be performed.

The applicability of defined matching rules to the attributes contained in a subschema specification (over and above the matching rules used in the definition of these attribute types) is indicated through the subschema specification operational attribute **matchingRuleUse**, defined in 14.7.7.

12.5.2 Matching rule definition

Matching rules may be defined as values of the **MATCHING-RULE** information object class:

```
MATCHING-RULE ::= CLASS {
        &AssertionType  OPTIONAL,
        &id             OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
        [ SYNTAX        &AssertionType ]
        ID              &id }
```

For a matching rule which is defined using this information object class:

a) **&AssertionType** is the syntax for an assertion using this matching rule; if it is omitted, the assertion syntax is the same syntax as that of the attribute the rule is applied to;

b) **&id** is the object identifier assigned to it.

The **objectIdentifierMatch** matching rule is defined as follows:

```
objectIdentifierMatch  MATCHING-RULE ::= {
        SYNTAX      OBJECT IDENTIFIER
        ID          id-mr-objectIdentifierMatch }
```

A presented value of type object identifier matches a target value of type object identifier if and only if they both have the same number of integral components and each integral component of the first is equal to the corresponding component of the second. This matching rule is inherent in the definition of the ASN.1 type object identifier. **objectIdentifierMatch** is an equality matching rule.

The **distinguishedNameMatch** is defined as follows:

```
distinguishedNameMatch  MATCHING-RULE  ::= {
        SYNTAX      DistinguishedName
        ID          id-mr-distinguishedNameMatch }
```

A presented distinguished name value matches 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 **AttributeTypeAndValue**;

c) corresponding **AttributeTypeAndValue** (i.e. those in corresponding RDNs and with identical attribute types) have attribute values which match as described in 9.4.

distinguishedNameMatch is an equality matching rule.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

12.6 DIT structure definition

12.6.1 Overview

A fundamental aspect of the Directory schema is the specification of where an entry of a particular class may be placed in the DIT and how it should be named, considering:

– the hierarchical relationship of entries in the DIT (DIT structure rules);

– the attribute or attributes used to form the RDN of the entry (name forms).

12.6.2 Name form definition

The definition of a name form involves:

a) specifying the named object class;

b) indicating the mandatory attributes to be used for the RDNs for entries of this object class where this name form applies;

c) indicating the optional attributes if any that may be used for the RDNs for entries of this object class where this name form applies;

d) assigning an object identifier for the name form.

If different sets of naming attributes are required for entries of a given structural object class, then a name form must be specified for each distinct set of attributes to be used for naming.

Only structural object classes are used in name forms.

For entries of a particular structural object class to exist in a portion of the DIB, at least one name form for that object class shall be contained in the applicable part of the schema. The schema contains additional name forms as required.

The RDN attribute (or attributes) need not be chosen from the list of permitted attributes of the structural object class as specified in its structural or alias object class definition.

NOTE – Naming attributes are governed by DIT content rules and DIT context use in the same way as other attributes.

A name form is only a primitive element of the full specification required to constrain the form of the DIT to that required by the administrative and naming authorities that determine the naming policies of a given region of the DIT. The remaining aspects of the specification of DIT structure are discussed in 12.6.5.

12.6.3 Name form specification

Name forms may be defined as values of the **NAME-FORM** information object class:

```
NAME-FORM ::= CLASS {
        &namedObjectClass       OBJECT-CLASS,
        &MandatoryAttributes    ATTRIBUTE,
        &OptionalAttributes     ATTRIBUTE OPTIONAL,
        &id                     OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
        NAMES                   &namedObjectClass
        WITH ATTRIBUTES         &MandatoryAttributes
        [ AND OPTIONALLY        &OptionalAttributes ]
        ID                      &id }
```

For a name form which is defined using this information object class:

a) **&namedObjectClass** is the structural object class it names;

b) **&MandatoryAttributes** is the set of attributes which must be present in the RDN of the entry it governs;

c) **&OptionalAttributes** is the set of attributes which may be present in the RDN of the entry it governs;

d) **&id** is the object identifier assigned to it.

All attribute types in the mandatory and optional lists shall be different.

12.6.4 The structural object class of an entry

Some subschema specifications will include name forms for no more than one structural object class per structural object class superclass chain represented in the subschema.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Some subschema specifications may include name forms for more than one structural object class per structural object class superclass chain represented in the subschema.

In either case, with respect to a particular entry, only the most subordinate structural object class in the structural superclass chain present in the entry's **objectClass** attribute determines the DIT content rule and DIT structure rule applying to the entry. This class is referred to as the structural object class of the entry and is indicated by the **structuralObjectClass** operational attribute.

12.6.5 DIT structure rule definition

A DIT structure rule is a specification provided by the subschema administrative authority which the Directory uses to control the placement and naming of entries within the scope of the subschema. Each object and alias entry is governed by a single DIT structure rule. A subschema governing a subtree of the DIT will typically contain several DIT structure rules permitting several types of entries within the subtree.

A DIT structure rule definition includes:

a) an integer identifier which is unique within the scope of the subschema;

b) an indication of the name form for entries governed by the DIT structure rule;

c) the set of allowed superior structure rules, if required.

The set of DIT structure rules for a subschema specify the forms of distinguished names for entries governed by the subschema.

A DIT structure rule allows entries in a given subschema to subscribe to a particular name form. The form of the last RDN component of an entry's **DistinguishedName** is determined by the name form of the DIT structure rule governing the entry.

The **namedObjectClass** component of the name form (the name form's object class) corresponds to the structural object class of the entry.

A DIT structure rule shall only permit entries belonging to the structural object class identified by its associated name form. It does not permit entries belonging to any of the subclasses of the structural object class.

With respect to a particular entry, the DIT structure rule governing the entry is termed the entry's *governing structure rule*. This rule may be identified by examining the entry's **governingStructureRule** attribute.

With respect to a particular entry, the DIT structure rule governing the entry's superior is termed the entry's *superior structure rule*.

An entry may only exist in the DIT as a subordinate to another entry (the superior) if a DIT structure rule exists in the governing subschema which:

– indicates a name form for the structural object class of the entry; and

– either includes the entry's superior structure rule as a possible superior structure rule *or* does not specify a superior structure rule, in which case the entry must be a subschema administrative point.

If an entry which is itself a subschema administrative point is not included for the purposes of subschema administration in its subschema subentry, then the subschema from the immediately superior subschema administrative area is used to govern the entry.

Entries which are administrative point entries but have no subschema subentry (e.g. newly created administrative point entries), have no governing structure rule. The Directory shall not allow subordinates to be created below such entries until a subschema subentry has been added.

If an entry is converted to a new subschema administrative point, then the governing structure rule of all entries in the new subschema administrative area is automatically changed to that implied by the new subschema.

12.6.6 DIT structure rule specification

The abstract syntax of a DIT structure rule is expressed by the following ASN.1 type:

```
DITStructureRule  ::=  SEQUENCE {
      ruleIdentifier                RuleIdentifier ,
                                    -- must be unique within the scope of the subschema
      nameForm                      NAME-FORM.&id,
      superiorStructureRules        SET OF RuleIdentifier OPTIONAL }

RuleIdentifier  ::=  INTEGER
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The correspondence between the parts of the definition, as listed in 12.6.5, and the various components of the ASN.1 type defined above, is as follows:

a) the **ruleIdentifier** component identifies the DIT structure rule uniquely within a subschema;

b) the **nameForm** component of the DIT structure rule specifies the name form for entries governed by the DIT structure rule;

c) the **superiorStructureRules** component identifies permitted superior structure rules for entries governed by the rule. If this component is omitted, then the DIT structure rule applies to a subschema administrative point.

The **STRUCTURE-RULE** information object class is provided to facilitate the documentation of DIT structure rules:

```
STRUCTURE-RULE ::= CLASS {
        &nameForm                NAME-FORM,
        &SuperiorStructureRules  STRUCTURE-RULE OPTIONAL,
        &id                      RuleIdentifier }
WITH SYNTAX {
        NAME FORM                &nameForm
        [ SUPERIOR RULES         &SuperiorStructureRules ]
        ID                       &id }
```

12.7 DIT content rule definition

12.7.1 Overview

A DIT content rule specifies the permissible content of entries of a particular structural object class via the identification of an optional set of auxiliary object classes, mandatory, optional and precluded attributes. Collective attributes shall be included in DIT Content rules if they are to be permitted in an entry.

A DIT content rule definition includes:

a) an indication of the structural object class to which it applies;

b) optionally, an indication of the auxiliary object classes allowed for entries governed by the rule;

c) optionally, an indication of the mandatory attributes, over and above those called for by the structural and auxiliary object classes, required for entries governed by the DIT content rule;

d) optionally, an indication of the optional attributes, over and above those called for by the structural and auxiliary object classes, permitted for entries governed by the DIT content rule;

e) optionally, an indication of *optional* attribute(s) from the entry's structural and auxiliary object classes which are precluded from appearing in entries governed by the rule.

For any valid subschema specification, there is at most one DIT content rule for each structural object class.

Every entry in the DIT is governed by at most one DIT content rule. This rule may be identified by examining the value of the entry's **structuralObjectClass** attribute.

If no DIT content rule is present for a structural object class, then entries of that class shall contain only the attributes permitted by the structural object class definition.

The DIT content rules of superclasses of the structural object class for an entry do not apply to that entry.

As a DIT content rule is associated with a structural object class, it follows that all entries of the same structural object class will have the same DIT content rule regardless of the DIT structure rule governing their location in the DIT.

An entry governed by a DIT content rule may, in addition to the structural object class of the DIT structure rule, be associated with a subset of the auxiliary object classes identified by the DIT content rule. This association is reflected in the entry's **objectClass** attribute.

An entry's content must be consistent with the object classes indicated by its **objectClass** attribute in the following way:

– mandatory attributes of object classes indicated by the **objectClass** attribute shall *always* be present in the entry;

– optional attributes (not indicated as additional optional or mandatory in the DIT content rule) of auxiliary object classes indicated by the DIT content rule may only be present if the **objectClass** attribute indicates these auxiliary object classes.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Mandatory attributes associated with the structural or indicated auxiliary object classes shall not be precluded in a DIT content rule.

12.7.2 DIT content rule specification

The abstract syntax of a DIT content rule is expressed by the following ASN.1 type:

```
DITContentRule ::= SEQUENCE {
        structuralObjectClass       OBJECT-CLASS.&id,
        auxiliaries                 SET OF OBJECT-CLASS.&id  OPTIONAL,
        mandatory           [1]     SET OF ATTRIBUTE.&id     OPTIONAL,
        optional            [2]     SET OF ATTRIBUTE.&id     OPTIONAL,
        precluded           [3]     SET OF ATTRIBUTE.&id     OPTIONAL }
```

The correspondence between the parts of the definition, as listed in 12.7.1, and the various components of the ASN.1 type defined above, is as follows:

a) the **structuralObjectClass** component identifies the structural object class to which the DIT content rule applies;

b) the **auxiliaries** component identifies the auxiliary object classes allowed for an entry to which the DIT content rule applies;

c) the **mandatory** component specifies user attribute types which an entry to which the DIT content rule applies shall contain in addition to those which it shall contain according to its structural and auxiliary object classes;

d) the **optional** components specify user attribute types which an entry to which the DIT content rule applies may contain in addition to those which it may contain according to its structural and auxiliary object classes;

e) the **precluded** component specifies a subset of the optional user attribute types of the structural and auxiliary object classes which are precluded from an entry to which the DIT content rule applies.

The **CONTENT-RULE** information object class is provided to facilitate the documentation of DIT content rules:

```
CONTENT-RULE ::= CLASS {
        &structuralClass            OBJECT-CLASS.&id    UNIQUE,
        &Auxiliaries                OBJECT-CLASS        OPTIONAL,
        &Mandatory                  ATTRIBUTE           OPTIONAL,
        &Optional                   ATTRIBUTE           OPTIONAL,
        &Precluded                  ATTRIBUTE           OPTIONAL }
WITH SYNTAX {
        STRUCTURAL OBJECT-CLASS     &structuralClass
        [ AUXILIARY OBJECT-CLASSES  &Auxiliaries ]
        [ MUST CONTAIN              &Mandatory ]
        [ MAY CONTAIN               &Optional ]
        [ MUST-NOT CONTAIN          &Precluded ] }
```

12.8 Context type definition

The definition of a context type involves:

a) specifying the syntax of the context;

b) specifying the syntax of a context assertion;

c) defining the semantics of the context;

d) specifying how matches are done; and

e) assigning an object identifier to the context type.

12.8.1 Context Value matching

A presented context assertion matches a stored context value of the same context type according to the description of matching which is part of the context definition.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

12.8.2 Context definition

Contexts are defined using the **CONTEXT** information object class:

```
CONTEXT ::= CLASS {
        &Type,
        &Assertion          OPTIONAL,
        &id                 OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
        WITH SYNTAX         &Type
        [ ASSERTED AS       &Assertion ]
        ID                  &id }
```

If the **&Assertion** is omitted, the context assertion syntax is the same as **&Type**.

When a context is defined, the specification shall include a description of the semantics of the context, and how a match is evaluated.

ITU-T Rec. X.520 | ISO/IEC 9594-6 specifies selected Context Definitions.

12.9 DIT Context Use definition

12.9.1 Overview

A DIT Context Use is a specification provided by the subschema administrative authority to specify the permissible context types that may be stored with an attribute, and the mandatory context types that shall be stored with an attribute.

A DIT Context Use definition includes:

a) an indication of the attribute type to which it applies;

b) optionally, an indication of the mandatory context types that shall be associated with values of the attribute type whenever the attribute is stored;

c) optionally, an indication of the optional context types that may be associated with values of the attribute type whenever the attribute is stored.

If no DIT Context Use definition is present for a given attribute type, then values of attributes of that type shall contain no context lists. For a given subschema administrative area, there can be only one DIT Context Use for a given attribute type. A DIT Context Use may be defined to apply to all attribute types, in which case it shall be the only DIT Context Use in the subschema.

12.9.2 DIT Context Use specification

The abstract syntax of a DIT Context Use is expressed by the following ASN.1 type:

```
DITContextUse ::= SEQUENCE {
        attributeType                   ATTRIBUTE.&id,
        mandatoryContexts       [1]     SET OF CONTEXT.&id OPTIONAL,
        optionalContexts        [2]     SET OF CONTEXT.&id OPTIONAL }
```

The correspondence between the parts of the definition, as listed in 12.9.1, and the various components of the ASN.1 type defined above, is as follows:

a) the **attributeType** component identifies the attribute type to which the DIT Context Use applies, or any attribute type (**id-oa-allAttributeTypes**);

b) the **mandatoryContexts** component specifies context types that shall be associated with an attribute value of the given type whenever the attribute is stored. If this is omitted, then attribute values may exist without context lists;

c) the **optionalContexts** component specifies context types that may be associated with an attribute value of the given type whenever the attribute is stored. If this is omitted but mandatoryContexts is present, then all attribute values shall appear with the mandatory context types and no others. If this is omitted and mandatoryContexts is also omitted, it is equivalent to having no DIT Context Use for the attribute type; that is, attribute values of the given attribute type shall not have associated context lists.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The **DIT-CONTEXT-USE-RULE** information object class is provided to facilitate the documentation of the DIT Context Use rules:

```
DIT-CONTEXT-USE-RULE ::= CLASS {
        &attributeType          ATTRIBUTE.&id   UNIQUE,
        &Mandatory              CONTEXT         OPTIONAL,
        &Optional               CONTEXT         OPTIONAL }
WITH SYNTAX {
        ATTRIBUTE TYPE          &attributeType
        [ MANDATORY CONTEXTS    &Mandatory ]
        [ OPTIONAL CONTEXTS     &Optional ] }
```

13 Directory System Schema

13.1 Overview

The Directory System Schema is a set of definitions and constraints concerning the information that the Directory itself needs to know in order to operate correctly. This information is specified in terms of subentries and operational attributes.

NOTE – The system schema enables the directory system to, for example:

- prevent the association of subentries of the wrong type with administrative entries (e.g. the creation of a subschema subentry subordinate to an administrative entry defined only as a security administrative entry);
- prevent the addition of inappropriate operational attributes to an entry or subentry (e.g. a subschema operational attribute to a person's entry).

Formally, the Directory System Schema comprises a set of:

a) Object class definitions that define the attributes that shall or may be present in a subentry of a given class;

b) Operational Attribute Type definitions that specify the characteristics of operational attributes known and used by the Directory.

The complete definition of an operational attribute includes a specification of the way in which the Directory uses and (if appropriate) provides or manages, the attribute in the course of its operation.

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

The Directory System Schema defined in this Directory Specification is an integral part of the Directory System itself. Each DSA participating in a directory system requires a full knowledge of the system schema established by its Administrative Authority. The system schema for an Administrative Area may be defined by the Administrative Authority using the notation defined in this clause.

The Directory System Schema is not regulated by DIT structure or content rules. When an element of system schema is defined, a specification of how it is used and where it appears in the DIT is provided.

Certain aspects of the directory system schema are specified in the following subclauses.

The directory system schema required to support directory distribution is specified in clauses 21 through 24.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

13.2 System schema supporting the administrative and operational information model

Although **subentry** and **subentryNameForm** are specified using the notation of clause 12, subentries are not regulated by DIT structure or DIT content rules.

13.2.1 The Subentry object class

The **subentry** object class is a structural object class and is defined as follows:

```
subentry  OBJECT-CLASS ::= {
        SUBCLASS OF     { top }
        KIND            structural
        MUST CONTAIN    { commonName | subtreeSpecification }
        ID              id-sc-subentry }
```

13.2.2 The Subentry name form

The **subentryNameForm** name form allows entries of class **subentry** to be named using the **commonName** attribute:

```
subentryNameForm  NAME-FORM  ::=  {
        NAMES                   subentry
        WITH ATTRIBUTES         { commonName }
        ID                      id-nf-subentryNameForm }
```

No other name form shall be used for subentries.

13.2.3 The Subtree Specification operational attribute

The **subtreeSpecification** operational attribute, whose semantics are specified in clause 10, is defined as follows:

```
subtreeSpecification  ATTRIBUTE  ::=  {
        WITH SYNTAX     SubtreeSpecification
        SINGLE VALUE    TRUE
        USAGE           directoryOperation
        ID              id-oa-subtreeSpecification }
```

This attribute is present in all subentries.

13.3 System schema supporting the administrative model

The Administrative Model defined in clause 10 requires that administrative entries contain an **administrativeRole** attribute to indicate that the associated administrative area is concerned with one or more administrative roles.

The **administrativeRole** operational attribute is specified as follows:

```
administrativeRole  ATTRIBUTE  ::=  {
        WITH SYNTAX                     OBJECT-CLASS.&id
        EQUALITY MATCHING RULE          objectIdentifierMatch
        USAGE                           directoryOperation
        ID                              id-oa-administrativeRole }
```

The values of this attribute defined by this Directory Specification are:

```
id-ar-autonomousArea
id-ar-accessControlSpecificArea
id-ar-accessControlInnerArea
id-ar-subschemaAdminSpecificArea
id-ar-collectiveAttributeSpecificArea
id-ar-collectiveAttributeInnerArea
id-ar-contextDefaultSpecificArea
```

The semantics of these values are defined in clause 11.

The **administrativeRole** operational attribute is also used to regulate the subentries permitted to be subordinate to an administrative entry. A subentry not of a class permitted by the **administrativeRole** attribute may not be subordinate to the administrative entry.

13.4 System schema supporting general administrative and operational requirements

The following clauses describe subschema operational attributes which are not attributes in the usual sense (i.e. are not held within an entry), but may be thought of as 'virtual' attributes, representing information which is derivable (e.g. from existing operational attributes, their values, and other information). Such virtual attributes are valid for all entries within an administrative area. This has the effect that these subschema operational attributes appear to be present in every entry.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

13.4.1 Timestamps

The **createTimestamp** indicates the time that an entry was created:

```
createTimestamp  ATTRIBUTE  ::= {
        WITH SYNTAX                     GeneralizedTime
                        -- as per 41.3 b) or c) of ITU-T Rec. X.680 | ISO/IEC 8824-1
        EQUALITY MATCHING RULE          generalizedTimeMatch
        ORDERING MATCHING RULE          generalizedTimeOrderingMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-createTimestamp }
```

The **modifyTimeStamp** indicates the time that an entry was last modified:

```
modifyTimestamp  ATTRIBUTE  ::= {
        WITH SYNTAX                     GeneralizedTime
                        -- as per 41.3 b) or c) of ITU-T Rec. X.680 | ISO/IEC 8824-1
        EQUALITY MATCHING RULE          generalizedTimeMatch
        ORDERING MATCHING RULE          generalizedTimeOrderingMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-modifyTimestamp }
```

The **subschemaTimestamp** indicates the time that the subschema subentry for the entry (see 14.3) was created or last modified. It is available in every entry:

```
subschemaTimestamp  ATTRIBUTE  ::= {
        WITH SYNTAX                     GeneralizedTime
                        -- as per 41.3 b) or c) of ITU-T Rec. X.680 | ISO/IEC 8824-1
        EQUALITY MATCHING RULE          generalizedTimeMatch
        ORDERING MATCHING RULE          generalizedTimeOrderingMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-subschemaTimestamp }
```

The **generalizedTimeMatch** and **generalizedTimeOrderingMatch** matching rules are defined in ITU-T Rec. X.520 | ISO/IEC 9594-6.

13.4.2 Entry Modifier operational attributes

The **creatorsName** operational attribute indicates the distinguished name of the Directory user that created an entry:

```
creatorsName  ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-creatorsName }
```

The **modifiersName** operational attribute indicates the distinguished name of the Directory user that last modified the entry:

```
modifiersName  ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-modifiersName }
```

These operational attributes shall use the primary distinguished name.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

13.4.3 Subentry identification operational attributes

The **subschemaSubentry** operational attribute identifies the subschema subentry that governs the entry. It is available in every entry:

```
subschemaSubentryList  ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-subschemaSubentryList }
```

The **accessControlSubentry** operational attribute identifies all access control subentries that affect the entry. It is available in every entry.

```
accessControlSubentryList  ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-accessControlSubentryList }
```

The **collectiveAttributeSubentry** operational attribute identifies all collective attribute subentries that affect the entry. It is available in every entry:

```
collectiveAttributeSubentryList  ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-collectiveAttributeSubentryList }
```

The **contextDefaultSubentry** operational attribute identifies all context default subentries that affect the entry. It is available in every entry:

```
contextDefaultSubentryList  ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-contextDefaultSubentryList }
```

13.4.4 Has Subordinates operational attribute

The **hasSubordinates** operational attribute indicates whether any subordinate entries exist below the entry holding this attribute. A value of **TRUE** indicates that subordinates may exist. A value of **FALSE** indicates that no subordinates exist. If this attribute is absent, no information is provided about the existence of subordinate entries. The attribute will ordinarily disclose the existence of subordinates even if the immediate subordinates are hidden by access controls – to prevent disclosure of the existence of subordinates the operational attribute itself must be protected by access controls.

NOTE – A value of **TRUE** may be returned when no subordinates exist if all possible subordinates are available only through a non-specific subordinate reference (see ITU-T Rec. X.518 | ISO/IEC 9594-4) or if the only subordinates are subentries.

```
hasSubordinates ATTRIBUTE ::= {
        WITH SYNTAX                     BOOLEAN
        EQUALITY MATCHING RULE          booleanMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-hasSubordinates }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

13.5 System schema supporting access control

13.5.1 Access control subentries

If a subentry contains prescriptive access control information, then its **objectClass** attribute shall contain the value **accessControlSubentry**:

```
accessControlSubentry  OBJECT-CLASS  ::= {
        KIND          auxiliary
        ID            id-sc-accessControlSubentry }
```

A subentry of this object class shall contain precisely one prescriptive ACI attribute of a type consistent with the value of the **id-sc-accessControlScheme** attribute of the corresponding access control specific point.

13.6 System schema supporting the collective attribute model

Subentries supporting collective attribute specific or inner administrative areas are defined as follows:

```
collectiveAttributeSubentry  OBJECT-CLASS  ::= {
        KIND          auxiliary
        ID            id-sc-collectiveAttributeSubentry }
```

A subentry of this object class shall contain at least one collective attribute.

Collective attributes contained within a subentry of this object class are conceptually available for interrogation and filtering at every entry within the scope of the subentry's **subtreeSpecification** attribute, but are administered via the subentry.

The **collectiveExclusions** operational attribute allows particular collective attributes to be excluded from an entry:

```
collectiveExclusions  ATTRIBUTE  ::= {
        WITH SYNTAX                     OBJECT IDENTIFIER
        EQUALITY MATCHING RULE          objectIdentifierMatch
        USAGE                           directoryOperation
        ID                              id-oa-collectiveExclusions }
```

This attribute is optional for every entry.

The **OBJECT IDENTIFIER** value **id-oa-excludeAllCollectiveAttributes** may be used, by its presence as a value of the **collectiveExclusions** attribute, to exclude all collective attributes from an entry.

13.7 System schema supporting context assertion defaults

Subentries providing default values for context assertions are defined as follows:

```
contextAssertionSubentry  OBJECT-CLASS  ::= {
        KIND              auxiliary
        MUST CONTAIN      {contextAssertionDefaults}
        ID                id-sc-contextAssertionSubentry }
```

A subentry of this object class shall contain a **contextAssertionDefaults** attribute:

```
contextAssertionDefaults  ATTRIBUTE  ::= {
        WITH SYNTAX                     TypeAndContextAssertion
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-oa-contextAssertionDefault }
```

Whenever a context is evaluated and no context assertion is provided by the user, the Directory provides context assertion defaults equal to the values of this attribute in the context assertion subentry controlling the entry being accessed, as described in 8.8.2.2.

NOTE – **TypeAndContextAssertion** is defined in 7.6 of (and evaluation of it is defined in 7.6.3 of) ITU-T Rec. X.511 | ISO/IEC 9594-3.

13.8 Maintenance of system schema

It is the responsibility of DSAs to maintain consistency of subentries and operational attributes with the system schema. Inconsistency between various aspects of system schema, and between system schema and subentries and operational attributes, shall not occur.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The Directory executes entry addition and modification procedures whenever a new subentry is added to the DIT or an existing subentry is modified. The Directory shall determine whether the proposed operation would violate the system schema; if it does the modification shall fail.

In particular, the Directory ensures that subentries added to the DIT are consistent with the values of the **administrativeRole** attribute, that the attributes within the subentry are consistent with the values of the subentry's **objectClass** attribute.

The value of the **administrativeRole** attribute may be modified to permit classes of subentries to be subordinate to the administrative entry that are not yet present. The value of the **administrativeRole** attribute shall not be modified so as to cause existing subentries to become inconsistent.

The Directory also ensures, where the values of operational attributes are provided by the Directory, that they are correct.

13.9 System schema for first-level subordinates

The Directory enforces the following rules and constraints on entries created immediately subordinate to the DIT root:

- All such entries shall be created as administrative point entries.
- The object class and naming attributes of such entries shall be as specified in CCITT Rec. X.660 | ISO/IEC 9834-1.

14 Directory schema administration

14.1 Overview

The overall administration of the directory schema of the global DIT is realized through independent administration of the subschemas of the autonomous administrative areas of the DIT Domains that constitute the global DIT.

Coordination of the administration of the directory schema at boundaries between DIT Domains is a subject for bilateral agreement between DMOs and is beyond the scope of this Directory Specification.

The subschema administrative capabilities defined in this clause for the purpose of managing a DIT domain include:

1) creation, deletion and modification of subschema subentries;
2) support of the publication mechanism for the purpose of permitting DSAs to include schema information in operational binding protocol exchanges and DUAs to retrieve subschema information via DAP;
3) subschema regulation for the purpose of ensuring that any modify operations will be performed in accordance with the applicable subschema specification.

14.2 Policy objects

A subschema policy object may be one of the following:

- a subschema administrative area;
- an object or alias entry within a subschema administrative area;
- a user attribute of such an object or alias entry.

An autonomous administrative area may be designated as a subschema specific administrative area in order to administer the subschema. This shall be indicated by the presence of the value **id-oa-subschemaAdminSpecificArea** in the associated administrative entry's **administrativeRole** attribute (in addition to the presence of the value **id-oa-autonomousArea**, and possibly other values).

Such an autonomous administrative area may be partitioned in order to deploy and administer the subschema of the specific partitions. In this case, the administrative entries for each of the subschema specific administrative areas are indicated by the presence of the value **id-oa-subschemaAdminSpecificArea** in these entries' **administrativeRole** attributes.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

4.3 Policy parameters

Subschema policy parameters are used to express the policies of the subschema Administrative Authority. These parameters, and the operational attributes used to represent them, are:

- a *DIT structure* parameter: used to define the structure of the subschema administrative area and to store information about obsolete DIT structure rules which some entries may have identified as their governing DIT structure rule. This parameter is represented by the **dITStructureRules** and **nameForms** operational attributes;
- a *DIT content* parameter: used to define the type of content of object and alias entries contained within the subschema administrative area and to store information about obsolete DIT content rules which the Directory may have used in determining the content of some entries. This parameter is represented by the **dITContentRules**, **objectClasses, attributeTypes**, **contextTypes**, and **dITContextUse** operational attributes;
- a *matching capability* parameter: used to define the matching capabilities supported by matching rules as applied to the attributes types defined in a subschema administrative area. This parameter is represented by the **matchingRules** and **matchingRuleUse** operational attributes.

A single subschema subentry is used by the subschema authority to administer the subschema for the subschema administrative area. For this purpose, the subschema subentry contains the operational attributes representing the policy parameters used to express subschema policies. The **subtreeSpecification** attribute of a subschema subentry shall specify the whole subschema administrative area, i.e. it shall be an empty sequence.

The subschema subentry is specified as follows:

```
subschema  OBJECT-CLASS  ::= {
        KIND                auxiliary
        MAY CONTAIN         {
                            dITStructureRules |
                            nameForms |
                            dITContentRules |
                            objectClasses |
                            attributeTypes |
                            contextTypes |
                            dITContextUse |
                            matchingRules |
                            matchingRuleUse }
        ID                  id-soc-subschema }
```

The operational attributes of the subschema subentry are defined in 14.7.

14.4 Policy procedures

There are two policy procedures associated with subschema administration:

- a subschema modification procedure;
- an entry modification procedure.

14.5 Subschema modification procedures

A subschema authority may administer a subschema in a dynamic fashion, including making restrictive subschema modifications. This may be accomplished by modifying the values of the subschema operational attributes, using Directory modify operations, effectively changing the subschema which is in force in the subschema administrative area. A subschema authority may also create new subschema areas, or remove existing subschema areas by creating or removing subschema subentries, respectively.

Before the subschema authority extends the DIT structure or DIT content rules by adding a new rule, or by adding an auxiliary object class, or a mandatory or an optional attribute to an existing rule, the referenced schema information shall be described in the appropriate attribute in the subschema subentry. Name forms, object classes, attribute types and matching rules that are referenced (directly or indirectly) by a **dITStructureRule**, **dITContentRule** or by a **matchingRuleUse** attribute shall not be removed from the subschema subentry.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The definition of information objects such as object classes, attribute types, matching rules and name forms which have been registered (i.e. assigned a name of type object identifier) are static and cannot be modified. Changes to the semantics of such information objects require the assignment of new object identifiers.

DIT structure and DIT content rules may be active or obsolete. Only active rules are used to regulate the DIT. The identification and preservation of obsolete rules is an administrative convenience allowing location (and possibly repair) of entries added under old rules that have since changed.

This obsolete mechanism shall be used where restrictive changes are made to DIT structure or DIT content rules creating inconsistencies in the DIB, otherwise the appropriate active rule may be modified directly. The Directory permits deletion of obsolete rules at any time.

NOTE – The obsolete mechanism provided in subschema operational attributes ensures that all entries with obsolete schema can be identified and repaired before the obsolete subschema operational attribute is deleted.

It is the responsibility of the Subschema Administrative Authority to maintain consistency of entries with the active subschema by means of the Directory abstract service, or by other local means. This may be done at the convenience of the Subschema Administrative Authority. It is not defined when such an adjustment of inconsistent entries should be done. However, deletion of obsolete rules prior to the location and repair of inconsistent entries will make this task more difficult.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

14.6 Entry addition and modification procedures

The Directory executes entry addition and modification procedures whenever a new entry is added to the DIT or an existing entry is modified. The Directory must determine whether the proposed operation would violate a subschema policy.

In particular, the Directory shall ensure that entries added to the DIT are consistent with appropriate active DIT structure and DIT content rules.

The Directory shall allow interrogation of entries which are inconsistent with their active rules.

The Directory enforces active rules when requested to modify the DIB. If an entry is inconsistent with its active rule, a request to modify the entry shall be permitted if it repairs an existing inconsistency, or does not introduce a new inconsistency. A request which introduces a new inconsistency shall fail.

For any valid entry in a valid subschema administrative area, there can be only one most subordinate structural object class in the structural object class superclass chain. When an entry is added to the DIT, the Directory determines this most subordinate structural object class from the **objectClass** attribute values provided and permanently associates it with the entry via the entry's **structuralObjectClass** attribute.

When an entry is created, values of the **objectClass** attribute shall be provided so that the content of the entry is consistent with the DIT content rule governing the entry. In particular, where a value of the **objectClass** attribute identifies a particular object class having superclasses other than **top**, then values for all of these superclasses must also be provided. Otherwise the Directory operation creating the entry shall fail.

Directory users may subsequently add or delete values of the **objectClass** attribute for the auxiliary object classes of an entry. The content of an entry shall remain consistent with the DIT content rule governing the entry following a change to the values of the **objectClass** attribute. In particular, where a value of the **objectClass** attribute identifies a particular object class having superclasses other than **top** is added or deleted, then values for all of these superclasses must also be added or deleted, except where such superclasses are also present in the superclass chains associated with other values not being added or deleted respectively.

14.7 Subschema policy attributes

The following subclauses specify the subschema policy operational attributes. These attributes are:

- present in the subschema subentry. The values of these attributes are administered via Directory modify operations using the distinguished name of the subschema subentry;
- available for interrogation in all entries governed by the subschema.

The ASN.1 parameterized type **DirectoryString { ub-schema }**, used in the following definitions, is defined in ITU-T Rec. X.520 | ISO/IEC 9594-6.

The **integerFirstComponentMatch** and **objectIdentifierFirstComponentMatch** equality matching rules are also defined in ITU-T Rec. X.520 | ISO/IEC 9594-6.

For management purposes, a number of human readable **name** components and a **description** component are optionally allowed as components of a number of the subschema policy operational attributes defined in the following subclauses.

A number of subschema policy operational attributes defined in the following clauses contain an **obsolete** component. This component is used to indicate whether the definition is active or obsolete in the subschema administrative area.

14.7.1 DIT Structure Rules operational attribute

The **dITStructureRules** operational attribute defines the DIT structure rules which are in force within a subschema:

```
dITStructureRules ATTRIBUTE  ::=  {
        WITH SYNTAX                     DITStructureRuleDescription
        EQUALITY MATCHING RULE          integerFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-dITStructureRule }

DITStructureRuleDescription  ::=  SEQUENCE {
        COMPONENTS OF       DITStructureRule,
        name            [1] SET OF DirectoryString { ub-schema } OPTIONAL,
        description         DirectoryString { ub-schema } OPTIONAL,
        obsolete            BOOLEAN DEFAULT FALSE }
```

The **dITStructureRules** operational attribute is multi-valued; each value defines one DIT structure rule.

The components of **dITStructureRule** have the same semantics as the corresponding ASN.1 definition in 12.6.6.

14.7.2 DIT Content Rules operational attribute

The **dITContentRules** operational attribute defines the DIT content rules which are in force within a subschema. Each value of the operational attribute is tagged by the object identifier of the structural object class to which it pertains:

```
dITContentRules ATTRIBUTE  ::=  {
        WITH SYNTAX                     DITContentRuleDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-dITContentRules }

DITContentRuleDescription  ::=  SEQUENCE {
        COMPONENTS OF       DITContentRule,
        name            [4] SET OF DirectoryString { ub-schema } OPTIONAL,
        description         DirectoryString { ub-schema }OPTIONAL,
        obsolete            BOOLEAN DEFAULT FALSE }
```

The **dITContentRules** operational attribute is multi-valued; each value defines one DIT content rule.

The components of **dITContentRule** have the same semantics as the corresponding ASN.1 definition in 12.7.2.

14.7.3 Matching Rules operational attribute

The **matchingRules** operational attribute specifies the matching rules used within a subschema:

```
matchingRules  ATTRIBUTE  ::=  {
        WITH SYNTAX                     MatchingRuleDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-matchingRules }

MatchingRuleDescription  ::=  SEQUENCE {
        identifier              MATCHING-RULE.&id,
        name                    SET OF DirectoryString { ub-schema }     OPTIONAL,
        description             DirectoryString { ub-schema }        OPTIONAL,
        obsolete                BOOLEAN                                  DEFAULT FALSE,
        information         [0] DirectoryString { ub-schema } }
                            -- describes the ASN.1 syntax
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The **identifier** component of a value of the **matchingRules** attribute is the object identifier identifying the matching rule.

The **description** component contains a natural language description of the algorithms associated with the rule.

The **information** component contains the ASN.1 definition of the assertion syntax of the rule.

Such an ASN.1 definition shall be given as an optional ASN.1 Imports production, followed by optional ASN.1 Assignment productions, followed by an ASN.1 Type production. All type names defined in Directory modules are implicitly imported and do not require explicit import. All type names, whether imported or defined via an Assignment, are local to the definition of this syntax. If the ASN.1 type includes a user-defined constraint and is not one of the ASN.1 types defined in the Directory modules, then the last UserDefinedConstraintParameter of the constraint shall be an actual parameter whose governing type is **SyntaxConstraint** and whose value is the object identifier assigned to the constraint.

```
SyntaxConstraint ::= OBJECT IDENTIFIER
```

NOTE 1 – The ASN.1 productions Imports, Assignment, and Type are defined in ITU-T Rec. X.680 | ISO/IEC 8824-1. UserDefinedConstraintParameter is defined in ITU-T Rec. X.680 | ISO/IEC 8824-1.

NOTE 2 – A typical ASN.1 definition is simply a Type name.

The **matchingRules** operational attribute is multi-valued; each value describes one matching rule.

14.7.4 Attribute Types operational attribute

The **attributeTypes** operational attribute specifies the attribute types used within a subschema:

```
attributeTypes ATTRIBUTE ::= {
        WITH SYNTAX                     AttributeTypeDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-attributeTypes }

AttributeTypeDescription  ::=  SEQUENCE  {
        identifier              ATTRIBUTE.&id,
        name                    SET OF DirectoryString { ub-schema }     OPTIONAL,
        description             DirectoryString { ub-schema }        OPTIONAL,
        obsolete                BOOLEAN                                  DEFAULT FALSE,
        information       [0]   AttributeTypeInformation }
```

The **identifier** component of a value of the **attributeTypes** attribute is the object identifier identifying the attribute type.

The **attributeTypes** operational attribute is multi-valued; each value describes one attribute type:

```
AttributeTypeInformation  ::=  SEQUENCE  {
        derivation              [0]   ATTRIBUTE.&id OPTIONAL,
        equalityMatch           [1]   MATCHING-RULE.&id                  OPTIONAL,
        orderingMatch           [2]   MATCHING-RULE.&id                  OPTIONAL,
        substringsMatch         [3]   MATCHING-RULE.&id                  OPTIONAL,
        attributeSyntax         [4]   DirectoryString { ub-schema }      OPTIONAL,
        multi-valued            [5]   BOOLEAN                            DEFAULT TRUE,
        collective              [6]   BOOLEAN                            DEFAULT FALSE,
        userModifiable          [7]   BOOLEAN                            DEFAULT TRUE,
        application                   AttributeUsage                     DEFAULT  userApplications }
```

The **derivation**, **equalityMatch**, **attributeSyntax**, **multi-valued**, **collective** and **application** components have the same semantic as the equivalent pieces of notation introduced by the corresponding information object class.

The **attributeSyntax** component contains a text string giving the ASN.1 definition of the attribute's syntax. Such an ASN.1 definition shall be given as specified for the **information** component of the Matching Rules operational attribute.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

14.7.5 Object Classes operational attribute

The **objectClasses** operational attribute specifies the object classes used within a subschema.

```
objectClasses ATTRIBUTE  ::=  {
        WITH SYNTAX                     ObjectClassDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-objectClasses }

ObjectClassDescription  ::=  SEQUENCE  {
        identifier              OBJECT-CLASS.&id,
        name                    SET OF DirectoryString { ub-schema }    OPTIONAL,
        description             DirectoryString { ub-schema }           OPTIONAL,
        obsolete                BOOLEAN                                 DEFAULT FALSE,
        information       [0]   ObjectClassInformation }
```

The **identifier** component of a value of the **objectClasses** attribute is the object identifier identifying the object class.

The **objectClasses** operational attribute is multi-valued; each value describes one object class:

```
ObjectClassInformation  ::=  SEQUENCE  {
        subclassOf              SET OF OBJECT-CLASS.&id         OPTIONAL,
        kind                    ObjectClassKind                 DEFAULT structural,
        mandatories       [3]   SET OF ATTRIBUTE.&id            OPTIONAL,
        optionals         [4]   SET OF ATTRIBUTE.&id            OPTIONAL }
```

The **subclassOf**, **kind**, **mandatories** and **optionals** components have the same semantics as the corresponding pieces of notation introduced by the corresponding information object class.

14.7.6 Name Forms operational attribute

The **nameForms** operational attribute specifies the name forms used within a subschema.

```
nameForms ATTRIBUTE  ::=  {
        WITH SYNTAX                     NameFormDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-nameForms }

NameFormDescription  ::=  SEQUENCE  {
        identifier              NAME-FORM.&id,
        name                    SET OF DirectoryString { ub-schema } OPTIONAL,
        description             DirectoryString { ub-schema } OPTIONAL,
        obsolete                BOOLEAN DEFAULT FALSE,
        information       [0]   NameFormInformation }
```

The **identifier** component of a value of the **nameForms** attribute is the object identifier identifying the object class.

The **nameForms** operational attribute is multi-valued; each value describes one name form:

```
NameFormInformation  ::=  SEQUENCE  {
        subordinate             OBJECT-CLASS.&id,
        namingMandatories       SET OF ATTRIBUTE.&id,
        namingOptionals SET OF ATTRIBUTE.&id OPTIONAL }
```

The **subordinate**, **mandatoryNamingAttributes** and **optionalNamingAttributes** components have the same semantics as the corresponding pieces of notation introduced by the corresponding information object class.

14.7.7 Matching Rule Use operational attribute

The **matchingRuleUse** operational attribute is used to indicate the attribute types to which a matching rule applies in a subschema:

```
matchingRuleUse ATTRIBUTE  ::=  {
        WITH SYNTAX                     MatchingRuleUseDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-matchingRuleUse }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
MatchingRuleUseDescription ::= SEQUENCE {
        identifier              MATCHING-RULE.&id,
        name                    SET OF DirectoryString { ub-schema }    OPTIONAL,
        description             DirectoryString { ub-schema }           OPTIONAL,
        obsolete                BOOLEAN                                 DEFAULT FALSE,
        information     [0]     SET OF ATTRIBUTE.&id }
```

The **identifier** component of a value of the **matchingRulesUse** attribute is the object identifier identifying the matching rule.

The **information** component of a value identifies the set of attribute types to which the matching rule applies.

14.7.8 Structural Object Class operational attribute

Every entry in the DIT has a **structuralObjectClass** operational attribute which indicates the structural object class of the entry:

```
structuralObjectClass ATTRIBUTE ::= {
        WITH SYNTAX                     OBJECT IDENTIFIER
        EQUALITY MATCHING RULE          objectIdentifierMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-soa-structuralObjectClass }
```

14.7.9 Governing Structure Rule operational attribute

Every entry in the DIT, with the exception of administrative point entries that have no subschema subentry, has a **governingStructureRule** operational attribute which indicates the governing structure rule of the entry:

```
governingStructureRule ATTRIBUTE ::= {
        WITH SYNTAX                     INTEGER
        EQUALITY MATCHING RULE          integerMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-soa-governingStructureRule }
```

14.7.10 ContextTypes operational attribute

The **contextTypes** operational attribute specifies the context types used within a subschema.

```
contextTypes ATTRIBUTE ::= {
        WITH SYNTAX                     ContextDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-contextTypes }

ContextDescription ::= SEQUENCE {
        identifier              CONTEXT.&id,
        name                    SET OF DirectoryString {ub-schema}      OPTIONAL,
        description             DirectoryString { ub-schema }           OPTIONAL,
        obsolete                BOOLEAN                                 DEFAULT FALSE,
        information     [0]     ContextInformation }
```

The **identifier** component of a value of the **contextTypes** operational attribute is the object identifier identifying the context type.

The **contextTypes** operational attribute is multi-valued; each value describes one context type:

```
ContextInformation ::= SEQUENCE {
        syntax                  DirectoryString { ub-schema } ,
        assertionSyntax         DirectoryString { ub-schema } OPTIONAL }
```

The **syntax** and **assertionSyntax** components have the same semantics as the corresponding pieces of notation introduced in the corresponding information object class.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The **syntax** component and the **assertionSyntax** component each contain a text string giving the ASN.1 definition of the context syntax and context assertion syntax respectively. Such an ASN.1 definition shall be given as an optional ASN.1 Imports production, followed by optional ASN.1 Assignment productions, followed by an ASN.1 Type production. All type names defined in Directory modules are implicitly imported and do not require explicit import. All type names, whether imported or defined via an Assignment, are local to the definition of this syntax. If the ASN.1 type includes a user-defined constraint and is not one of the ASN.1 types defined in the Directory modules, then the last UserDefinedConstraintParameter of the constraint shall be an actual parameter whose governing type is **SyntaxConstraint** and whose value is the object identifier assigned to the constraint.

NOTE 1 – **UserDefinedConstraintParameter** and the ASN.1 productions Imports, Assignment, and Type are defined in ITU-T Rec. X.680 | ISO/IEC 8824-1. **SyntaxConstraint** is defined in 14.7.3.

NOTE 2 – A typical ASN.1 definition is simply a Type name.

14.7.11 DIT Context Use operational attribute

The **dITContextUse** operational attribute is used to indicate the contexts which shall or may be used with an attribute:

```
dITContextUse ATTRIBUTE  ::= {
        WITH SYNTAX                     DITContextUseDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-dITContextUse }

DITContextUseDescription  ::=  SEQUENCE  {
        identifier                ATTRIBUTE.&id,
        name                      SET OF DirectoryString { ub-schema }     OPTIONAL,
        description               DirectoryString { ub-schema }            OPTIONAL,
        obsolete                  BOOLEAN                                  DEFAULT FALSE,
        information         [0]   DITContextUseInformation }
```

The **identifier** component of a value of the **dITContextUse** operational attribute is the object identifier of the attribute type to which it applies. The value **id-oa-allAttributeTypes** indicates that it applies to all attribute types.

The **information** component of a value identifies the mandatory and optional context types associated with the attribute type identified by **identifier:**

```
DITContextUseInformation  ::=  SEQUENCE  {
        mandatoryContexts     [1]    SET OF CONTEXT.&id  OPTIONAL,
        optionalContexts      [2]    SET OF CONTEXT.&id  OPTIONAL }
```

SECTION 7 – SECURITY

15 Security model

15.1 Definitions

This Directory Specification makes use of the following terms defined in CCITT Rec. X.800 | ISO 7498-2:

- access control;
- authentication;
- security policy;
- confidentiality;
- integrity.

The following terms are defined in this Directory Specification:

15.1.1 access control scheme: The means by which access to Directory information and potentially to access rights themselves may be controlled.

15.1.2 protected item: An element of Directory information to which access can be separately controlled. The protected items of the Directory are entries, attributes, attribute values and names.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

15.2 Security policies

The Directory exists in an environment where various administrative authorities control access to their portion of the DIB. Such access is generally in conformance with some administration controlled security policy (see ITU-T Rec. X.509 | ISO/IEC 9594-8).

Two aspects or components of the security policy which effect access to the Directory are the authentication procedures and the access control scheme.

NOTE – Clause 16 defines two access control schemes known as Basic Access Control and Simplified Access Control, and clause 17 defines Rule-based Access Control. These schemes may be used in conjunction with local administrative controls; however, since local administrative policy has no standardized representation, it cannot be communicated in shadowed information.

15.2.1 Authentication procedures and mechanisms

Authentication procedures and mechanisms in the context of the Directory include the methods to verify and propagate where necessary:

– the identity of DSAs and Directory users;

– the identity of the origin of information received at an access point.

NOTE 1 – The administrative authority may stipulate different provisions for the authentication of administrative users as compared to provisions for the authentication of non-administrative users.

General-use authentication procedures are defined in ITU-T Rec. X.509 | ISO/IEC 9594-8 and can be used in conjunction with the access control schemes defined in this Directory Specification to enforce security policy.

NOTE 2 – Future editions of the Directory Specifications may define other access control schemes.

NOTE 3 – Local administrative policy may stipulate that authentication taking place in certain other DSAs (e.g. DSAs in other DMDs) is to be disregarded.

In general, there will be a mapping function from the authenticated identity (e.g. human user identity as authenticated by an authentication exchange) to the access control identity (e.g. the distinguished name of an entry, together with an optional unique identifier, representing the user). A particular security policy may state that the authenticated identity and the access control identity are the same.

For names in the access control identity, primary distinguished names shall be used. Similarly, where access control uses names in its specification of grants and denial, primary distinguished names shall be used.

15.2.2 Access control scheme

The definition of an access control scheme in the context of the Directory includes methods to:

– specify access control information (ACI);

– enforce access rights defined by that access control information;

– maintain access control information.

The enforcement of access rights applies to controlling access to:

– Directory information related to names;

– Directory user information;

– Directory operational information including access control information.

Administrative authorities may make use of all or parts of any standardized access control scheme in implementing their security policies, or may freely define their own schemes at their discretion.

However, administrative authorities may stipulate separate provisions for the protection of some or all of the Directory operational information. Administrative authorities are not required to provide ordinary users with the means to detect provisions for the protection of operational information.

NOTE 1 – Administrative policy may grant or deny any form of access to particular attributes (e.g. operational attributes) irrespective of access controls which may otherwise apply.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The Directory provides a means for the access control scheme in force in a particular portion of the DIB to be identified through the use of the operational attribute **accessControlScheme.** The scope of such a scheme is defined by an Access Control Specific Area (ACSA), which is a specific administrative area that is the responsibility of the corresponding Security Authority. This attribute is placed in the Administrative Entry for the corresponding Administrative Point. Only administrative entries for Access Control Specific Points are allowed to contain an **accessControlScheme** attribute.

NOTE 2 – If this operational attribute is missing with respect to access to a given entry, then the DSA shall behave as for a 1988 edition DSA (i.e. it is a local matter to determine an access control mechanism and its effect on operations, results and errors).

```
accessControlScheme  ATTRIBUTE  ::= {
        WITH SYNTAX                     OBJECT IDENTIFIER
        EQUALITY MATCHING RULE          objectIdentifierMatch
        SINGLE VALUE                    TRUE
        USAGE                           directoryOperation
        ID                              id-aca-accessControlScheme }
```

Any subentry or entry in an ACSA is permitted to contain entry ACI if and only if such ACI is permitted and consistent with the value of the **accessControlScheme** attribute of the corresponding ACSA.

15.3 Protection of Directory operations

15.3.1 Protection and Security Transformations

The directory operations (arguments, results and errors) can be protected using the **PROTECTION**, **SECURITY-TRANSFORMATION** and **PROTECTION-MAPPING** notation based upon the Generic Upper Layers Security specifications (as defined in ITU-T Rec. X.830 | ISO/IEC 11586-1).

Depending on the protection requirements, a system may support one or more of the following:

a) The signed **PROTECTION-MAPPING** as defined in (ITU-T Rec. X.830 | ISO/IEC 11586-1) to provide integrity and data origin authentication using digital signature techniques. This maps onto the **dirSignedTransformation** which results in encoding identical to the same data type used with the **SIGNED** ASN.1 construct (as defined in ITU-T Rec. X.509 | ISO/IEC 9594-8);

b) The encrypted **PROTECTION-MAPPING** to provide confidentiality (and under some algorithms and key management schemes, integrity and data origin authentication) using encryption techniques. This uses the following definitions:

```
genEncryptedTransform {KEY-INFORMATION: SupportedKIClasses }  SECURITY-TRANSFORMATION  ::=
    {
    IDENTIFIER                  { enhancedSecurity  gen-encrypted(2) }
    INITIAL-ENCODING-RULES      { joint-iso-itu-t  asn1(1)  ber(1) }

                                -- This default for initial encoding rules may be overridden
                                -- using a static protected parameter (initEncRules).

    XFORMED-DATA-TYPE           SEQUENCE {

            initEncRules        OBJECT IDENTIFIER DEFAULT { joint-iso-itu-t  asn1(1)  ber(1) },
            encAlgorithm        AlgorithmIdentifier OPTIONAL,  -- Identifies the encryption algorithm,
            keyInformation      SEQUENCE {
                kiClass         KEY-INFORMATION.&kiClass ({SupportedKIClasses}),
                keyInfo         KEY-INFORMATION.&KiType ({SupportedKIClasses} {@kiClass})
                                } OPTIONAL,

                -- Key information may assume various formats, governed by supported members
                -- of the KEY-INFORMATION information object class (defined in ITU-T
                -- Rec. X.830 | ISO/IEC 11586-1)

            encData     BIT STRING ( CONSTRAINED BY {

                -- the encData value must be generated following
                -- the procedure specified below -- })
            }
    }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Other details

Encoding process:	The input data is encrypted.
Encoding process local inputs:	Identifier of encryption algorithm, (optional) algorithm parameters, key identifier.
Decoding process:	The data is decrypted.
Decoding process local inputs:	Algorithm and key identifier if they are not conveyed as protected parameters.
Decoding process outputs:	Decrypted data.
Parameters:	Optional static protected parameters are: initial encoding rules, identifier of encryption algorithm, parameters of encryption algorithm, key information.
Transformation qualifiers:	Key identifier.
Errors:	An error condition occurs if the encryption or decryption process fails.
Security Services:	Confidentiality.

```
encrypted PROTECTION-MAPPING ::= {
    SECURITY-TRANSFORMATION {genEncryptedTransform} }
```

c) The **signedAndEncrypt PROTECTION-MAPPING** to provide a combination of the protection described above.

```
signedAndEncrypt PROTECTION-MAPPING ::= {
    SECURITY-TRANSFORMATION {signedAndEncryptedTransform} }

signedAndEncryptedTransform {KEY-INFORMATION: SupportedKIClasses}
    SECURITY-TRANSFORMATION ::= {
        IDENTIFIER                  { enhancedSecurity dir-encrypt-sign (1) }
        INITIAL-ENCODING-RULES      { joint-iso-itu-t asn1 (1) ber-derived (2) distinguished-encoding (1) }
        XFORMED-DATA-TYPE
            PROTECTED
            {
                PROTECTED
                {
                unprotectedData ABSTRACT-SYNTAX.&Type,
                signed
                },
            encrypted
            }
}
```

15.3.2 Selecting Protection to be Applied

The application of protection to directory operations is generally optional using the following notation:

```
OPTIONALLY-PROTECTED {ToBeProtected, PROTECTION-MAPPING:generalProtection} ::=
    CHOICE {
        toBeProtected    ToBeProtected,
                             -- no DIRQOP specified for operation
        signed           PROTECTED {ToBeProtected, Signed},
                             -- DIRQOP is Signed
        protected        [APPLICATION 0]
                             PROTECTED { ToBeProtected, generalProtection } }
                             -- DIRQOP is other than Signed
```

The specific protection mappings to be applied to the operations for a particular open system is defined using the following **DIRQOP** notation:

a) The protection required for all the operation argument, results and errors (excluding results/errors from chained operations – see below) over a binding is indicated at bind establishment using a **DIRQOP** specification; if no **DIRQOP** is identified at bind establishment, then the operations on that binding are unprotected.

b) In a **DIRQOP** specification, if no protection is specified for a particular operation argument, result or errors then this is unprotected.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

c) The requestor may indicate in a request alternative protection required on the result or errors of a particular operation.

d) For chained operations, unless the requestor has indicated alternative protection requirements, the protection required on the result/errors is indicated in the request to reflect the DIRQOP selected on the DUA to DSA binding for the operation being chained.

e) If a DSA cannot provide the required protection in a response or error, then it may return with unprotected information although this information may be rejected by the recipient.

A managed object for the DIRQOP is defined in ITU-T Rec. X.530 | ISO/IEC 9594-10. A default DIRQOP for a directory component can be made available using the following attribute in the entry for that directory component.

```
defaultDirQop ATTRIBUTE  ::=  {
        WITH SYNTAX                     OBJECT IDENTIFIER
        EQUALITY MATCHING RULE          objectIdentifierMatch
        USAGE                           directoryOperation
        ID                              id-at-defaultDirQop }

DIRQOP  ::=  CLASS
-- This information object class is used to define the quality of protection
-- required throughout directory operation.
-- The Quality Of Protection can be signed, encrypted, signedAndEncrypt
{
        &dirqop-Id                                  OBJECT IDENTIFIER UNIQUE,
        &dirBindError-QOP                           PROTECTION-MAPPING:protectionReqd,
        &dirErrors-QOP                              PROTECTION-MAPPING:protectionReqd,
        &dapReadArg-QOP                             PROTECTION-MAPPING:protectionReqd,
        &dapReadRes-QOP                             PROTECTION-MAPPING:protectionReqd,
        &dapCompareArg-QOP                          PROTECTION-MAPPING:protectionReqd,
        &dapCompareRes-QOP                          PROTECTION-MAPPING:protectionReqd,
        &dapListArg-QOP                             PROTECTION-MAPPING:protectionReqd,
        &dapListRes-QOP                             PROTECTION-MAPPING:protectionReqd,
        &dapSearchArg-QOP                           PROTECTION-MAPPING:protectionReqd,
        &dapSearchRes-QOP                           PROTECTION-MAPPING:protectionReqd,
        &dapAbandonArg-QOP                          PROTECTION-MAPPING:protectionReqd,
        &dapAbandonRes-QOP                          PROTECTION-MAPPING:protectionReqd,
        &dapAddEntryArg-QOP                         PROTECTION-MAPPING:protectionReqd,
        &dapAddEntryRes-QOP                         PROTECTION-MAPPING:protectionReqd,
        &dapRemoveEntryArg-QOP                      PROTECTION-MAPPING:protectionReqd,
        &dapRemoveEntryRes-QOP                      PROTECTION-MAPPING:protectionReqd,
        &dapModifyEntryArg-QOP                      PROTECTION-MAPPING:protectionReqd,
        &dapModifyEntryRes-QOP                      PROTECTION-MAPPING:protectionReqd,
        &dapModifyDNArg-QOP                         PROTECTION-MAPPING:protectionReqd,
        &dapModifyDNRes-QOP                         PROTECTION-MAPPING:protectionReqd,
        &dspChainedOp-QOP                           PROTECTION-MAPPING:protectionReqd,
        &dispShadowAgreeInfo-QOP                    PROTECTION-MAPPING:protectionReqd,
        &dispCoorShadowArg-QOP                      PROTECTION-MAPPING:protectionReqd,
        &dispCoorShadowRes-QOP                      PROTECTION-MAPPING:protectionReqd,
        &dispUpdateShadowArg-QOP                    PROTECTION-MAPPING:protectionReqd,
        &dispUpdateShadowRes-QOP                    PROTECTION-MAPPING:protectionReqd,
        &dispRequestShadowUpdateArg-QOP             PROTECTION-MAPPING:protectionReqd,
        &dispRequestShadowUpdateRes-QOP             PROTECTION-MAPPING:protectionReqd,
        &dopEstablishOpBindArg-QOP                  PROTECTION-MAPPING:protectionReqd,
        &dopEstablishOpBindRes-QOP                  PROTECTION-MAPPING:protectionReqd,
        &dopModifyOpBindArg-QOP                     PROTECTION-MAPPING:protectionReqd,
        &dopModifyOpBindRes-QOP                     PROTECTION-MAPPING:protectionReqd,
        &dopTermOpBindArg-QOP                       PROTECTION-MAPPING:protectionReqd,
        &dopTermOpBindRes-QOP                       PROTECTION-MAPPING:protectionReqd
}
WITH SYNTAX
{
        DIRQOP-ID                       &dirqop-Id
        DIRECTORYBINDERROR-QOP          &dirBindError-QOP
        DIRERRORS-QOP                   &dirErrors-QOP
        DAPREADARG-QOP                  &dapReadArg-QOP
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
    DAPREADRES-QOP                          &dapReadRes-QOP
    DAPCOMPAREARG-QOP                       &dapCompareArg-QOP
    DAPCOMPARERES-QOP                       &dapCompareRes-QOP
    DAPLISTARG-QOP                          &dapListArg-QOP
    DAPLISTRES-QOP                          &dapListRes-QOP
    DAPSEARCHARG-QOP                        &dapSearchArg-QOP
    DAPSEARCHRES-QOP                        &dapSearchRes-QOP
    DAPABANDONARG-QOP                       &dapAbandonArg-QOP
    DAPABANDONRES-QOP                       &dapAbandonRes-QOP
    DAPADDENTRYARG-QOP                      &dapAddEntryArg-QOP
    DAPADDENTRYRES-QOP                      &dapAddEntryRes-QOP
    DAPREMOVEENTRYARG-QOP                   &dapRemoveEntryArg-QOP
    DAPREMOVEENTRYRES-QOP                   &dapRemoveEntryRes-QOP
    DAPMODIFYENTRYARG-QOP                   &dapModifyEntryArg-QOP
    DAPMODIFYENTRYRES-QOP                   &dapModifyEntryRes-QOP
    DAPMODIFYDNARG-QOP                      &dapModifyDNArg-QOP
    DAPMODIFYDNRES-QOP                      &dapModifyDNRes-QOP
    DSPCHAINEDOP-QOP                        &dspChainedOp-QOP
    DISPSHADOWAGREEINFO-QOP                 &dispShadowAgreeInfo-QOP
    DISPCOORSHADOWARG-QOP                   &dispCoorShadowArg-QOP
    DISPCOORSHADOWRES-QOP                   &dispCoorShadowRes-QOP
    DISPUPDATESHADOWARG-QOP                 &dispUpdateShadowArg-QOP
    DISPUPDATESHADOWRES-QOP                 &dispUpdateShadowRes-QOP
    DISPREQUESTSHADOWUPDATEARG-QOP          &dispRequestShadowUpdateArg-QOP
    DISPREQUESTSHADOWUPDATERES-QOP          &dispRequestShadowUpdateRes-QOP
    DOPESTABLISHOPBINDARG-QOP               &dopEstablishOpBindArg-QOP
    DOPESTABLISHOPBINDRES-QOP               &dopEstablishOpBindRes-QOP
    DOPMODIFYOPBINDARG-QOP                  &dopModifyOpBindArg-QOP
    DOPMODIFYOPBINDRES-QOP                  &dopModifyOpBindRes-QOP
    DOPTERMINATEOPBINDARG-QOP               &dopTermOpBindArg-QOP
    DOPTERMINATEOPBINDRES-QOP               &dopTermOpBindRes-QOP
}
```

16 Basic Access Control

16.1 Scope and application

This clause defines one specific access control scheme (of possibly many) for the Directory. The access control scheme defined herein is identified with the **accessControlScheme** operational attribute by giving it the value **basic-access-control**. Subclause 15.2.2 describes which entries contain the **accessControlScheme** operational attribute.

NOTE – An access control scheme known as "Simplified Access Control" is specified in 16.9. It is defined as a subset of the Basic Access Control sheme. When Simplified Access Control is used, the **accessControlScheme** operational attribute shall have the value **simplified-access-control.** Additional access control shemes known as "Rule-based Access Control" are specified in clause 17.

The scheme defined here is only concerned with providing means of controlling access to the Directory information within the DIB (potentially including tree structure and access control information). It does not address controlling access for the purpose of communication with a DSA application-entity. Control of access to information means the prevention of unauthorized detection, disclosure, or modification of that information.

16.2 Basic Access Control model

The Basic Access Control model for the Directory defines, for every Directory operation, one or more points at which access control decisions take place. Each access control decision involves:

- that element of Directory information being accessed, called the *protected item*;
- the user requesting the operation, called the *requestor*;
- a particular right necessary to complete a portion of the operation, called the *permission*;
- one or more operational attributes that collectively contain the security policy governing access to that item, called *ACI items*.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Thus, the basic access control model defines:

- the protected items;
- the user classes;
- the permission categories required to perform each Directory operation;
- the scope of application and syntax of ACI items;
- the basic algorithm, called the Access Control Decision Function (ACDF), used to decide whether a particular requestor has a particular permission by virtue of applicable ACI items.

16.2.1 Protected items

A protected item is an element of Directory information to which access can be separately controlled. The protected items of the Directory are entries, attributes, attribute values and names. For convenience in specifying access control policies, Basic Access Control provides the means to identify collections of related items, such as attributes in an entry or all attribute values of a given attribute, and to specify a common protection for them.

16.2.2 Access control permissions and their scope

Access is controlled by granting or denying permissions. The permission categories are described in 16.2.3 and 16.2.4.

The scope of access controls can be a single entry or a collection of entries that are logically related by being within the scope of a subentry for a particular administrative point.

Permission categories are generally independent. Since all Directory entries have a relative position within the DIT, access to user and operational information always involves some form of access to DIT related information. Thus, there are two main forms of access control decision associated with a Directory operation: access to entries as named objects (referred to as *entry access*); and access to attributes containing user and operational information (referred to as *attribute access*). For many Directory operations, both forms of permission are required. In addition, where applicable, separate permissions control the name or error type returned. Some important aspects of permissions categories, forms of access, and access control decision making are as follows:

a) To perform Directory operations on entire entries (e.g. read an entry or add an entry), it is usually necessary for permission to be granted with respect to the attributes and values contained within that entry. Exceptions are permissions controlling entry renaming and removal: in neither case is attribute or attribute value permissions taken into account.

b) To perform Directory operations that require access to attributes or attribute values, it is necessary to have entry access permission to the entry or entries that contain those attributes or values.

NOTE 1 – The removal of an entry or of an attribute does not require access to the contents of the entry or of the attribute.

c) The decision whether or not to permit entry access is strictly determined by the position of the entry in the DIT, in terms of its distinguished name, and is independent of how the Directory locates that entry.

d) A design principle of Basic Access Control is that access may be allowed only when there is an explicitly provided grant present in the access control information used by the Directory to make the access control decision. Granting one form of access (e.g. entry access) never automatically or implicitly grants the other form (e.g. attribute access). In order to administer meaningful Directory access control policies, it is thus usually necessary to explicitly set access policy for both forms of access.

NOTE 2 – Certain combinations of grants or denials are illogical, but it is the responsibility of users, rather than the Directory, to ensure that such combinations are absent.

NOTE 3 – Consistent with the above design principle, granting or denying permissions for an attribute value does not automatically control access to the related attribute. Moreover, in order to access an attribute value(s) in the course of a Directory interrogation operation, a user must be granted access to both the attribute type and its value(s).

e) The only default access decision provided in the model is to deny access in the absence of explicit access control information that grants access.

f) A denial specified in access control information always overrides a grant, all other factors being equal.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

g) A particular DSA may not have the access control information governing the Directory data it caches. Security Administrators should be aware that a DSA with the capability of caching may pose a significant security risk to other DSAs, in that it may reveal information to unauthorized users.

h) For the purposes of interrogation, collective attributes that are associated with an entry are protected precisely as if they were attributes part of the entry.

NOTE 4 – For the purposes of modification, collective attributes are associated with the subentry that holds them, not with entries within the scope of the subentry. Modify-related access controls are therefore not relevant to collective attributes, except when they apply to the collective attribute and its values within the subentry.

16.2.3 Permission categories for entry access

The permission categories used to control entry access are *Read*, *Browse*, *Add*, *Remove*, *Modify*, *Rename*, *DiscloseOnError*, *Export*, and *Import* and *ReturnDN*. Their use is described in more detail in ITU-T Rec. X.511 | ISO/IEC 9594-3. Annex J provides an overview of their meaning in general situations. This subclause introduces the categories by briefly indicating the intent associated with the granting of each. The actual influence of a particular granted permission on access control decisions must, however, be understood in the full context of the ACDF and access control decision points for each Directory operation.

a) *Read*, if granted, permits read access for Directory operations which specifically name an entry (i.e. as opposed to the List and Search operations) and provides visibility to the information contained in the entry to which it applies.

b) *Browse*, if granted, permits entries to be accessed using Directory operations which do not explicitly provide the name of the entry.

c) *Add*, if granted, permits creation of an entry in the DIT subject to controls on all attributes and attribute values to be placed in the new entry at time of creation.

NOTE 1 – In order to add an entry, permission must also be granted to add at least the mandatory attributes and their values.

NOTE 2 – There is no specific "add subordinate permission". Permission to add an entry is controlled using **prescriptiveACI** operational attributes as described in 16.3.

d) *Remove*, if granted, permits the entry to be removed from the DIT regardless of controls on attributes or attribute values within the entry.

e) *Modify*, if granted, permits the information contained within an entry to be modified.

NOTE 3 – In order to modify information contained within an entry other than the distinguished name attribute values, appropriate attribute and value permissions must also be granted.

f) Granting *Rename* is necessary for an entry to be renamed with a new RDN, taking into account the consequential changes to the distinguished names of subordinate entries, if any; if the name of the superior is unchanged, the grant is sufficient.

NOTE 4 – In order to rename an entry, there are no prerequisite permissions to contained attributes or values, including the RDN attributes; this is true even when the operation causes new attribute values to be added or removed as a result of the changes of RDN.

g) *DiscloseOnError*, if granted, permits the name of an entry to be disclosed in an error (or empty) result.

h) *Export*, if granted, permits an entry and its subordinates (if any) to be exported; that is, removed from the current location and placed in a new location subject to the granting of suitable permissions at the destination. If the last RDN is changed, *Rename* is also required at the current location.

NOTE 5 – In order to export an entry or its subordinates, there are no prerequisite permissions to contained attributes or values, including the RDN attributes; this is true even when the operation causes attribute values to be added or removed as a result of the changes of RDN.

i) *Import*, if granted, permits an entry and its subordinates, if any, to be imported; that is, removed from some other location and placed at the location to which the permission applies (subject to the granting of suitable permissions at the source location).

NOTE 6 – In order to import an entry or its subordinates, there are no prerequisite permissions to contained attributes or values, including the RDN attributes; this is true even when the operation causes attribute values to be added or removed as a result of the changes of RDN.

j) *ReturnDN*, if granted, allows the distinguished name of the entry to be disclosed in an operation result.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

16.2.4 Permission categories for attribute and attribute value access

The permission categories used to control attribute and attribute value access are *Compare*, *Read*, *FilterMatch*, *Add*, *Remove*, and *DiscloseOnError*. They are described in more detail in ITU-T Rec. X.511 | ISO/IEC 9594-3. Annex J provides an overview of their meaning in general situations. This subclause introduces the categories by briefly indicating the intent associated with the granting of each. The actual influence of a particular granted permission on access control decisions must, however, be understood in the full context of the ACDF and access control decision points for each Directory operation.

a) *Compare*, if granted, permits attributes and values to be used in a compare operation.

b) *Read*, if granted, permits attributes and values to be returned as entry information in a read or search access operation.

c) *FilterMatch*, if granted, permits evaluation of a filter within a search criterion.

d) *Add*, if granted for an attribute, permits adding an attribute subject to being able to add all specified attribute values. If granted for an attribute value, it permits adding a value to an existing attribute.

e) *Remove*, if granted for an attribute, permits removing an attribute complete with all of its values. If granted for an attribute value, it permits the attribute value to be removed from an existing attribute.

f) *DiscloseOnError*, if granted for an attribute, permits the presence of the attribute to be disclosed by an attribute or security error. If granted for an attribute value, it permits the presence of the attribute value to be disclosed by an attribute or security error.

16.3 Access control administrative areas

The DIT is partitioned into subtrees termed autonomous administrative areas, each of which is under the administrative authority of a single Domain Management Organization. It may be further partitioned into subtrees termed specific administrative areas for the purposes of specific aspects of administration; alternatively, the whole of an autonomous administrative area may comprise a single specific administrative area. Each such specific administrative area is the responsibility of a corresponding specific administrative authority. A particular administrative area may be shared by several specific administrative authorities. See clause 10.

16.3.1 Access control areas and Directory Access Control Domains

In the case of access control, the specific administrative authority is a Security Authority, and the specific administrative area is termed an Access Control Specific Area (ACSA). The root of the ACSA is termed an Access Control Specific Point. Each Access Control Specific Point is represented in the DIT by an Administrative Entry which includes **access-control-specific-area** as a value of its **administrativeRole** operational attribute; it has (potentially) one or more subentries which contain access control information. Similarly, each Access Control Inner Point is represented in the DIT by an Administrative Entry which contains **access-control-inner-area** as a value of its **administrativeRole** operational attribute; it also has (potentially) one or more subentries which contain access control information. Each such administrative entry which has a subentry containing prescriptive ACI information has **basic-access-control**, **simplified-access-control**, or other relevant value as a value of its **accessControlScheme** operational attribute. Each subentry that is an Access Control Specific Point and which contains access control information, has **accessControlSubentry** as a value of its object-class attribute. An administrative entry and its subentries may hold operational attributes (such as access control information) which relate, respectively, to the administrative point (and possibly its subentries) and to collections of entries (within the administrative area) defined by the subentry **subtreeSpecification**.

The **accessControlScheme** attribute shall be present if and only if the holding administrative entry is an access control specific entry. An administrative entry can never be both an access control specific and an access control inner entry; corresponding values can therefore never be present simultaneously in the **administrativeRole** attribute.

The scope of a subentry that contains access control information, as defined by its **subtreeSpecification** (which may include subtree refinements), is termed a Directory Access Control Domain (DACD).

NOTE – A DACD can contain zero entries, and can encompass entries that have not yet been added to the DIT.

The Security Authority may permit an Access Control Specific Area to be partitioned into subtrees termed inner (administrative) areas. Each such inner area is termed an Access Control Inner Area (ACIA) with **access-control-inner-area** as the value of the **administrativeRole** operational attribute. Each subentry of the corresponding administrative point that contains prescriptive ACI has, as before, an **accessControlSubentry** value within its object class attribute.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The scope (**subtreeSpecification**) specified in a subentry within an ACIA is also a DACD and contains entries inside the associated Access Control Inner Area.

ACIAs allow a degree of delegation of access control authority within the ACSA. The authority for the ACSA still retains authority within the ACIA since the ACI in the subentries of the ACSA's administrative point apply as well as the ACI in the subentries of the relevant ACIAs (subclause 16.6 explains how the ACSA controls authority).

In summary, in evaluating access controls, the type of access control scheme (e.g. Basic Access Control) is indicated by the **accessControlScheme** attribute value of the relevant access control specific entry; the role of each relevant administrative entry within the ACSA is indicated by its **administrativeRole** attribute values; the presence of prescriptive access control in a particular subentry is indicated by an **accessControlSubentry** value in its object class attribute.

Subentries, like other entries, can hold an **entryACI** attribute for protection of its own contents.

16.3.2 Associating controls with administrative areas

Access to a given entry is (potentially) controlled by the totality of superior access control administrative points (both inner and specific) up to and including the first non-inner access control administrative point or Autonomous Administrative Point encountered moving up the DIT from the entry towards the root. Access Control Specific Points superior to this access control administrative point have no effect on access control to the given entry.

NOTE 1 – An Autonomous Administrative Point is considered implicitly to be an Access Control Specific Point for the purpose of this description, even if it is not associated with any prescriptive controls.

Some important points regarding the association between access controls and administrative areas are:

a) Access controls for Directory information may apply to only selected entries, or may have scope extending throughout portions of the DIB that are logically related by a common security policy and a common Access Control administration.

b) Access control may be imposed on entries within ACSAs or within ACIAs by placing **prescriptiveACI** attributes (see 16.5) within one or more subentries of the corresponding Access Control Administrative Entry, with scope defined by an appropriate s**ubtreeSpecification**.

NOTE 2 – **prescriptiveACI** attributes are not collective attributes. There are a number of significant differences between **prescriptiveACI** and collective attributes:

- although a **prescriptiveACI** attribute may affect access control decisions for each entry within the scope of the subentry that holds it, the **prescriptiveACI** attribute is not considered to supply accessible information to any such entry or to be in any sense a part of such an entry;
- **prescriptiveACI** attributes are associated with the access control aspects of administration, and are associated with Access Control Specific and Inner Points, not with entry-collection administrative points;
- The purpose of a **prescriptiveACI** attribute is to express a policy that influences across a defined set of entries, while the purpose of a collective attribute is to provide information that associates a user-accessible set of attributes within a defined set of entries;
- **prescriptiveACI** attributes represent policy information that will, in general, not be widely accessible by ordinary users. Administrative users who need to access **prescriptiveACI** information can access them as operational attributes within subentries.

c) A **prescriptiveACI** operational attribute contains **ACIItems** (see 16.4.1) common to all entries within the scope of the subentry, i.e. DACD, in which the **prescriptiveACI** occurs. A DACD normally contains entries inside the associated Access Control Specific Area (but can contain no entries at all).

d) Although particular **ACIItems** may specify attributes or values as protected items, **ACIItems** are logically associated with entries. The particular set of **ACIItems** associated with an entry is a combination of:

- **ACIItems** that apply to that particular entry, specified as values of the **entryACI** operational attribute, if present (see 16.5.2);
- **ACIItems** from **prescriptiveACI** operational attributes applicable to the entry by virtue of being placed in subentries of administrative entries whose scope includes the particular entry (see 16.5.1).

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

e) Each entry (controlled by **entryACI** and/or **prescriptiveACI**) necessarily falls within one and only one ACSA. Each such entry may also fall within one or more ACIAs nested inside the ACSA containing the entry. The **prescriptiveACI** that potentially affects the outcome of access control decisions for a given entry are located within subentries (of the administrative entry) for the ACSA and for each ACIA containing the entry. Other subentries cannot affect access control decisions regarding that entry.

f) If an entry is within the scope of more than one DACD, the complete set of **ACIItems** that may potentially affect access control decisions regarding that entry includes all **prescriptiveACI** item attributes of those DACDs, in addition to any **entryACI** attributes in the entry itself. An example is shown in Figure 11. The effective access control at entry E1 is a combination of the **prescriptiveACI** for DACD1, DACD2, DACD3, and **entryACI** (if present) in entry E1. The effective access control at entry E2 is a combination of the **prescriptiveACI** for DACD1 and DACD3, and **entryACI** (if present) in entry E2.

NOTE 3 – Protection of access control information is described in 16.6.

g) The **subtreeSpecification** attribute in each subentry defines a collection of entries within an administrative area. Since a **subtreeSpecification** may define a subtree refinement, DACDs may arbitrarily overlap within the intersection of their respective administrative areas. For simplicity, Figure 11 does not show administrative points, subentries, or administrative areas; however, it may be considered as three DACDs in the same ACSA with each DACD corresponding to a single subentry of the administrative point for that ACSA (and there are no ACIAs). Alternatively, Figure 11 may be considered in the context of a single ACSA containing a single ACIA where DACD1 is congruent to the ACSA and DACD3 is congruent to the ACIA (DACD1 and DACD2 would correspond to subentries of the ACSA administrative point and DACD3 would correspond to a subentry of the ACIA administrative point). An administrative area is congruent to a DACD when the collection of entries in the DACD is the same as the collection of entries in the implicitly defined subtree corresponding to the administrative area. See the example in Annex K for figures depicting the relationship between administrative entries, administrative areas, subentries and DACDs.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

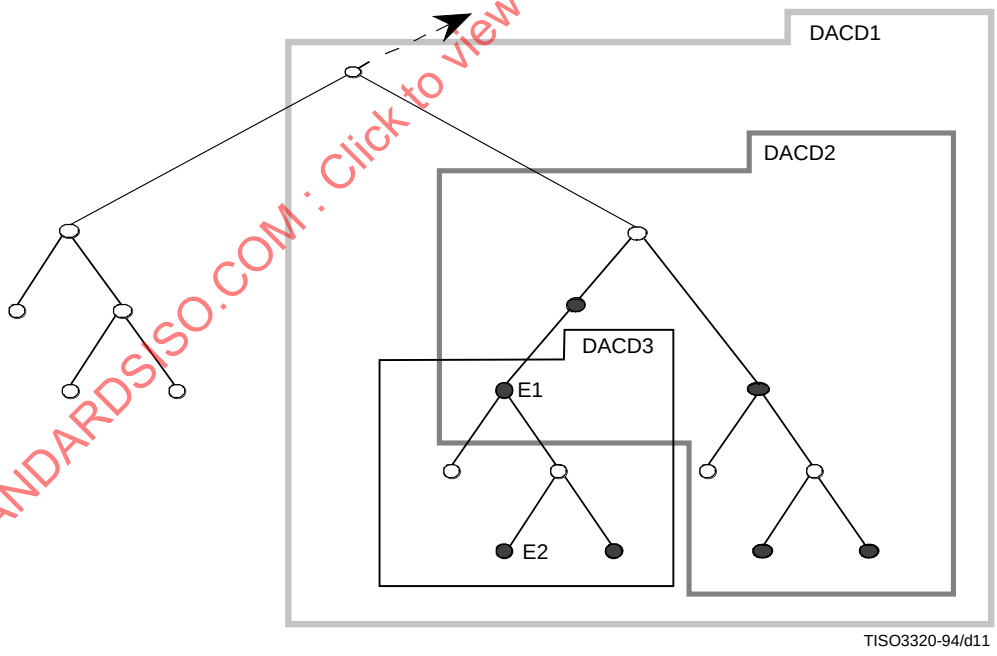


Figure 11 – Effective Access Control using DACDs

16.4 Representation of Access Control Information

16.4.1 ASN.1 for Access Control Information

Access Control Information is represented as a set of **ACIItem**s, where each **ACIItem** grants or denies permissions in regard to certain specified users and protected items.

In ASN.1 the information is expressed as:

```
ACIItem ::= SEQUENCE {
        identificationTag               DirectoryString { ub-tag },
        precedence                      Precedence,
        authenticationLevel             AuthenticationLevel,
        itemOrUserFirst                 CHOICE {
                itemFirst               [0]  SEQUENCE {
                        protectedItems          ProtectedItems,
                        itemPermissions         SET OF ItemPermission },
                userFirst               [1]  SEQUENCE {
                        userClasses             UserClasses,
                        userPermissions         SET OF UserPermission } } }

Precedence ::= INTEGER (0..255)

ProtectedItems ::= SEQUENCE {
        entry                           [0]  NULL                           OPTIONAL,
        allUserAttributeTypes           [1]  NULL                           OPTIONAL,
        attributeType                   [2]  SET OF AttributeType           OPTIONAL,
        allAttributeValues              [3]  SET OF AttributeType           OPTIONAL,
        allUserAttributeTypesAndValues  [4]  NULL                           OPTIONAL,
        attributeValue                  [5]  SET OF AttributeTypeAndValue   OPTIONAL,
        selfValue                       [6]  SET OF AttributeType           OPTIONAL,
        rangeOfValues                   [7]  Filter                         OPTIONAL,
        maxValueCount                   [8]  SET OF MaxValueCount           OPTIONAL,
        maxImmSub                       [9]  INTEGER                        OPTIONAL,
        restrictedBy                    [10] SET OF RestrictedValue         OPTIONAL,
        contexts                        [11] SET OF ContextAssertion        OPTIONAL }

MaxValueCount ::= SEQUENCE {
        type            AttributeType,
        maxCount        INTEGER }

RestrictedValue ::= SEQUENCE {
        type            AttributeType,
        valuesIn        AttributeType }

UserClasses ::= SEQUENCE {
        allUsers        [0] NULL                                OPTIONAL,
        thisEntry       [1] NULL                                OPTIONAL,
        name            [2] SET OF NameAndOptionalUID           OPTIONAL,
        userGroup       [3] SET OF NameAndOptionalUID           OPTIONAL,
                            -- dn component must be the name of an
                            -- entry of GroupOfUniqueNames
        subtree         [4] SET OF SubtreeSpecification         OPTIONAL }

ItemPermission ::= SEQUENCE {
        precedence              Precedence OPTIONAL,
                        -- defaults to precedence in ACIItem --
        userClasses             UserClasses,
        grantsAndDenials        GrantsAndDenials }

UserPermission ::= SEQUENCE {
        precedence              Precedence OPTIONAL,
                        -- defaults to precedence in ACIItem
        protectedItems          ProtectedItems,
        grantsAndDenials        GrantsAndDenials }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
AuthenticationLevel ::= CHOICE {
        basicLevelsSEQUENCE {
                level                   ENUMERATED { none (0), simple (1), strong (2) },
                localQualifier          INTEGER OPTIONAL },
        other                   EXTERNAL }

GrantsAndDenials ::= BIT STRING {
        -- permissions that may be used in conjunction
        -- with any component of ProtectedItems

        grantAdd                (0),
        denyAdd                 (1),
        grantDiscloseOnError    (2),
        denyDiscloseOnError     (3),
        grantRead               (4),
        denyRead                (5),
        grantRemove             (6),
        denyRemove              (7),
        -- permissions that may be used only in conjunction
        -- with the entry component
        grantBrowse             (8),
        denyBrowse              (9),
        grantExport             (10),
        denyExport              (11),
        grantImport             (12),
        denyImport              (13),
        grantModify             (14),
        denyModify              (15),
        grantRename             (16),
        denyRename              (17),
        grantReturnDN           (18),
        denyReturnDN            (19),
        -- permissions that may be used in conjunction
        -- with any component, except entry, of ProtectedItems
        grantCompare            (20),
        denyCompare             (21),
        grantFilterMatch        (22),
        denyFilterMatch         (23) }

AttributeTypeAndValue ::= SEQUENCE {
        type            ATTRIBUTE.&id ({SupportedAttributes}),
        value           ATTRIBUTE.&Type({SupportedAttributes}{@type}) }
```

16.4.2 Description of ACIItem Parameters

16.4.2.1 IdentificationTag

IdentificationTag is used to identify a particular **ACIItem**. This is used to discriminate among individual **ACIItem**s for the purposes of protection, management and administration.

16.4.2.2 Precedence

Precedence is used to control the relative order in which **ACIItem**s are considered during the course of making an access control decision in accordance with 16.8. **ACIItem**s having higher precedence values may prevail over others with lower precedence values, other factors being equal. Precedence values are integers and are compared as such.

Precedence can be used by a superior authority within the Security Authority to permit partial delegation of access control policy setting within an ACSA. This can be achieved by the superior authority setting a general policy at a high precedence and authorizing users representing the subordinate authority (e.g. associated with an ACIA) to create and modify ACI with a lower precedence, in order to tailor the general policy for specific purposes. The partial delegation thus requires the means for the superior authority to limit the maximum precedence which the subordinate authority can assign to ACI under its control.

Basic Access Control does not specify or describe how to limit the maximum precedence that can be used by a subordinate authority. This must be done by local means.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

16.4.2.3 AuthenticationLevel

AuthenticationLevel defines the minimum requestor authentication level required for this **ACIItem**. It has two forms:

- **basicLevels** which indicates the level of authentication, optionally qualified by positive or negative integer **localQualifier**;
- **other**: an externally defined measure.

When **basicLevels** is used, an **AuthenticationLevel** consisting of a **level** and optional **localQualifier** shall be assigned to the requestor by the DSA according to local policy. For a requestor's authentication level to exceed a minimum requirement, the requestor's **level** must meet or exceed that specified in the **ACIItem**, and in addition the requestor's **localQualifier** must be arithmetically greater than or equal to that of the **ACIItem**. Strong authentication of the requestor is considered to exceed a requirement for simple or no authentication, and simple authentication exceeds a requirement for no authentication. For access control purposes, the "simple" authentication level requires a password; the case of identification only, with no password supplied, is considered "none". If a **localQualifier** is not specified in the **ACIItem**, then the requestor need not have a corresponding value (if such a value is present it is ignored).

When **other** is used, an appropriate **AuthenticationLevel** shall be assigned to the requestor by the DSA according to local policy. The form of this **AuthenticationLevel** and the method by which it is compared with the **AuthenticationLevel** in the ACI is a local matter.

NOTE 1 – An authentication level associated with an explicit denial indicates the minimum level to which a requestor must be authenticated in order not to be denied access. For example, an **ACIItem** that denies access to a particular user class and requires strong authentication will deny access to all requestors who cannot prove, by means of a strongly authenticated identity, that they are not in that user class.

NOTE 2 – The DSA may base authentication level on factors other than values received in protocol exchanges.

16.4.2.4 itemFirst and userFirst Parameters

Each **ACIItem** contains a choice of **itemFirst** or **userFirst**. The choice allows grouping of permissions depending on whether they are most conveniently grouped by user classes or by protected items. **itemFirst** and **userFirst** are equivalent in the sense that they capture the same access control information; however, they organize that information differently. The choice between them is based on administrative convenience. The parameters used in **itemFirst** or **userFirst** are described below.

a) **ProtectedItems** define the items to which the specified access controls apply. It is defined as a set selected from the following:

- **entry** means the entry contents as a whole and does not necessarily include the information in the entry.
- **allUserAttributeTypes** means all user attribute type information associated with the entry, but not values associated with those attributes.
- **allUserAttributeTypesAndValues** means all user attribute information associated with the entry, including all values of all user attributes.
- **attributeType** means attribute type information pertaining to specific attributes but not values associated with the type.

  allAttributeValues means all attribute value information pertaining to specific attributes.
- **attributeValue** means a specific value of specific attributes.
- **selfValue** means the attribute value assertion corresponding to the current requestor. The protected item **selfValue** applies only when the access controls are to be applied with respect to a specific authenticated user. It can only apply in the specific case where the attribute specified is of **DistinguishedName** or **uniqueMember** syntax and the attribute value within the specified attribute matches the distinguished name of the originator of the operation.

  NOTE 1 – **allUserAttributeTypes** and **allUserAttributeTypesAndValues** do not include operational attributes, which should be specified on a per attribute basis, using attributeType, allAttributeValues or **attributeValue**.
- **rangeOfValues** means any attribute value which matches the specified filter, i.e. for which the specified filter evaluated on that attribute value would return TRUE.

  NOTE 2 – The filter is not evaluated on any entries in the DIB; it is evaluated using the semantics defined in 7.8 of ITU-T Rec. X.511 | ISO/IEC 9594-3, operating on a fictitious entry that contains only the single attribute value which is the protected item.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The remaining items provide constraints that may disable the granting of certain permissions to protected items in the same SEQUENCE:

– **maxValueCount** restricts the maximum number of attribute values allowed for a specified attribute type. It is examined if the protected item is an attribute value of the specified type and the permission sought is add. Values of that attribute in the entry are counted without regard to context or access control and as though the operation which adds the values were successful. If the number of values in the attribute exceeds maxCount, the ACI item is treated as not granting add access.

– **maxImmSub** restricts the maximum number of immediate subordinates of the superior entry to an entry being added or imported. It is examined if the protected item is an entry, the permission sought is add or import, and the immediate superior entry is in the same DSA as the entry being added or imported. Immediate subordinates of the superior entry are counted without regard to context or access control as though the entry addition or importing were successful. If the number of subordinates exceeds maxImmSub, the ACI item is treated as not granting add or import access.

– **restrictedBy** restricts values added to the attribute type to being values that are already present in the same entry as values of the attribute valuesIn. It is examined if the protected item is an attribute value of the specified type and the permission sought is add. Values of the valuesIn attribute are checked without regard to context or access control and as though the operation which adds the values were successful. If the value to be added is not present in valuesIn the ACI item is treated as not granting add access.

– **contexts** restricts values added to the entry to having context lists that satisfy all of the context assertions in **contexts**. It is examined if the protected item is an attribute value and the permission sought is *add*. If the value to be added does not satisfy the context assertions the ACI item is treated as not granting *add* access; if it does satisfy all of them, the ACI item is treated as not denying *add* access.

NOTE 3 – This is only relevant when the permission sought is *add*, and all context assertions shall be satisfied. It does not provide for general use of contexts to differentiate protected items for other permissions.

b) **UserClasses** defines a set of zero or more users the permissions apply to. The set of users is selected from the following:

– **allUsers** means every directory user (with possible requirements for **authenticationLlevel**).

– **thisEntry** means the user with the same distinguished name as the entry being accessed.

– **name** is the user with the specified distinguished name (with an optional unique identifier).

– **userGroup** is the set of users who are members of the **groupOfUniqueNames** entry, identified by the specified distinguished name (with an optional unique identifier). Members of a group of unique names are treated as individual object names, and not as the names of other groups of unique names. How group membership is determined is described in 16.4.2.5.

– **subtree** is the set of users whose distinguished names fall within the definition of the (unrefined) subtree.

Names used to specify a user, group or subtree shall be primary distinguished names. Context and alternative distinguished values shall not be included. The access control decision function is not required to determine the primary distinguished name for alternative names with which it is supplied.

NOTE 4 – This means that if a requestor has supplied an alternative name that has not been subsequently resolved by the Directory to the primary distinguished name, access control based on primary distinguished names may fail to recognize the requestor as belonging to the user class granted or denied access.

c) **SubtreeSpecification** is used to specify a subtree relative to the root entry named in **base**. The **base** represents the distinguished name of the root of subtree. The subtree extends to the leaves of the DIT unless otherwise specified in **chop**. The use of a **specificationFilter** component is not permitted; if present, it shall be ignored.

NOTE 5 – **SubtreeSpecification** does not allow subtree refinement because a refinement might require a DSA to use a distributed operation in order to determine if a given user is in a particular user class. Basic Access Control is designed to avoid remote operations in the course of making an access control decision. Membership in a subtree whose definition includes only **base** and **chop** can be evaluated locally, whereas membership in a subtree definition using **specificationFilter** can only be evaluated by obtaining information from the user's entry which is potentially in another DSA.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

d) **ItemPermission** contains a collection of users and their permissions with respect to **ProtectedItems** within an **itemFirst** specification. The permissions are specified in **grantsAndDenials** as discussed in item f) of this subclause. Each of the permissions specified in **grantsAndDenials** is considered to have the precedence level specified in **precedence** for the purpose of evaluating access control information as discussed in 16.8. If **precedence** is omitted within **ItemPermission,** then precedence is taken from the **precedence** specified for the **ACIItem** (see 16.4.2.2).

e) **UserPermission** contains a collection of protected items and the associated permissions with respect to **userClasses** within a **userFirst** specification. The protected items are specified in **protectedItems** as discussed in 16.4.2. The associated permissions are specified in **grantsAndDenials** as discussed in item f) of this subclause. Each of the permissions specified in **grantsAndDenials** is considered to have the precedence level specified in **precedence** for the purpose of evaluating access control information as discussed in 16.8. If **precedence** is omitted within **UserPermission**, the precedence is taken from the **precedence** specified for the **ACIItem** (see 16.4.2.2).

f) **GrantsAndDenials** specify the access rights that are granted or denied in the **ACIItem** specification. The precise semantics of these permissions with respect to each protected item is discussed in ITU-T Rec. X.511 | ISO/IEC 9594-3.

g) **UniqueIdentifier** may be used by the authentication mechanism to distinguish between instances of distinguished name reuse. The value of the unique identifier is assigned by the authentication authority according to its policy and is provided by the authenticating DSA. If this field is present, then for an accessing user to match the **name** user class of an **ACIItem** that grants permissions, in addition to the requirement that the user's distinguished name match the specified distinguished name, the authentication of the user must yield an associated unique identifier, and that value must match for equality with the specified value.

NOTE 6 – When authentication is based on supplied **SecurityParameters**, the unique identifier associated with the user may be taken from the **subjectUniqueIdentifier** field of the sender's **Certificate** in the optional **CertificationPath**.

16.4.2.5 Determining group membership

Determining whether a given requestor is a group member requires checking two criteria. Also, the determination may be constrained if the group definition is not known locally. The criteria for membership and the treatment of non-local groups are discussed below.

a) A DSA is not required to perform a remote operation to determine whether the requestor belongs to a particular group for the purposes of Basic Access Control. If membership in the group cannot be evaluated, the DSA shall assume that the requestor does not belong to the group if the ACI item grants the permission sought, and does belong to the group if it denies the permission sought.

NOTE 1 – Access control administrators must beware of basing access controls on membership of non-locally available groups or groups which are available only through replication (and which may, therefore, be out of date).

NOTE 2 – For performance reasons, it is usually impractical to retrieve group membership from remote DSAs as part of the evaluation of access controls. However, in certain circumstances it may be practical, and a DSA is permitted, for example, to perform remote operations to obtain or refresh a local copy of a group entry or use the Compare operation to check membership prior to applying this clause.

b) In order to determine whether the requestor is a member of a **userGroup** user class, the following criteria apply:

– The entry named by the **userGroup** specification must be an instance of the object class **groupOfNames** or **groupOfUniqueNames**.

– The name of the requestor must be a value of the **member** or **uniqueMember** attribute of that entry.

NOTE 3 – Values of the **member** or **uniqueMember** attribute that do not match the name of the requestor are ignored, even if they represent the names of groups of which the originator could be found to be a member. Hence, nested groups are not supported when evaluating access controls.

NOTE 4 – Names used in **member** or **uniqueMember** shall be primary distinguished names. Context, and alternative values with context, shall not be included.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

16.5 The ACI operational attributes

Access control information is stored in the Directory as an operational attribute of entries and subentries. The operational attribute is multi-valued, which allows ACI to be represented as a set of **ACIItem**s (defined in 16.4).

16.5.1 Prescriptive access control information

A Prescriptive ACI attribute is defined as an operational attribute of a subentry. It contains access control information applicable to entries within that subentry's scope:

```
prescriptiveACI  ATTRIBUTE  ::= {
        WITH SYNTAX                     ACIItem
        EQUALITY MATCHING RULE          directoryStringFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-aca-prescriptiveACI }
```

16.5.2 Entry access control information

An Entry ACI attribute is defined as operational attributes of an entry. It contains access control information applicable to the entry in which it appears, and that entry's contents:

```
entryACI  ATTRIBUTE  ::= {
        WITH SYNTAX                     ACIItem
        EQUALITY MATCHING RULE          directoryStringFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-aca-entryACI }
```

16.5.3 SubentryACI

Subentry ACI attributes are defined as operational attributes of administrative entries, and provide access control information that applies to each of the subentries of the corresponding administrative point. Prescriptive ACI within the subentries of a particular administrative point never applies to the same or any other subentry of that administrative point, but can be applicable to the subentries of subordinate administrative points. Subentry ACI attributes are contained only in administrative points and do not affect any element of the DIT other than immediately subordinate subentries.

In evaluating access control for a specific subentry, the ACI that must be considered is:

- the **entryACI** within the subentry itself (if any);
- the **subentryACI** within the associated administrative entry (if any);
- **prescriptiveACI** associated with other relevant administrative points within the same access control specific area (if any).

```
subentryACI  ATTRIBUTE  ::= {
        WITH SYNTAX                     ACIItem
        EQUALITY MATCHING RULE          directoryStringFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-aca-subentryACI }
```

16.6 Protecting the ACI

ACI operational attributes may be subjected to the same protection mechanisms as ordinary attributes. Some important related points are:

a) The **identificationTag** provides an identifier for each **ACIItem**. This tag can be used to remove a specific **ACIItem** value, or to protect it by prescriptive or entry ACI.

NOTE 1 – Directory rules ensure that only one **ACIItem** per access control attribute possesses any specific **identificationTag** value.

b) The creation of subentries for an Administrative Entry may be access controlled by means of the **subentryACI** operational attribute in the Administrative Entry.

NOTE 2 – The right to create prescriptive access controls may also be governed directly by security policy; this provision is required to create access controls in new autonomous administrative areas.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

16.7 Access control and Directory operations

Each Directory operation involves making a series of *access control decisions* on the various protected items that the operation accesses.

For some operations (e.g. Modify operations), each such access control decision must grant access for the operation to succeed; if access is denied to any protected item, the whole operation fails. For other operations, protected items to which access is denied are simply omitted from the operation result and processing continues.

If the requested access is denied, further access control decisions may be needed to determine if the user has **DiscloseOnError** permissions to the protected item. Only if **DiscloseOnError** permission is granted may the Directory respond with an error that reveals the existence of the protected item; in all other cases, the Directory acts so as to conceal the existence of the protected item.

The access control requirements for each operation, i.e. the protected items and the access permission required to access each protected item, are specified in ITU-T Rec. X.511 | ISO/IEC 9594-3.

The algorithm by which any particular access control decision is made is specified in 16.8.

16.8 Access Control Decision Function

This subclause specifies how an access control decision is made for any particular protected item. It provides a conceptual description of the Access Control Decision Function (ACDF) for **basic-access-control.** It describes how ACI items are processed in order to decide whether to grant or deny a particular requestor a specified permission to a given protected item.

16.8.1 Inputs and outputs

For each invocation of the ACDF, the inputs are:

a) the requestor's Distinguished Name (as defined in 7.3 of ITU-T Rec. X.511 | ISO/IEC 9594-3), unique identifier, and authentication level, or as many of these as are available;

b) the protected item (an entry, an attribute, or an attribute value) being considered at the current decision point for which the ACDF was invoked;

c) the requested permission category specified for the current decision point;

d) the ACI items associated with the entry containing (or which is) the protected item. Protected items are described in 16.4.2.4. The scope of influence for ACI items within a **prescriptiveACI** attribute is described in 16.3.2 and 16.5.1. The scope of influence for ACI items within an **entryACI** attribute is described in 16.3.2 and 16.5.2. The scope of influence for ACI items within a **subentryACI** attribute is described in 16.5.3.

In addition, if the ACI items include any of the protected item constraints described in 16.4.2.4, the whole entry and the number of immediate subordinates of its superior entry may also be required as inputs.

The output is a decision to *grant* or *deny* access to the protected item.

In any particular instance of making an access control decision, the outcome shall be the same as if the steps in 16.8.2 through 16.8.4 were performed.

16.8.2 Tuples

For each ACI value in the ACI items of 16.8.1 d), expand the value into a set of *tuples*, one tuple for each element of the **itemPermissions** and **userPermissions** sets. Collect all tuples from all ACI values into a single set. Each tuple contains the following items:

(userClasses, authenticationLevel, protectedItems, grantsAndDenials, precedence)

For any tuple whose **grantsAndDenials** specify both grants and denials, replace the tuple with two tuples – one specifying only grants and the other specifying only denials.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

16.8.3 Discarding non-relevant tuples

Perform the following steps to discard all non-relevant tuples:

1) Discard all tuples that do not include the requestor in the tuple's **userClass** [16.4.2.4 b)] as follows:

   – For tuples that grant access, discard all tuples that do not include the requestor's identity in the tuples's **userClasses** element taking into account **uniqueIdentifier** elements if relevant. Where a tuple specifies a **uniqueIdentifier**, a matching value must be present in the requestor's identity if the tuple is not to be discarded. Discard tuples that specify an authentication level higher than that associated with the requestor in accordance with 16.4.2.3.

   – For tuples that deny access, retain all tuples that include the requestor in the tuple's **userClasses** element, taking into account **uniqueIdentifier** elements if relevant. Also retain all tuples that deny access and which specify an authentication level higher than that associated with the requestor in accordance with 16.4.2.3. All other tuples that deny access are discarded.

   NOTE 1 – The second requirement in the second sub-item above (i.e. to retain any tuple that denies access and also specifies an authentication level higher than that associated with the requestor) reflects the fact that the requestor has not adequately proved non-membership in the user class for which the denial is specified.

2) Discard all tuples that do not include the protected item in **protectedItems** [16.4.2.4 a)].

3) Examine all tuples that include the **maxValueCount**, **maxImmSub**, **restrictedBy**, or **contexts**. Discard all such tuples which grant access and which do not satisfy any of these constraints [16.4.2.4 a)].

4) Discard all tuples that do not include the requested permission as one of the set bits in **grantsAndDenials** [16.4.1, 16.4.2.4 f)].

   NOTE 2 – The order in which discarding of non-relevant tuples is performed does not change the output of the ACDF.

16.8.4 Selecting highest precedence, most specific tuples

Perform the following steps to select those tuples of highest precedence and specificity:

1) Discard all tuples having a **precedence** less than the highest remaining precedence.

2) If more than one tuple remains, choose the tuples with the *most specific user class*. If there are any tuples matching the requestor with **UserClasses** element **name** or **thisEntry**, discard all other tuples. Otherwise if there are any tuples matching **UserGroup**, discard all other tuples. Otherwise if there are any tuples matching **subtree**, discard all other tuples.

3) If more than one tuple remains, choose the tuples with the *most specific protected item*. If the protected item is an attribute and there are tuples that specify the attribute type explicitly, discard all other tuples. If the protected item is an attribute value, and there are tuples that specify the attribute value explicitly, discard all other tuples. A protected item which is a **rangeOfValues** is to be treated as specifying an attribute value explicitly.

Grant access if and only if one or more tuples remain and all grant access. Otherwise deny access.

16.9 Simplified Access Control

16.9.1 Introduction

This subclause describes the functionality of an access control scheme, known as Simplified Access Control, that is designed to provide a subset of functionality found in Basic Access Control.

16.9.2 Definition of Simplified Access Control functionality

The functionality of Simplified Access Control is defined as follows:

a) access control decisions shall be made only on the basis of **ACIItem** values of **prescriptiveACI** and **subentryACI** operational attributes.

   NOTE 1 – **entryACI**, if present, shall not be used to make access control decisions.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

b) access control specific administrative areas shall be supported. Access control inner administrative areas shall not be used. Particular access decisions shall be made on the basis of **ACIItem** values obtained from a single Administrative Point, or from subentries of that Administrative Point.

NOTE 2 – Values of **prescriptiveACI** attributes appearing in subentries of Administrative Points containing no **id-ar-accessControlSpecificArea** Administrative Role attribute value shall not be used to make access control decisions.

c) all other provisions shall be as defined for basic access control.

17 Rule-based Access Control

17.1 Scope and application

This clause defines a specific access control scheme (of possibly many) for the Directory. The access control scheme defined herein is identified with the **accessControlScheme** operational attribute by giving it the value **rule-based-access-control** or if used in conjunction with the basic or simplified access control schemes defined in clause 16, **rule-and-basic-access-control** or **rule-and-simple-access-control**. Subclause 15.2.2 describes which entries contain the **accessControlScheme** operational attribute.

The scheme defined here is only concerned with controlling access to the Directory information within the DIB (potentially including tree structure and access control information). It does not address controlling access for the purpose of communication with a DSA application-entity. Control of access to information means the prevention of unauthorized detection, disclosure, or modification of that information.

17.2 Rule-based Access Control model

There may be environments where information relating to the clearance (instead of identity) of the requestor is used in determining whether or not access to an attribute value is to be denied. This is defined as Rule-based Access Control and uses administratively imposed access control policy rules in determining when access is to be denied to certain contents of the Directory. If access is denied under Rule-based Access Control, it cannot be allowed under other access control schemes. The Rule-based Access Control model identifies the information used in determining whether access is to be denied. This is applied to every operation. Each access control decision involves:

a) Access control information associated with the attribute values being accessed. This access control information is called a security label.

b) Access control information associated with the user requesting the operation. This access control information is called the clearance. The user requesting the operation is called the requestor.

c) Rules which define whether an access is authorized given a security label and a clearance, called security policies.

See Figure 12.

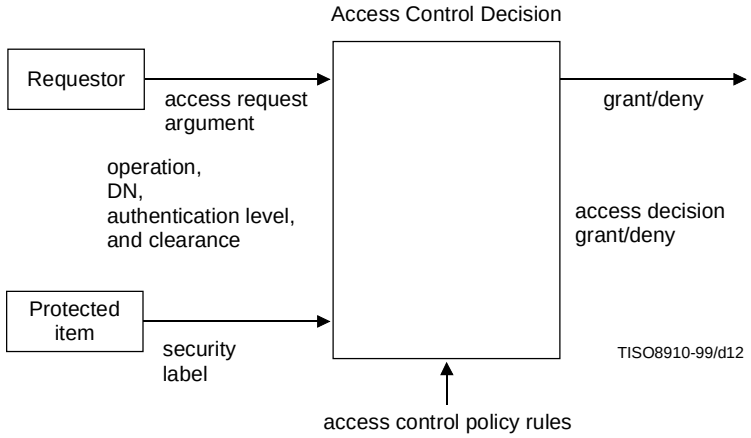


Figure 12 – Rule-based Access Control Decision Model

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The security label(s) can be securely associated with attribute values by binding the label to the information through the use of a digital signature or other integrity mechanism. A security label is a property of the attribute value and is associated with the value as a context.

The clearance is needed to enable a comparison to be made against the security label. The clearance can be bound to the Distinguished Name of the requestor through a certificate extension field (subject Directory attribute) or through an attribute certificate. The means selected for providing the clearance is a matter for the security policy in effect.

NOTE – The use of other clearance information, (e.g. that associated with any intermediate DSAs which may have chained the operation), is outside the scope of this Directory Specification.

The security rules to be applied in making an access control decision are defined as part of the security policy. The security policy is either identified in the security label or defined for the environment containing the labelled object.

17.3 Access control administrative areas

As for basic access control (see 16.3), the DIT is divided into administrative areas including Access Control Specific Areas. The administrative entry for an ACSA identifies the labelling security policies (access rules) that are applicable for that administrative area as well as the applicable access control scheme (**rule-based-access-control** or **rule-and-basic-access-control** or **rule-and-simple-access-control** or some other access control scheme).

17.4 Security Label

17.4.1 Introduction

Security labels may be used to associate security-relevant information with attributes within the Directory.

Security labels may be assigned to an attribute value in line with the security policy in force for that attribute. The security policy may also define how security labels are to be used to enforce that security policy.

A security label comprises a set of elements optionally including a security policy identifier, a security classification, a privacy mark, and a set of security categories (see 8.5.9 of ITU-T Rec. X.411 | ISO/IEC 10021-4). The security label is bound to the attribute value using a digital signature or other integrity mechanism.

17.4.2 Administration of Security Labels

A security label is assigned to an attribute value by an administrative function before being placed in the Directory.

This administrative function is responsible for assigning security labels to attribute values in line with the security policy in force for the ACSA.

The binding of a security label is protected using a digital signature or other integrity mechanism. This protection is applied by the administrative function, or creator of the attribute value.

17.4.3 Labelled Attribute Values

A security label context associates a security label with an attribute value. Only a single label can be associated with an attribute value. That is, the security label context is single-valued. In addition, matching rules for the security label context are not supported.

NOTE – The concept of contexts is introduced in 8.7.

```
attributeValueSecurityLabelContext CONTEXT ::= {
        SYNTAX      SignedSecurityLabel     -- At most one security label context can be assigned to an
                                            -- attribute value
        ID          id-avc-attributeValueSecurityLabelContext }

SignedSecurityLabel  ::=  SIGNED {SEQUENCE  {
        attHash             HASH {AttributeTypeAndValue},
        issuer              Name            OPTIONAL, -- name of labelling authority
        keyIdentifier       KeyIdentifier   OPTIONAL,
        securityLabel       SecurityLabel } }

KeyIdentifier  ::=  OCTET STRING
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

This context is not used to filter or select particular attributes, as for other contexts, and the mechanisms associated with contexts (fallback, default context values, etc.) are not used to apply rule-based access control.

The **attHash** component contains the resulting value of applying a cryptographic hashing procedure to DER-encoded octets, as defined in ITU-T Rec. X.509 | ISO/IEC 9594-8.

The **issuer** component conveys the name of the labelling authority.

The **keyIdentifier** component may be the identifier of a certified public key as held in the Subject Public Key Identifier extension field defined in ITU-T Rec. X.509 | ISO/IEC 9594-8 or the identifier of a symmetric key and associated security control information.

The **securityLabel** component is composed of a set of elements optionally including a security policy identifier, a security classification, a privacy mark, and a set of security categories as defined in 8.5.9 of ITU-T Rec. X.411 | ISO/IEC 10021-4.

17.5 Clearance

A clearance attribute associates a clearance with a named entity including DUAs.

```
clearance  ATTRIBUTE  ::=  {
        WITH SYNTAX     Clearance
        ID              id-at-clearance }

Clearance  ::=  SEQUENCE  {
        policyId                OBJECT IDENTIFIER,
        classList               ClassList                   DEFAULT {unclassified},
        securityCategories      SET OF SecurityCategory     OPTIONAL }

ClassList  ::=  BIT STRING {
        unmarked        (0),
        unclassified    (1),
        restricted      (2),
        confidential    (3),
        secret          (4),
        topSecret       (5) }
```

The **policyId** component conveys an identifier that may be used to identify the security policy in force to which the clearance classList and securityCategories relates.

The **classList** component includes a list of classifications that are associated with the named entity.

The **securityCategories** (see 8.5.9 of ITU-T Rec. X.411 | ISO/IEC 10021-4) component, if present, provides further restrictions within the context of a **classList**.

NOTE – A clearance is securely bound to a named entity using an Attribute Certificate (ITU-T Rec. X.509 | ISO/IEC 9594-8), a public key Certificate extension field (e.g. within the **SubjectDirectoryAttribute** extension) (ITU-T Rec. X.509 | ISO/IEC 9594-8), or means outside the scope of this Directory Specification.

17.6 Access Control and Directory operations

Each Directory operation involves making a series of access control decisions on the attribute values that the operation accesses.

For some operations (e.g. RemoveEntry operation), even though the operation may appear to have succeeded if access is denied to one or more attribute values, the hidden attributes would remain in the directory. For other operations, protected items to which access is denied are simply omitted from the operation result and processing continues.

The access control requirements for each operation are specified in ITU-T Rec. X.511 | ISO/IEC 9594-3.

The algorithm by which any particular access control decision is made is specified as:

- If access to all the attribute values of an entry is denied under **rule-based-access-control**, the access is denied to that entry for all operations.
- If access to all the attribute values of an attribute is denied under **rule-based-access-control**, the access is denied to that attribute for all operations.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

- Rule-based access control affects operations on reading attribute values (e.g. Read, Search) in that the attribute value is not visible (the operation is carried out as though the attribute value is not present) if access is denied to the attribute value.
- Rule-based access control affects operations which involve removing an entry (e.g. Remove Entry) in that they do not remove those attribute values to which access is denied.
- Rule-based access control affects operations which involve removing an attribute type (e.g. Modify Entry – Remove Attribute) in that they do not remove those attribute values to which access is denied.
- Rule-based access control affects operations which involve removing an attribute value (e.g. Modify Entry – Remove Value) in that these operations fail if the access is denied to the attribute value.

17.7 Access Control Decision Function

This subclause specifies how an access control decision is made for any particular attribute value. It provides a conceptual description of the Access Control Decision Function (ACDF) for **rule-based-access-control**. It describes how a clearance and a security label are processed in order to decide whether to grant or deny a particular requestor a specified permission to a given attribute value. The decision function applies the security policy rules which establish whether an access is authorized on an attribute value given its security label and the requestor's clearance. The definition of the security rules is outside the scope of the Directory Specifications. A simplified example of security policy rules for **rule-based-access-control** is given in K.10.

17.7.1 Inputs and outputs

For each invocation of the ACDF, the inputs are:

a) the requestor's clearance (as defined in 17.5);

b) attribute value being considered at the current decision point for which the ACDF was invoked;

c) the security policy in force for the access-control-specific area;

d) security label bound to the attribute value.

The output is a decision whether to deny access to the attribute value.

For any particular instance of making an access control decision, the outcome shall be the same as if the steps in 17.6 were performed.

17.8 Use of Rule-based and Basic Access Control

If both rule-based and basic access control are in effect, the order in which they are applied is a local matter, except that if access is denied to the entry, an attribute type or an attribute value by either mechanism it shall not be granted by the other mechanism. In this respect, *DiscloseOnError* (see 16.2.3 and 16.2.4) permission of **basic-access-control** is a permission that shall not override a deny of **rule-based-access-control**.

18 Cryptographic Protection in Storage

18.1 Data Integrity in Storage

18.1.1 Introduction

In some situations, the Directory may not give sufficient assurance that data is unchanged in storage, regardless of access controls. The integrity of data stored in the Directory may be validated using digital signatures held as part of the Directory Information. Either the digital signature of an entry or selected attributes within an entry may be held as an attribute (see 18.1.2), or the digital signature of a single attribute value may be held in a context (see 18.1.3).

NOTE – The DSA shall maintain the encoding of the attribute placed in the Directory to ensure that the signature calculated on the returned result is correct.

18.1.2 Protection of an Entry or Selected Attribute Types

Data integrity of attributes in storage is provided through use of digital signatures held alongside the attributes they are protecting. The integrity of a whole entry, or of all attribute values for selected attributes in an entry, is protected by an attribute holding a digital signature of all the attribute values being protected.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

This digital signature is created by an authority or directory user responsible for placing the information in the directory entry. The digital signature can be validated by any user reading the attribute values for the entry. The directory service itself is not involved in the creation or validation of the digital signature held in this attribute.

This integrity mechanism protects the integrity of directory attributes both in storage and during transfer between components of the Directory (DSAs and DUAs). This integrity mechanism does not depend on the security of the directory service itself.

Digital signatures applied to the whole entry do not include operational or collective attributes. Any attribute value contexts are included.

Additional control information is held along with the digital signature. This includes information on the authority or the user who created the digital signature, the information to limit the validity of the signature and a reference, in the form of a certificate serial number, which could be used in a certificate revocation list to revoke the validity of the signed attributes.

The following defines an attribute type to hold a digital signature, along with associated control information, which provides integrity of a whole entry or all values of selected attribute types.

```
attributeIntegrityInfo ATTRIBUTE  ::=  {
        WITH SYNTAX                     AttributeIntegrityInfo
        EQUALITY MATCHING RULE          attributeIntegrityMatch
        ID                              id-at-attributeIntegrityInfo}

AttributeIntegrityInfo  ::=  SIGNED { SEQUENCE  {
        issuer          Name,                   -- Authority or data originators name
        scope           Scope,                  -- Identifies the attributes protected
        subject         Name  OPTIONAL,         -- If not present can be implied from Name of entry
        keyIdentifier   KeyIdentifier  OPTIONAL,
        attribsHash     AttribsHash } }         --  Hash value of protected attributes

Scope  ::=  CHOICE  {
        wholeEntry      [0]   NULL,             -- Signature protects all attribute values in this entry
        selectedTypes   [1]   SelectedTypes
                              -- Signature protects all attribute values of the selected attribute types
        }

SelectedTypes  ::=  SEQUENCE OF AttributeType

AttribsHash  ::=  HASH { SEQUENCE {
        subject                 Name,
        protectedAttributes     SEQUENCE OF ProtectedAttributes } }
                -- Attribute type and values with associated context values for the selected Scope
```

18.1.2.1 Attribute Integrity Matching Rule

The matching rule for the issuer, scope or key identifier is as follows:

```
attributeIntegrityMatch  MATCHING-RULE  ::=  {
        SYNTAX      AttributeIntegrityAssertion
        ID          id-mr-attributeIntegrityMatch }

AttributeIntegrityAssertion  ::=  SEQUENCE  {
        issuer          Issuer          OPTIONAL,
        scope           Scope           OPTIONAL,
        keyIdentifier   KeyIdentifier   OPTIONAL }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The matching rule returns TRUE if all of the components that are present in the presented value match the corresponding components of the attribute value, as follows:

a) **issuer** matches if the stored attribute value contains the name component of the same name type as indicated in the presented value;

b) **scope** matches if it is equal to the scope component of the stored attribute value;

c) **keyIdentifier** matches if it is equal to the **keyIdentifier** component of the stored attribute value.

NOTE – Conformant implementations are not required to recognize all possible forms.

18.1.3 Context for Protection of a Single Attribute Value

The following defines a context to hold a digital signature, along with associated control information, which provides integrity for a single attribute value. Any attribute value contexts are included in the signature, excluding the context used to hold signatures.

```
attributeValueIntegrityInfoContext  CONTEXT  ::=  {
        SYNTAX          AttributeValueIntegrityInfo
        ASSERTED AS     AVIAssertion
        ID              id-avc-attributeValueIntegrityInfoContext}

AttributeValueIntegrityInfo  ::=  SIGNED { SEQUENCE {
        issuer          Name,                           -- Authority or data originators name
        subject         Name            OPTIONAL,       -- May be implied by Name for entry
        keyIdentifier   KeyIdentifier   OPTIONAL,
        aVIHash         AVIHash } }                     -- Hash value of protected attribute

AVIHash  ::=  HASH { SEQUENCE {
        subject                         Name OPTIONAL,
                -- Not present if name already in  AttributeValueIntegrityInfo
        protectedAttributeValue         AttributeTypeValueContexts } }
                -- Attribute type and value with associated context values

AttributeTypeValueContext ::= SEQUENCE {
        type            ATTRIBUTE.&id ({SupportedAttributes}),
        value           ATTRIBUTE.&Type ({SupportedAttributes}{@type}),
        contextList     SET SIZE (1..MAX) OF Context OPTIONAL }

AVIAssertion ::= SEQUENCE {
        issuer          Name            OPTIONAL,
        keyIdentifier   KeyIdentifier   OPTIONAL }
```

NOTE – This context is not generally used to filter or select particular attributes, as for other contexts, however, it is possible to use context selection to obtain attribute values for which there are digital signatures from known issuers or key identifiers.

18.2 Confidentiality of stored data

18.2.1 Introduction

In some situations, the Directory may not give sufficient assurance that data is kept confidential in storage, regardless of access controls. Confidentiality of attributes in storage is provided through use of:

a) A template for the definition of an attribute type which is an encrypted variant of an existing attribute type.

b) An attribute for distributing confidentiality keys.

Once the Directory accepts a protected variant of an attribute value, the matching rules of the original (cleartext) attribute cannot be invoked.

NOTE – Any mechanism can be used to distribute the keys required to protect attributes defined using the "Confidential Attribute" template. Further information on key management techniques can be found in ISO/IEC 11770: Information technology – Security techniques – Key Management.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

18.2.2 Encrypted Attribute Value Template

If an encrypted variant is required of an existing directory attribute type, the following syntax is used:

```
EncryptedAttributeSyntax {AttributeSyntax} ::= SEQUENCE {
        keyInfo     SEQUENCE OF KeyIdOrProtectedKey,
        encAlg      AlgorithmIdentifier,
        encValue    ENCRYPTED { AttributeSyntax } }

KeyIdOrProtectedKey ::= SEQUENCE {
        keyIdentifier       [0]   KeyIdentifier     OPTIONAL,
        protectedKeys       [1]   ProtectedKey      OPTIONAL }
              -- At least one key identifier or protected key must be present

ProtectedKey ::= SEQUENCE {
        authReaders         AuthReaders,      -- if absent, use attribute in authorized reader entry
        keyEncAlg           AlgorithmIdentifier  OPTIONAL,      -- algorithm to encrypt encAttrKey
        encAttKey           EncAttKey }

                            -- confidentiality key protected with authorized user's
                            -- protection mechanism

AuthReaders ::= SEQUENCE OF Name

EncAttKey ::= PROTECTED {SymmetricKey, keyProtection}

SymmetricKey ::= BIT STRING

keyProtection PROTECTION-MAPPING ::= {
        SECURITY-TRANSFORMATION {genEncryption} }
```

NOTE 1 – It is not reasonable that ordering matching rules or substring matching rules be used for encrypted attributes. It is also recommended that encrypted attributes are only defined for user attributes.

AttributeSyntax is the clear text syntax of the attribute and **EncryptedAttributeSyntax** the syntax of the equivalent confidential attribute.

The **keyIdentifier** may be the identifier of a certified public key as held in the Subject Public Key Identifier extension field defined in ITU-T Rec. X.509 | ISO/IEC 9594-8, clause 12, or the identifier of a symmetric key and associated security control information.

The **authReaders** identifies those subjects who are permitted to retrieve the **ProtectedKey**.

The **keyEncAlg** is the identifier for the algorithm used to encrypt the **encAttKey**, which is the identifier for the key used to apply the confidentiality service.

The **genEncryption SECURITY-TRANSFORMATION** defines the transformation which is used for key distribution. This is further defined in 15.3.1. This transformation may be used with any encryption algorithm, including reversible public key algorithms and symmetric key algorithms.

NOTE 2 – Key management, which includes key distribution, is within the purview of some administratively controlled security policy. Therefore, this specification does not constrain the **genEncryption SECURITY-TRANSFORMATION** in such a manner as to preclude the security manager of the DMD from specifying the precise details of key distribution. Furthermore, this specification does not require qualities of the **genEncryption SECURITY-TRANSFORMATION** such that these qualities preclude the use of particular key distribution mechanisms.

Support for an attribute does not imply support for the equivalent encrypted attribute.

Where a user attribute is defined in this set of specifications, the object identifier for the encrypted equivalent of that attribute is allocated in ITU-T Rec. X.520 | ISO/IEC 9594-6, Annex A.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

18.2.3 Attribute for Confidentiality Key

If it is necessary to distribute information about the encryption that provides confidentiality of stored data, the following attribute shall be used:

```
confKeyInfo  ATTRIBUTE  ::= {
        WITH SYNTAX                     ConfKeyInfo
        EQUALITY MATCHING RULE          readerAndKeyIDMatch
        ID                              id-at-confKeyInfo }

ConfKeyInfo  ::=  SEQUENCE  {
        keyIdentifier       KeyIdentifier,
        protectedKey        ProtectedKey }
```

18.2.4 Reader and Key Identifier Matching Rule

A matching rule for a list of authorized readers and their associated keys is as follows:

```
readerAndKeyIDMatch MATCHING-RULE  ::=  {
        SYNTAX      ReaderAndKeyIDAssertion
        ID          id-mr-readerAndKeyIDMatch }

ReaderAndKeyIDAssertion  ::=  SEQUENCE {
        keyIdentifier       KeyIdentifier,
        authReaders         AuthReaders OPTIONAL }
```

The matching rule returns a TRUE if all of the components that are present in the presented value match the corresponding components of the attribute value, as follows:

a) **keyIdentifier** matches if it is equal to the **keyIdentifier** component of the **ProtectedKey** in the stored attribute value;

b) if present the **authReaders** matches if it is equal to one of the names in the **authReaders** component of the **ProtectedKey** in the stored attribute value.

SECTION 8 – DSA MODELS

19 DSA Models

This clause is concerned with general models describing various aspects of the components comprising the Directory, Directory System Agents (DSAs). Subsequent clauses treat additional DSA models.

19.1 Definitions

For the purposes of this Directory Specification, the following definitions apply:

19.1.1 context prefix: The sequence of RDNs leading from the Root of the DIT to the initial vertex of a naming context, corresponds to the distinguished name of that vertex.

19.1.2 DIB fragment: The portion of the DIB that is held by one master DSA, comprising one or more naming contexts.

19.1.3 naming context: A subtree of entries held in a single master DSA.

19.2 Directory Functional Model

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 13 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 ITU-T Rec. X.518 | ISO/IEC 9594-4.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

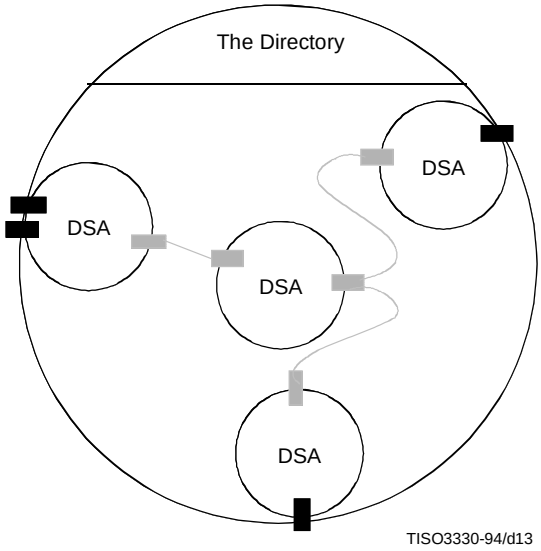


Figure 13 – The Directory Provided by Multiple DSAs

NOTE – A DSA will likely exhibit local behaviour and structure which is outside the scope of envisaged Directory Specifications. 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.

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 ITU-T Rec. X.519 | ISO/IEC 9594-5.

Clause 19 specifies the models that are used as the basis for specifying the distributed aspects of the Directory. A framework for the specification of operational models concerned with particular aspects of the operation of the components of the Directory, DSAs, is provided in clauses 23 through 26.

19.3 Directory Distribution Model

This subclause defines the principles according to which the DIB can be distributed.

Each entry within the DIB is administered by one, and only one, DSA's Administrator who is said to have administrative authority for that entry. Maintenance and management of an entry shall take place in a DSA administered by the administrative authority for the entry. This DSA is the *master DSA* for the entry.

Each master DSA within the Directory holds a *fragment* of the DIB. The DIB fragment held by a master DSA is described in terms of the DIT and comprises one or more naming contexts. A *naming context* is a subtree of the DIT, all entries of which have a common administrative authority and are held in the same master DSA. A naming context starts at a vertex of the DIT (other than the root) and extends downwards to leaf and/or non-leaf vertices. Such vertices constitute the border of the naming context. The superior of the starting vertex of a naming context is not held in that master DSA. Subordinates of the non-leaf vertices belonging to the border denote the start of further naming contexts.

NOTE 1 – The DIT is therefore partitioned into disjoint naming contexts, each under the administrative authority of a single master DSA.

NOTE 2 – A naming context in itself is not an administrative area having an administrative point or an explicit subtree specification, but it may coincide with an administrative area.

It is possible for a master DSA's administrator to have administrative authority for several disjoint naming contexts. For every naming context for which a master DSA has administrative authority, it shall logically hold the sequence of RDNs which lead from the root of the DIT to the initial vertex of the subtree comprising the naming context. This sequence of RDNs is called the *context prefix* of the naming context.

NOTE 3 – The primary distinguished name of the naming context shall be used as the context prefix. Contexts and alternative values with context may optionally be included in the RDNs.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

A master DSA's administrator may delegate administrative authority for any immediate subordinates of any entry held locally to another master DSA. A master DSA that delegated authority is called a *superior DSA* and the context that holds the superior entry of one for which the administrative authority was delegated, is called the *superior naming context*. Delegation of administrative authority begins with the root and proceeds downwards in the DIT; that is, it can only occur from an entry to its subordinates.

Figure 14 illustrates a hypothetical DIT logically partitioned into five naming contexts (named A, B, C, D and E), which are physically distributed over three DSAs (DSA1, DSA2, and DSA3).

From the example, it can be seen that the naming contexts held by particular master DSAs may be configured so as to meet a wide range of operational requirements. Certain master DSAs may be configured to hold those entries that represent higher level naming domains within some logical part(s) of the DIB, the organizational structure of a large company say, but not necessarily all the subordinate entries. Alternatively, master DSAs may be configured to hold only those naming contexts representing primarily leaf entries.

From the above definitions, the limiting case for a naming context can be either a single entry or the whole of the DIT.

Whilst the logical to physical mapping of the DIT onto master DSAs is potentially arbitrary, the task of information location and management is simplified if the master DSAs are configured to hold a small number of naming contexts.

DSAs may hold entry-copies as well as entries. Shadowed entries, the only sort of entry-copy considered in the Directory Specifications, are maintained by means of the shadowing service described in ITU-T Rec. X.525 | ISO/IEC 9594-9. In addition to this standardized sort of replicated information, two additional non-standardized sorts of entry-copy may be encountered in the Directory.

– Copies of an entry may be stored in other DSA(s) through bilateral agreement.

– Copies of an entry may be acquired by storing (locally and dynamically) a cache-copy of an entry which results from a request.

NOTE 4 – The means by which these copies are maintained and managed is not defined in these Directory Specifications. Due to more precise handling of features like access control, it is recommended that the shadow service be used instead of using cached-copies.

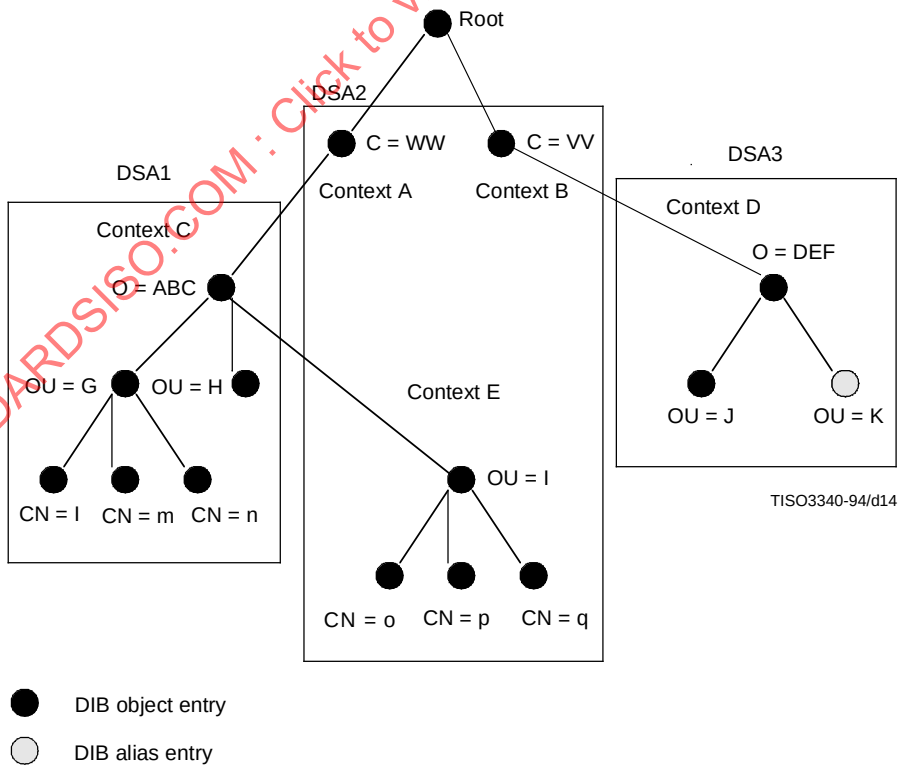


Figure 14 – Hypothetical DIT

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

A DSA holding an entry-copy is a *shadow DSA* for that entry. A shadow DSA may hold a copy of a naming context or a portion thereof. The specification of the portion of a naming context that is shadowed is termed a *unit of replication*.

As described in 9.2 of ITU-T Rec. X.525 | ISO/IEC 9594-9, a unit or replication is defined within the Directory information model, and a specification mechanism is provided. The shadowing mechanism in the Directory is based on the definition of the subset of the DIT that will be shadowed. This subset is called *unit of replication*. The unit of replication comprises a three-part specification which defines the scope of the portion of the DIT to be replicated, the attributes to be replicated within that scope, and the requirements for subordinate knowledge. The unit of replication also implicitly causes the shadowed information to include policy information in the form of operational attributes held in entries and subentries (e.g. access control information) which is to be used to correctly perform Directory operations. The prefix information to be included begins at an autonomous administrative point and extends to the replication base entry.

The originator of a Directory request is informed (via **fromEntry**) as to whether information returned in response to a request is from an entry-copy or not. A service control, **dontUseCopy**, is defined which allows the user to prohibit the use of entry-copies to satisfy the request (although copy information may be used in name resolution).

NOTE 5 – Such name resolution will fail in some instances for a valid alternative name when resolved against a copy held in a pre-1997 edition DSA, or in a later edition DSA holding a copy with incomplete name information, where an RDN includes an attribute type for which there are multiple distinguished values differentiated by context.

In order for a DUA to begin processing a request it shall hold some information, specifically the presentation address, about at least one DSA that it can contact initially. How it acquires and holds this information is a local matter.

During the process of modification of entries it is possible that the Directory may become inconsistent. This will be particularly likely if modification involves aliases or aliased objects which may be in different DSAs. The inconsistency shall be corrected by specific administrator action, for example to delete aliases if the corresponding aliased objects have been deleted. The Directory continues to operate during this period of inconsistency.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

SECTION 9 – DSA INFORMATION MODEL

20 Knowledge

20.1 Definitions

For the purposes of this Directory Specification, the following definitions apply:

20.1.1 category: A characteristic of a knowledge reference that qualifies it as identifying a master or a shadow DSA.

20.1.2 commonly usable: A characteristic of a replicated area that permits general distribution of the access point of the DSA holding it; a commonly usable replicated area is normally a complete shadow copy of a naming context.

20.1.3 cross reference: A knowledge reference containing information about a DSA that holds an entry or entry-copy. This is used for optimization. The entry need have no superior or subordinate relationship to any entry in the DSA holding the cross reference.

20.1.4 immediate superior reference: A knowledge reference containing information about a DSA that holds the naming context (or a commonly usable replicated area derived from it) that is immediately superior to one held by the DSA for which the knowledge reference is relevant.

20.1.5 knowledge (information): DSA operational information held by a DSA that it uses to locate remote entry or entry-copy information.

20.1.6 knowledge reference: Knowledge which associates, either directly or indirectly, a DIT entry or entry-copy with the DSA in which it is located.

20.1.7 master knowledge: Knowledge of the master DSA for a naming context.

20.1.8 non-specific subordinate reference: A knowledge reference containing information about a DSA that holds one or more unspecified subordinate entries or entry-copies.

20.1.9 reference path: A continuous sequence of knowledge references.

20.1.10 shadow knowledge: Knowledge of one or more shadow DSAs for a naming context (if the knowledge is specific) or contexts (if non-specific).

20.1.11 subordinate reference: A knowledge reference containing information about a DSA that holds a specific subordinate entry or entry-copy.

20.1.12 superior reference: A knowledge reference containing information about a DSA considered capable of resolving (i.e. finding any entry within) the whole of the DIT.

20.2 Introduction

The DIB is distributed across a large number of master DSAs, each holding and having administrative authority for a DIB fragment. The principles governing this distribution are specified in 19.3.

In addition, these and other DSAs may hold copies of portions of the DIB.

It is a requirement of the Directory that, for particular modes of user interaction, the distribution of the directory be rendered transparent, thereby giving the effect that the whole of the DIB appears to be within each and every DSA.

In order to support this operational requirement, it is necessary that each DSA be able to gain access to the information held in the DIB associated with any name (i.e. any object's distinguished or alias names). If the DSA does not itself hold an object entry or object entry-copy associated with the name, it must be able to interact with a DSA that does, either directly or indirectly by means of direct and/or indirect interactions with other DSAs.

When the Directory user indicates that entry-copy information shall not be used to satisfy his request, the DSA servicing the request must be able to gain access, directly or indirectly, to the master DSA holding the entry information associated with the name supplied in the user's request.

This clause defines knowledge as that DSA operational information required to achieve these technical objectives. Subsequent clauses specify the representation of knowledge in the context of a general DSA information model.

NOTE – The preceding statements represent technical objectives of the Directory. Realization of these technical objectives depends on other matters (e.g. policy matters) in addition to a consistent configuration of knowledge in DSAs. Clauses 23 through 26 establish a framework to address some of these matters.

Annex M contains an illustration of the modelling of knowledge. The illustration is based on the hypothetical DIT given in Figure 14.

20.3 Knowledge References

Knowledge is that operational information held by a DSA that represents a partial description of the distribution of entry and entry-copy information held in other DSAs. Knowledge is used by a DSA to determine an appropriate DSA to contact when a request received from a DUA or another DSA cannot be satisfied with locally held information.

Knowledge consists of knowledge references. A *knowledge reference* associates, either directly or indirectly, the name of a Directory entry with a DSA holding the entry or a copy of the entry.

Names used in knowledge references, whether as context prefixes, DSA names or entry names, shall be primary distinguished values. Context, and alternative values with context, may also be included in RDNs.

NOTE – Name resolution may fail for a valid alternative name when knowledge references are held in pre-1997 DSAs which do not recognize multiple distinguished values differentiated by context, or in DSAs not holding all the alternative distinguished names in knowledge references or entry-copies.

20.3.1 Knowledge Categories

There are two categories of knowledge reference, master knowledge references and shadow knowledge references.

Master knowledge is knowledge of the access point of the master DSA for a naming context.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Shadow knowledge is knowledge of DSAs holding replicated Directory information; it may be distributed by shadow suppliers to shadow consumers by means of the replication procedures described in ITU-T Rec. X.525 | ISO/IEC 9594-9. Shadow knowledge is knowledge of the access point of a set of one or more shadow DSAs for a replicated area (a naming context or a portion thereof).

A DSA that is the object of shadow knowledge shall hold a commonly usable replicated area. One form of replicated area that is commonly usable is a complete shadow copy of a naming context. An incomplete shadow copy of a naming context held by a DSA may be commonly usable if it is sufficiently complete to satisfy the interrogation requests that users commonly make to the DSA. It is the responsibility of the administrative authority who causes shadow knowledge of a DSA holding an incomplete copy of a naming context to be distributed that the replicated area be commonly usable.

A given DSA may hold both master and shadow knowledge, the latter involving multiple shadow DSAs, regarding a particular naming context. The specific knowledge used in the processing of a request received from a DUA or another DSA, e.g. in the name resolution process, is determined by a DSA-specific selection procedure whereby the DSA computes, based on any non-standardized criteria deemed appropriate by the administrative authority, an access point of a DSA capable of progressing the request.

NOTE – The Directory Specifications do not constrain how master and shadow knowledge is used by DSAs (other than indirectly through constraints on DSA behaviour, for example, the **dontUseCopy** and **copyShallDo** service controls as specified in ITU-T Rec. X.511 | ISO/IEC 9594-3).

20.3.2 Knowledge Reference Types

The knowledge possessed by a DSA is defined in terms of a set of one or more knowledge references where each reference associates, either directly or indirectly, entries (or entry-copies) of the DIB with the DSA which holds those entries (or entry-copies).

A DSA may hold the following types of knowledge reference:

- a superior reference;
- immediate superior references;
- subordinate references;
- non-specific subordinate references; and
- cross references.

A knowledge reference of a particular type shall be either a master or shadow knowledge reference.

In addition, a DSA that participates in shadowing as a shadow supplier and/or consumer may hold one or more of the following types of knowledge reference:

- supplier references; and
- consumer references.

These knowledge reference types are described below.

20.3.2.1 Superior Reference

A superior reference consists of:

- the Access Point of a DSA.

Each non-first level DSA (see 20.5) maintains precisely one superior reference. The superior reference shall form part of a reference path to the root. Unless some method outside the standard is employed to ensure this, for example within a DMD, this shall be accomplished by referring to a DSA which holds a naming context or replicated area whose context prefix has fewer RDNs than the context prefix with fewest RDNs held by the DSA holding the reference.

20.3.2.2 Immediate Superior References

An immediate superior reference consists of:

- the context prefix of a naming context that is immediately superior to one held (as entries or entry-copies) by the DSA holding the reference;
- the Access Point of the DSA holding that naming context (as entries or entry-copies).

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Immediate superior references are an optional reference type that only occur when there is a hierarchical operational binding to the referenced DSA (see clause 26 in ITU-T Rec. X.518 | ISO/IEC 9594-4). In the absence of such explicit operational bindings, an immediate superior naming context may be referenced by means of a cross reference.

20.3.2.3 Subordinate References

A *subordinate reference* consists of:

- a context prefix corresponding to a naming context immediately subordinate to one held (as entries or entry-copies) by the DSA holding the reference;
- the Access Point of the DSA holding that naming context (as entries or entry-copies).

All naming contexts immediately subordinate to naming contexts held by a master DSA shall be represented by subordinate references (or non-specific subordinate references as described in 20.3.2.4).

In the case where a DSA holds entry-copies, the subordinate naming contexts may or may not be represented, depending on the shadowing agreement in effect.

20.3.2.4 Non-Specific Subordinate References

A *non-specific subordinate reference* consists of:

- the Access Points of a DSA that holds the entries (or entry-copies) of one or more immediately subordinate Naming Contexts.

This type of reference is optional, to allow for the case in which a DSA is known to contain some subordinate entries (or entry-copies) but the specific RDNs of those entries (or entry-copies) is not known.

For each naming context that it holds, a master DSA may hold zero or more non-specific subordinate references. DSAs accessed via a non-specific reference shall be able to resolve the request directly (either success or failure). In the event of failure, a **serviceError** reporting a problem of **unableToProceed** is returned to the requestor.

In the case where a DSA holds entry-copies, the non-specific subordinate references may or may not be represented, depending on the shadowing agreement in effect.

20.3.2.5 Cross References

A *cross reference* consists of:

- a Context Prefix;
- the Access Point of a DSA which holds the entries or entry-copies for that naming context.

This type of reference is optional and serves to optimize Name Resolution. A DSA may hold any number (including zero) of cross references.

20.3.2.6 Supplier References

A supplier reference held by a shadow consumer DSA consists of:

- the context prefix of the naming context from which the replicated area received from the shadow supplier is derived;
- the identifier of the shadowing agreement that the shadow consumer has established with a shadow supplier;
- the Access Point of the shadow supplier DSA;
- an indication of whether the shadow supplier of the replicated area is or is not the master; and
- optionally, the access point of the master DSA if the supplier is not the master.

20.3.2.7 Consumer References

A consumer reference held by a shadow supplier DSA consists of:

- the context prefix of a naming context from which the replicated area provided by the shadow supplier is derived;
- the identifier of the shadowing agreement that the shadow supplier has established with a consumer; and
- the Access Point of the shadow consumer DSA.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

20.4 Minimum Knowledge

It is a property of the Directory that each entry can be accessed independently of where a request is generated.

It is also a property of the Directory that, to achieve adequate levels of performance and availability, some requests can be satisfied using a copy of an entry, while other requests may only be satisfied using the entry itself (i.e. the information held at the master DSA for the entry).

To realize these location independence properties of the Directory, each DSA must maintain a minimum quantity of knowledge which depends on the particular configuration of the DSA.

The objective of these minimum requirements is to permit the distributed name resolution process to establish a reference path, as a continuous sequence of master knowledge references, to all naming contexts within the Directory.

Beyond these minimum requirements, additional knowledge may be employed to establish other reference paths to copies of naming contexts. Cross reference knowledge (master and shadow) may be employed to establish optimized reference paths to naming contexts and copies of naming contexts.

The minimum knowledge requirements for DSAs are specified in 20.4.1-20.4.4.

20.4.1 Superior Knowledge

Each DSA that is not a first level DSA shall maintain a single superior reference.

20.4.2 Subordinate Knowledge

A DSA that is the master DSA of a naming context shall maintain subordinate or non-specific subordinate references of category master knowledge to each master DSA holding (as master) an immediately subordinate naming context.

20.4.3 Supplier Knowledge

For each shadow supplier DSA that supplies it with a replicated area, a shadow consumer DSA shall maintain a supplier reference. If the shadow consumer's subordinate knowledge for the copy of the naming context is incomplete, it shall use its supplier reference to establish a reference path to subordinate information. This procedure is described in clause 20 of ITU-T Rec. X.518 | ISO/IEC 9594-4.

20.4.4 Consumer Knowledge

For each shadow consumer DSA that it supplies with a replicated area, a shadow supplier DSA shall maintain a consumer reference.

20.5 First Level DSAs

The DSA referenced by a superior reference assumes the burden of establishing a reference path to all of the DIT that is unknown to the referring DSA. A DSA referenced by other DSAs may itself maintain a superior reference. This recursive superior referral process stops at a set of *first level DSAs* upon whom the ultimate responsibility for the establishment of reference paths falls.

A first level DSA is characterized as follows:

a) it does not hold a superior reference;

b) it may hold one or more naming contexts immediately subordinate to the root of the DIT (as master or shadow DSA for the naming context); and

c) it holds a subordinate reference (of category master and/or shadow) for each naming context immediately subordinate to the root of the DIT that it does not itself hold.

The administrative authorities for first level DSAs are jointly responsible for the administration of the immediate subordinates of the root of the DIT. The procedures governing this joint administration are determined by multilateral agreements which are outside the scope of the Directory Specifications.

To limit the quantity of interrogation requests that might be directed to a master first level DSA (i.e. a DSA that is a master for a naming context immediately subordinate to the root of the DIT), it is possible to establish shadow first level DSAs for that master first level DSA. Such shadow DSAs hold copies of the entries and subordinate references immediately subordinate to the root held in its master (or supplier) first level DSA. They therefore may serve as the superior reference for non-first level DSAs.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

21 Basic Elements of the DSA Information Model

21.1 Definitions

For the purposes of this Directory Specification, the following definitions apply:

21.1.1 DSA information tree: The set of all DSEs held by a DSA when viewed from the perspective of their names.

21.1.2 DSA-shared attribute: An operational attribute in the DSA information model associated with a particular name whose value or values, if held by several DSAs, are identical (except during periods of transient inconsistency).

21.1.3 DSA-specific attribute: An operational attribute in the DSA information model associated with a particular name whose value or values, if held by several DSAs, need not be identical.

21.1.4 DSA-specific entry (DSE): The information held by a DSA that is associated with a particular name; the DSE may (but need not) contain the information associated with the corresponding Directory entry.

21.1.5 DSE type: An indication of the particular purpose of a DSE; a DSE may serve multiple purposes and thus have multiple types.

21.2 Introduction

The Directory information model describes how the Directory as a whole represents information about objects having a distinguished name and optionally alias names. In its description of the DIT, entries and attributes, the composition of the Directory as a set of potentially cooperating DSAs is abstracted from the model.

The DSA information model, on the other hand, is especially concerned with DSAs and the information that must be held by DSAs in order that the set of DSAs comprising the Directory may together realize the Directory information model. It is concerned with:

- how Directory information (object and alias entries and subentries) is mapped onto DSAs;
- how copies of Directory information may be held by DSAs;
- the operational information required by DSAs to perform name resolution and operation evaluation; and
- the operational information required by DSAs to engage in shadowing and to use shadowed information.

The purpose for modelling a representation of DSA operational information such as knowledge is to establish the general framework for management access to DSA operational information.

21.3 DSA-Specific Entries and their Names

In the DSA information model, the information repositories holding the information associated with a particular name are termed *DSA-Specific Entries* (DSEs). Directory entries exist in the DSA information model only as information elements from which DSEs may be composed. Operational attributes specific to the DSA information model comprise the other variety of information element from which DSEs may be composed.

If a DSA holds any information concerning a name directly (i.e. information held in a repository identified by the name), it is said to know or have knowledge of that name.

For each name known by a DSA, all the information held by the DSA directly associated with the name other than the name itself is represented by one DSE. This latter information (i.e. the RDN and its relationship to the DIT) is not represented explicitly as attributes in the DSA information model; the set of names known by a DSA constitute an implicit fabric on which the associated DSEs can be considered to be attached.

NOTE 1 – One consequence of the way the DSA information model handles names is that, for DSEs that are not of type entry, alias or subentry, the AVA(s) expressing the RDN of the DSE is not modelled as held in (an) attribute(s).

Where alternative names exist because of naming attributes having multiple distinguished values differentiated by context, a single DSE represents all the information held by the DSA about all the alternative names. This is modelled in the DSA information model as a single name with context-specific variants, rather than as separate names.

NOTE 2 – For consistent name resolution and interworking with pre-1997 DSAs, every DSA shall have information about at least the primary distinguished values of all attributes contributing to a name, and desirably as many of the alternative distinguished values as possible.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The set of all names known by a DSA, together with the information associated with each name, when viewed from the perspective of these names, is termed the *DSA information tree* for that DSA. A DSA information tree is depicted in Figure 15.

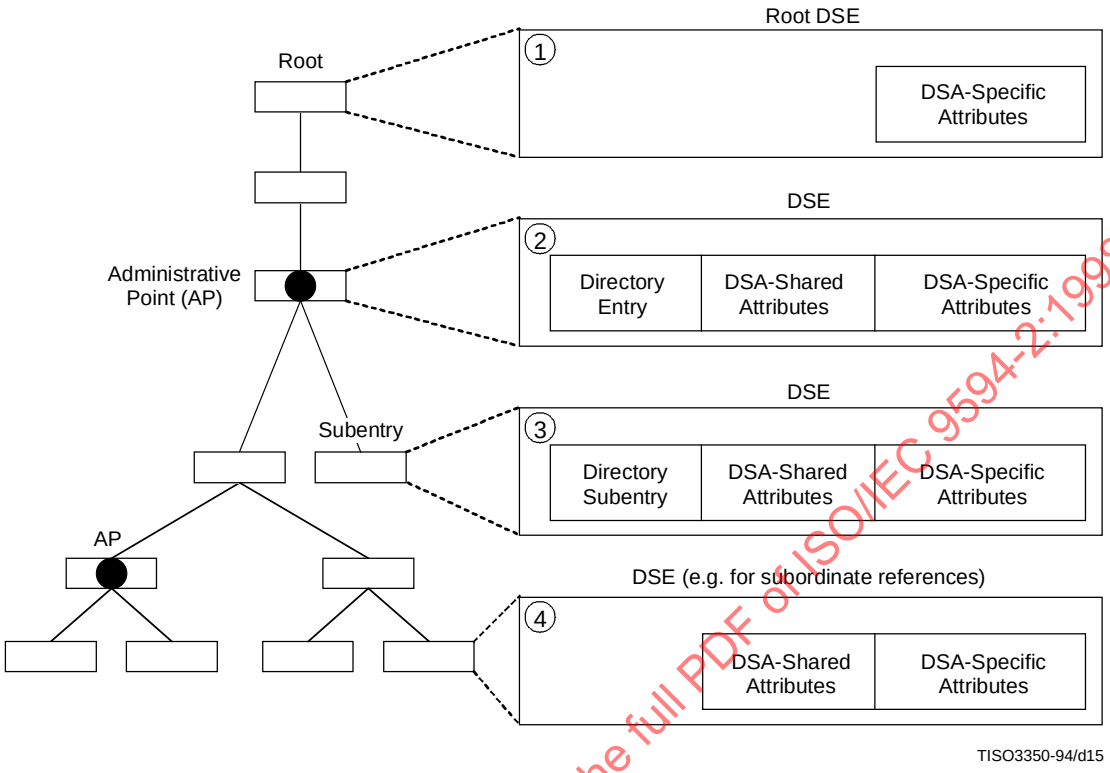


Figure 15 – A DSA Information Tree

The minimum information that a DSA may associate with a name, and thus know the name, consists of an expression of the purpose for which the name is known (i.e. the role played by the name in the operation of the DSA knowing it). This purpose is represented in the DSA information model by the DSA-specific attribute, **dseType**.

In addition, a DSE may hold other information associated with the name such as an entry or entry-copy, DSA-shared attributes and DSA-specific attributes.

A DSE may represent a Directory entry directly, a portion of an entry or no Directory information. The information held in a DSE varies, depending on its type or purpose. In general, the following sorts of DSEs may occur in DSAs.

– A DSE directly representing a Directory entry contains the user and operational attributes corresponding to that Directory entry (as depicted in DSE 2 in Figure 15). The DSE may also contain DSA-shared and DSA-specific attributes.

– A DSE representing a portion of an entry (as a result of shadowing) contains some of the user and operational attributes corresponding to the Directory entry, DSA-specific attributes and may also contain DSA-shared attributes.

– A subentry DSE representing, for example prescriptive ACI or collective attributes, contains the relevant user and operational attributes corresponding to a Directory subentry (as depicted in DSE 3 in Figure 15). The DSE may also contain DSA-shared and DSA-specific attributes.

– A DSE representing no Directory entry information contains only DSA-shared and/or DSA-specific attributes (as depicted in DSEs 1 and 4 in Figure 15). For example, a DSE representing a subordinate reference may have a DSA-shared attribute that indicates the master access point and a DSA-specific attribute to indicate that the DSE is a subordinate reference.

NOTE – The DSE is a conceptual entity which facilitates the specification and modelling of information components in a consistent and convenient way. Although DSEs are said to "hold" or "store" information, this is not intended to impose any particular constraints or data structure on implementations.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

21.4 Basic Elements

A DSE is comprised of three basic elements: the DSE type, some number of DSA operational attributes (the DSE type is one of these) and optionally an entry or entry-copy.

21.4.1 DSA Operational Attributes

Two varieties of operational attribute occur in the DSA information model that do not correspond to information in Directory entries. Those are DSA-shared and DSA-specific attributes.

A *DSA-shared attribute* is an operational attribute in the DSA information model associated with a particular name whose value or values, if held by several DSAs, are identical (except during periods of transient inconsistency). A DSA may hold a shadow-copy of a DSA-shared attribute.

A *DSA-specific attribute* is an operational attribute in the DSA information model associated with a particular name whose value or values, if held by several DSAs, need not be identical. A DSA-specific attribute represents operational information that is specific to the functioning of the DSA holding it. A DSA cannot hold a shadow-copy of a DSA-specific attribute.

NOTE – While a shadow-supplier DSA may provide a shadow-consumer DSA with a DSA-specific attribute, this is conceptually not a shadow-copy of information held by the supplier but, rather, information produced by the supplier for the consumer which the consumer may then use and modify.

21.4.2 DSE Types

The type of a DSE, represented in the DSA information model by the DSA-specific operational attribute **dseType**, indicates the particular purpose (or role) of a DSE. This purpose is indicated by the named bits of the single value of the **dseType** attribute. As a DSE may serve several purposes, several named bits of the **dseType** attribute may be set to represent these purposes. A number of combinations of named bits that are likely to occur are specified in Annex M.

The phrase "a DSE of type **x**" is used in the Directory Specifications to indicate that the named bit **x** of the DSE's **dseType** attribute has been set. For a DSE of type **x**, other named bits may or may not be set, as required. The alternate phrase "the DSE type includes **x**" may also be used.

The syntactic specification of the **dseType** operational attribute may be expressed using the attribute notation as follows:

```
dseType  ATTRIBUTE  ::= {
        WITH SYNTAX                     DSEType
        EQUALITY MATCHING RULE          bitStringMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           dSAOperation
        ID                              id-doa-dseType }
```

This DSA-specific operational attribute is managed by the DSA itself.

The ASN.1 type that represents the syntax of the possible values of the **dseType** attribute is **DSEType**. Its definition is:

```
DSEType  ::=  BIT STRING {
        root            (0),    -- root DSE --
        glue            (1),    -- represents knowledge of a name only --
        cp              (2),    -- context prefix --
        entry           (3),    -- object entry --
        alias           (4),    -- alias entry --
        subr            (5),    -- subordinate reference --
        nssr            (6),    -- non-specific subordinate reference --
        supr            (7),    -- superior reference --
        xr              (8),    -- cross reference --
        admPoint        (9),    -- administrative point --
        subentry        (10),   -- subentry --
        shadow          (11),   -- shadow copy --
        immSupr         (13),   -- immediate superior reference --
        rhob            (14),   -- rhob information --
        sa              (15),   -- subordinate reference to alias entry --
        dsSubentry      (16) }  -- DSA Specific subentry --
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The values of **DSEType** are:

a) **root**: The root DSE contains DSA-specific attributes, used by the DSA, that characterize that DSA as a whole. The name corresponding to the root DSE is the degenerate name consisting of a sequence of zero RDNs.

NOTE – Information that characterizes a DSA that is to be made available via the Directory abstract service is contained in the DSA's entry. A DSA may, but need not, hold its own entry or a copy of its own entry.

b) **glue**: A glue DSE represents knowledge of a name only. A DSA holding a context prefix DSE or a cross reference DSE may hold glue DSEs to represent the names of the superiors of the context prefix or cross reference DSE if no other operational information (e.g. knowledge) is associated with those names. This is illustrated in Figure 15. A DSE of type glue shall not have any other **DSEType** bit set.

c) **cp**: The DSE representing the context prefix of a naming context.

d) **entry**: A DSE that holds an object entry.

e) **alias**: A DSE that holds an alias entry.

f) **subr**: A DSE that holds a specific knowledge attribute to represent a subordinate reference.

g) **nssr**: A DSE that holds a non-specific knowledge attribute to represent a non-specific subordinate reference.

h) **supr**: A DSE that holds a specific knowledge attribute to represent the DSAs superior reference.

i) **xr**: A DSE that holds a specific knowledge attribute to represent a cross reference.

j) **admPoint**: A DSE corresponding to an administrative point.

k) **subentry**: A DSE that holds a subentry.

l) **shadow**: A DSE that holds a shadow-copy of an entry (or part of an entry) or other information (e.g. knowledge) received from a shadow-supplier; this named bit is set by the shadow consumer.

m) **immSupr**: A DSE that holds a specific knowledge attribute to represent a immediate superior reference.

n) **rhob**: A DSE that holds administrative point and subentry information received from a superior DSA in a Relevant Hierarchical Operational Binding (i.e. in either a Hierarchical Operational Binding or a Non-specific Hierarchical Binding as described in clauses 25 and 26 of ITU-T Rec. X.518 | ISO/IEC 9594-4).

o) **sa**: A qualifier of a subr DSE indicating that the subordinate naming context entry is an alias.

p) **dsSubentry**: A DSE that holds a DSA specific subentry.

The use of this operational attribute to represent aspects of the DSA information model is described in clause 21.

22 Representation of DSA Information

This clause treats the representation of DSA information. It describes the representation of DSA operational information (knowledge), Directory user information and Directory operational information.

22.1 Representation of Directory User and Operational Information

This clause specifies the representation of Directory user and Directory operational information in the DSA information model.

22.1.1 Object Entry

An object entry is represented by a DSE of type **entry** which contains the user and Directory operational attributes associated with the Directory entry. The name of the DSE is the name of the object entry (i.e. the object's distinguished name).

If the DSE holds a copy of the entry, the DSE type includes **shadow**.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

If the name of the object entry includes any alternative distinguished names differentiated by context, the name of the DSE may also include those alternative distinguished names differentiated by context. In the case of a DSE that holds a shadow of the entry, the name of the DSE may include a subset of the alternative distinguished names. In the case of a DSE that is not a copy, the name of the DSE shall include all distinguished names.

NOTE – For consistency and interworking with pre-1997 DSAs, the name of a DSE holding a copy shall include at least the primary distinguished value of any naming attribute. Thus the copy has at least the primary distinguished name of the object entry. Name resolution is enhanced if every distinguished value (and thus every alternative distinguished name) is present.

22.1.2 Alias Entry

An alias entry is represented by a DSE of type **alias** which contains the attributes associated with the alias entry (i.e. the RDN attributes and the aliased object name attribute). The name of the DSE is the name of the alias entry.

If the DSE holds a copy of the alias entry, the DSE type includes **shadow**.

If the name of the alias entry includes any alternative distinguished names differentiated by context, the name of the DSE may also include those alternative distinguished names differentiated by context. In the case of a DSE that holds a shadow of the alias entry, the name of the DSE may include a subset of the alternative distinguished names. In the case of a DSE that is not a copy, the name of the DSE shall include all distinguished names.

NOTE – For consistency and interworking with pre-1997 DSAs, the name of a DSE holding a copy shall include at least the primary distinguished value of any naming attribute. Thus the copy has at least the primary distinguished name of the alias entry. Name resolution is enhanced if every distinguished value (and thus every alternative distinguished name) is present.

22.1.3 Administrative Point

An administrative point is represented by a DSE of type **admPoint** which contains the attributes associated with the administrative point. The name of the DSE is the name of the administrative point.

If the DSE represents an entry, the DSE type includes **entry**. If the DSE holds a copy of the administrative point information, the DSE type includes **shadow**.

If the name of the administrative point includes any alternative distinguished names differentiated by context, the name of the DSE may also include those alternative distinguished names differentiated by context. In the case of a DSE that holds a shadow of the administrative point, the name of the DSE may include a subset of the alternative distinguished names. In the case of a DSE that is not a copy, the name of the DSE shall include all distinguished names.

NOTE – For consistency and interworking with pre-1997 DSAs, the name of a DSE holding a copy shall include at least the primary distinguished value of any naming attribute. Thus the copy has at least the primary distinguished name of the administrative point. Name resolution is enhanced if every distinguished value (and thus every alternative distinguished name) is present.

22.1.4 Subentry

A subentry is represented by a DSE of type **subentry** which contains the operational and user information associated with the subentry. The name of the DSE is the name of the subentry.

If the DSE holds a copy of the subentry, the DSE type is **subentry** and **shadow**.

22.2 Representation of Knowledge References

A knowledge reference consists of a DSE of an appropriate type which holds a correspondingly appropriate DSA operational attribute and which is identified by a name bearing a defined relationship to the naming context held by the referenced DSA.

The name of this DSE shall be the primary distinguished name and may include alternative names and context information if they are present in the context prefix of the naming context held by the referenced DSA. In the case of a DSE that holds a shadow, the name of the DSE may include a subset of the alternative names. In the case of a DSE that is not a copy, the name of the DSE shall include all distinguished names.

NOTE – Name resolution is enhanced if every distinguished value (and thus every alternative distinguished name) is present.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

22.2.1 Knowledge Attribute Types

DSA operational attributes are defined in the DSA information model to express a DSA's:

- knowledge of its own access point;
- superior knowledge;
- specific knowledge (its subordinate references);
- non-specific knowledge (its non-specific subordinate references);
- knowledge of its supplier(s), optionally including the master, if it is a shadow consumer;
- knowledge of its consumer(s) if it is a shadow supplier; and
- knowledge of secondary shadows, if it is a shadow supplier.

Object Identifier values are assigned in Annex E for these operational attributes.

22.2.1.1 My Access Point

The **myAccessPoint** operational attribute type is used by a DSA to represent its own access point. It is a DSA-specific attribute. All DSAs shall hold this attribute in their root DSE. It is single-valued and managed by the DSA itself.

```
myAccessPoint  ATTRIBUTE  ::= {
        WITH SYNTAX                     AccessPoint
        EQUALITY MATCHING RULE          accessPointMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           dSAOperation
        ID                              id-doa-myAccessPoint }
```

The ASN.1 type **AccessPoint** is defined in ITU-T Rec. X.518 | ISO/IEC 9594-4. Its ASN.1 specification is reproduced here for the convenience of the reader.

```
AccessPoint  ::=  SET  {
        ae-title                [0]     Name,
        address                 [1]     PresentationAddress
        protocolInformation     [2]     SET OF ProtocolInformation  OPTIONAL }
```

NOTE – The **Name** in the **ae-title** may be the primary distinguished name or an alternative distinguished name; however, consistency and interworking with pre-1997 DSAs is enhanced if the primary distinguished name is used.

How a DSA obtains the information held in **myAccessPoint** is not described in the Directory Specifications.

The **myAccessPoint** attribute type is held in a DSE of type **root.**

The information held in **myAccessPoint** may be employed in the DOP when establishing or modifying an operational binding.

22.2.1.2 Superior Knowledge

The **superiorKnowledge** operational attribute type is used by a non-first level DSA to represent its superior reference. It is a DSA-specific attribute. All non-first level DSAs shall hold this attribute in their root DSE. It is single-valued and managed by the DSA itself.

```
superiorKnowledge  ATTRIBUTE  ::= {
        WITH SYNTAX                     AccessPoint
        EQUALITY MATCHING RULE          accessPointMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           dSAOperation
        ID                              id-doa-superiorKnowledge }
```

A DSA may acquire the information held in **superiorKnowledge** by means not described in the Directory Specifications. It might also construct it from its immediate superior references, e.g. from its immediate superior reference whose context prefix has the least number of RDNs in its name.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The **superiorKnowledge** attribute type is held in a DSE of type **root.**

The information held in **superiorKnowledge** may be employed by a DSA when constructing a continuation reference returned in a DAP or DSP referral or when performing chaining.

22.2.1.3 Specific Knowledge

Specific knowledge consists of the access points for the master DSA of a naming context and/or shadow DSAs for that naming context. It is specific because the context prefix of the naming context is known and associated with the access point information. Specific knowledge is represented by the **specificKnowledge** operational attribute type. It is a DSA-shared attribute, is single-valued, and managed by the DSA itself.

```
specificKnowledge  ATTRIBUTE  ::= {
        WITH SYNTAX                     MasterAndShadowAccessPoints
        EQUALITY MATCHING RULE          masterAndShadowAccessPointsMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           distributedOperation
        ID                              id-doa-specificKnowledge }
```

The ASN.1 type **MasterAndShadowAccessPoints** is defined in ITU-T Rec. X.518 | ISO/IEC 9594-4. Its ASN.1 specification is reproduced here for the convenience of the reader.

```
MasterAndShadowAccessPoints  ::=  SET OF MasterOrShadowAccessPoint

MasterOrShadowAccessPoint  ::=  SET {
        COMPONENTS OF       AccessPoint,
        category                [3]     ENUMERATED {
                master                  (0),
                shadow                  (1) } DEFAULT master }
```

A DSA may acquire the information held in **specificKnowledge** by means not described in the Directory Specifications. In the case of a cross reference (DSE of type **xr**), it might also construct it from information received in the **crossReference** component of **ChainingResults** of a DSP reply. In the case of a subordinate reference (DSE of type **subr**), it might construct it from information received in the DOP when establishing or modifying a HOB.

The **specificKnowledge** attribute type is held in a DSE of type **subr**, **immSupr**, or **xr**. It is used by a DSA to represent subordinate, immediate superior and cross references.

The information held in **specificKnowledge** may be employed by a DSA when constructing a continuation reference returned in a DAP or DSP referral (or when performing chaining) and when constructing Shadowed DSA-Specific Entries (SDSEs) of type **subr**, **immSupr**, or **xr** provided in the DISP.

22.2.1.4 Non-Specific Knowledge

Non-specific knowledge consists of the access points for the master DSA of one or more naming contexts and/or shadow DSAs for the same one or more naming contexts. It is non-specific because the context prefixes of the naming context(s) is (are) not known. The immediate superior of the naming context(s) is known, however, and the access point information is associated with its name. Non-specific knowledge is represented by the **nonSpecificKnowledge** operational attribute type. It is a DSA-shared attribute, is multi-valued and managed by the DSA itself.

```
nonSpecificKnowledge  ATTRIBUTE  ::= {
        WITH SYNTAX                     MasterAndShadowAccessPoints
        EQUALITY MATCHING RULE          masterAndShadowAccessPointsMatch
        NO USER MODIFICATION            TRUE
        USAGE                           distributedOperation
        ID                              id-doa-nonSpecificKnowledge }
```

The **MasterAndShadowAccessPoints** value consists of an access point for a master DSA holding one or more subordinate naming contexts, and zero or more access points of DSAs holding shadows of some or all of these naming contexts.

A DSA may acquire the information held in **nonSpecificKnowledge** by means not described in the Directory Specifications. In the case of a non-specific subordinate reference (DSE of type **nssr**), it might also construct it from information received in the DOP when establishing or modifying a NHOB.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The **nonSpecificKnowledge** attribute type is held in a DSE of type **nssr**. It is used to represent non-specific subordinate references.

The information held in **nonSpecificKnowledge** may be employed by a DSA when constructing a continuation reference returned in a DAP or DSP referral (or when performing chaining) and when constructing SDSEs of type **nssr** provided in the DISP.

22.2.1.5 Supplier Knowledge

The supplier knowledge of a shadow consumer DSA consists of the access point(s) and shadowing agreement identifier(s) for its supplier(s) of a copy (or copies) of a replicated area. Optionally, if the supplier is not the master of the naming context from which a replicated area is derived, the access point of the master may be included in supplier knowledge. Supplier knowledge is represented by the **supplierKnowledge** operational attribute type. It is DSA-specific, multi-valued and managed by the DSA itself.

The ASN.1 syntax for a value of **supplierKnowledge** is **SupplierInformation**. A value of this attribute is composed of a shadow supplier DSA's access point and the agreement ID of the shadowing agreement between the supplier DSA and the consumer DSA holding the DSA-specific attribute (expressed as a value of the type **SupplierOrConsumer**), an indication of whether the supplier of the replicated area is or is not the master of the naming context from which it is derived, and, if not, optionally, the access point of the master DSA.

```
SupplierOrConsumer ::= SET {
        COMPONENTS OF           AccessPoint,        -- supplier or consumer --
        agreementID         [3] OperationalBindingID }

SupplierInformation ::= SET {
        COMPONENTS OF           SupplierOrConsumer, -- supplier --
        supplier-is-master  [4] BOOLEAN DEFAULT TRUE,
        non-supplying-master [5] AccessPoint OPTIONAL }

supplierKnowledge ATTRIBUTE ::= {
        WITH SYNTAX                     SupplierInformation
        EQUALITY MATCHING RULE          supplierOrConsumerInformationMatch
        NO USER MODIFICATION            TRUE
        USAGE                           dSAOperation
        ID                              id-doa-supplierKnowledge }
```

A DSA may acquire the information held in **supplierKnowledge** by means not described in the Directory Specifications. A shadow consumer DSA might also construct it from information received in the DOP when establishing or modifying a shadowing agreement.

The **supplierKnowledge** attribute type is held in a DSE of type **cp**. It is used to represent one or more supplier references. All shadow consumer DSAs shall hold a value of this attribute for each shadowing agreement they engage in as a consumer.

The information held in **supplierKnowledge** may be employed by a DSA when constructing a continuation reference returned in a DAP or DSP referral. The **agreementID** component (its type, **OperationalBindingID**, is defined in 25.2) of **supplierKnowledge** is required in the operations of the DOP for managing a shadowing agreement and in all the DISP operations.

22.2.1.6 Consumer Knowledge

The consumer knowledge of a shadow supplier DSA consists of the access point(s) and shadowing agreement identifier(s) for the consumer(s) of a copy (or copies) of a naming context provided to them by the supplier. Consumer knowledge is represented by the **consumerKnowledge** operational attribute type. It is DSA-specific, multi-valued and managed by the DSA itself.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The ASN.1 syntax for a value of **consumerKnowledge** is **ConsumerInformation** (which has the same syntax as **SupplierOrConsumer**, but refers to a consumer access point).

```
ConsumerInformation  ::=  SupplierOrConsumer        -- consumer --

consumerKnowledge  ATTRIBUTE  ::= {
        WITH SYNTAX                     ConsumerInformation
        EQUALITY MATCHING RULE          supplierOrConsumerInformationMatch
        NO USER MODIFICATION            TRUE
        USAGE                           dSAOperation
        ID                              id-doa-consumerKnowledge }
```

A DSA may acquire the information held in **consumerKnowledge** by means not described in the Directory Specifications. A shadow supplier DSA might also construct it from information received in the DOP when establishing or modifying shadowing agreements.

The **consumerKnowledge** attribute type is held in a DSE of type **cp**. It is used to represent one or more consumer references. All shadow supplier DSAs shall hold a value of this attribute for each shadowing agreement they engage in as a supplier.

The **agreementID** component of **consumerKnowledge** is required in the operations of the DOP for managing a shadowing agreement and in all the DISP operations.

22.2.1.7 Secondary Shadow Knowledge

Secondary shadow knowledge consists of information a supplier DSA (e.g. a master DSA) may choose to maintain regarding consumer DSAs that are engaged in secondary shadowing from its perspective. Secondary shadow knowledge is represented by the **secondaryShadows** operational attribute type. It is DSA-specific, multiple-valued and managed by the DSA itself. The ASN.1 syntax for a value of **secondaryShadows** is **SupplierAndConsumers**. It consists of the access point of a shadow supplier and a list of its direct consumers.

```
SupplierAndConsumers  ::=  SET {
        COMPONENTS OF           AccessPoint,        -- supplier --
        consumers               [3]     SET OF AccessPoint }

secondaryShadows  ATTRIBUTE  ::= {
        WITH SYNTAX                     SupplierAndConsumers
        EQUALITY MATCHING RULE          supplierAndConsumersMatch
        NO USER MODIFICATION            TRUE
        USAGE                           dSAOperation
        ID                              id-doa-secondaryShadows }
```

The **consumers** component of **SuppliersAndConsumers** contains only access points of DSAs that hold commonly usable copies of a replicated area.

A supplier DSA may obtain the information required to construct values of this attribute from a consumer DSA by following the procedure described in 23.1.1 of ITU-T Rec. X.518 | ISO/IEC 9594-4.

The **secondaryShadows** attribute type is held in a DSE of type **cp**.

Support for secondary shadow knowledge is optional.

22.2.1.8 Matching Rules

Four equality matching rules for the preceding knowledge attributes are specified below. They apply to attributes with syntaxes of types **AccessPoint**, **MasterAndShadowAccessPoints**, **SupplierInformation**, **ConsumerInformation** and **SuppliersAndConsumers**.

22.2.1.8.1 Access Point Match

The Access Point Match rule is specified as:

```
accessPointMatch  MATCHING-RULE  ::= {
        SYNTAX      Name
        ID          id-kmr-accessPointMatch }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The **accessPointMatch** matching rule applies to attribute values of type **AccessPoint.** A value of the assertion syntax is derived from a value of the attribute syntax by using the value of the **[0]** context specific tag (**Name**) component. Two values are considered to match for equality if the **Name** component of each match using the matching procedure for **DistinguishedName** values.

22.2.1.8.2 Master And Shadow Access Points Match

The Master and Shadow Access Point Match equality matching rule is specified as:

```
masterAndShadowAccessPointsMatch  MATCHING-RULE  ::= {
        SYNTAX      SET OF Name
        ID          id-kmr-masterShadowMatch }
```

The **masterAndShadowAccessPointsMatch** matching rule applies to attributes of type **MasterAndShadowAccessPoints**. A value of the assertion syntax is derived from a value of the attribute syntax by removing the **category** and **address** components of each **SET** in the **SET OF MasterOrShadowAccessPoints**. Two such values are considered to match for equality if both values have the same number of **SET OF** elements, and, after ordering the **SET OF** elements of each in any convenient fashion, the **ae-title** component of each pair of **SET OF** elements matches using the matching procedure for **distinguishedNameMatch**.

22.2.1.8.3 Supplier or Consumer Information Match

The Supplier or Consumer Information Match rule is specified as:

```
supplierOrConsumerInformationMatch  MATCHING-RULE  ::= {
        SYNTAX      SET {
                ae-title                [0]     Name,
                agreement-identifier    [2]     INTEGER }
        ID          id-kmr-supplierConsumerMatch }
```

The **supplierOrConsumerInformationMatch** matching rule applies to attribute values of type **SupplierInformation** or **ConsumerInformation** (and other attributes having values compatible with **SupplierInformation** or **ConsumerInformation**). A value of the assertion syntax is derived from a value of the attribute syntax by selecting the **SET** components with tags that match the **SET** components of the assertion syntax. Two such values are considered to match for equality if the **ae-title** component of each (after removing the explicit **[0]** tag information) matches using the matching procedure for **DistinguishedName** values and the **identifier** component contained in the **agreement** component of each (after removing the explicit **[2]** and **SEQUENCE** tag information) matches using the matching procedure for **INTEGER** values.

22.2.1.8.4 Suppliers And Consumers Match

The Supplier and Consumers Match rule is specified as:

```
supplierAndConsumersMatch  MATCHING-RULE  ::= {
        SYNTAX      Name
        ID          id-kmr-supplierConsumersMatch }
```

The Supplier and Consumers Match rule applies to attribute values of type **SupplierAndConsumers** (and other attributes having values compatible with **SupplierAndConsumers**). Two such values are considered to match for equality if the **ae-title** component of each (after removing the explicit **[0]** tag information) matches using the matching procedure for **DistinguishedName** values.

22.2.2 Knowledge Reference Types

This subclause specifies the representation of knowledge in the DSA information model.

22.2.2.1 Self Reference

A self reference represents a DSA's knowledge of its own access point. It is represented by a value of the attribute **myAccessPoint** held in the DSA's root DSE, a DSE of type **root**.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

22.2.2.2 Superior Reference

A superior reference is represented by a DSE of type **supr** and **root** which contains a **superiorKnowledge** attribute.

22.2.2.3 Immediate Superior Reference

An immediate superior reference is represented by a DSE of type **immSupr** which contains a **specificKnowledge** attribute. The name of the DSE holding the attribute corresponds to the context prefix of the naming context held by the referenced superior DSA.

Since a **specificKnowledge** attribute value may contain access points of several DSAs, it may therefore represent several immediate superior references, at most one of category **master** and zero or more of **category** shadow.

If the DSE holding the immediate superior reference is received from a shadow supplier, the DSE type includes **shadow**.

22.2.2.4 Subordinate Reference

A subordinate reference is represented by a DSE of type **subr** which contains a **specificKnowledge** attribute. The name of the DSE holding the attribute corresponds to the context prefix of the relevant naming context held by the referenced subordinate DSA.

Since a **specificKnowledge** attribute value may contain access points of several DSAs, it may therefore represent several subordinate references, at most one of category **master** and zero or more of **category** shadow.

If the DSE holding the subordinate reference is shadowed information, received from a shadow supplier, the DSE type includes **shadow**.

The DSE may also include **immSupr** in a DSA holding two naming contexts, one superior to the other, which are separated by a third single-entry naming context held in another DSA. An example of this situation is depicted in Annex M.

22.2.2.5 Non-Specific Subordinate Reference

A non-specific subordinate reference is represented by a DSE of type **nssr** (and **entry** normally) which contains a **nonSpecificKnowledge** attribute. The name of the DSE holding the attribute corresponds to the name formed by eliminating the final RDN of the context prefixes of the naming context held by the referenced subordinate DSAs.

Since a **nonSpecificKnowledge** attribute value may contain access points of several DSAs, it may therefore represent several non-specific subordinate references, at most one of category **master** and zero or more of **category** shadow. Each **nonSpecificKnowledge** attribute value represents a related set of non-specific subordinate references – the DSAs of category **shadow** hold one or more replicated areas derived from the naming context(s) held by the DSA of category **master**.

If the DSE holding the non-specific subordinate reference is shadowed information, received from a shadow-supplier, the DSE type includes **shadow**.

The DSE includes **shadow** in the situation of a shadow DSA when the DSE corresponds to an entry for which the master DSA has non-specific subordinate knowledge and for which only the **nonSpecificKnowledge** attribute for the non-specific subordinate reference is shadowed.

The DSE includes **cp** and **shadow** in the situation of a shadow DSA whose replicated area does not include the context prefix entry and the master DSA for the naming context has non-specific subordinate knowledge for the context prefix.

The DSE includes **admPoint** and **shadow** in the situation of a shadow DSA when the DSE corresponds to an administrative point, the entry information for the administrative point is not shadowed, and the master DSA for the naming context has non-specific subordinate knowledge for the administrative point.

When the administrative point coincides with a context prefix in the preceding two cases, the DSE may include **admPoint**, **cp** and **shadow**.

22.2.2.6 Cross Reference

A cross reference is represented by a DSE of type **xr** which contains a **specificKnowledge** attribute. The name of the DSE holding the attribute corresponds to the context prefix of the naming context held by the referenced DSA.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Since a **specificKnowledge** attribute value may contain access points of several DSAs, it may therefore represent several cross references, at most one of category **master** and zero or more of **category** shadow.

22.2.2.7 Supplier Reference

A supplier reference is represented by a DSE of type **cp** which contains a **supplierKnowledge** attribute. The name of the DSE holding the attribute corresponds to the context prefix of the shadowed naming context.

Since a **supplierKnowledge** attribute may have several values, it may represent several supplier references. Each attribute value represents one supplier reference.

22.2.2.8 Consumer Reference

A consumer reference is represented by a DSE of type **cp** which contains a **consumerKnowledge** attribute. The name of the DSE holding the attribute corresponds to the context prefix of the shadowed naming context.

Since a **consumerKnowledge** attribute may have several values, it may represent several consumer references. Each attribute value represents one consumer reference.

22.3 Representation of Names and Naming Contexts

22.3.1 Names and Glue DSEs

As described in 21.3, the minimum information that a DSA may associate with a name is the purpose for which it holds the name, represented by a DSE holding a value of the attribute **dseType**. When a DSE contains only such a minimal information, its DSE type shall be **glue**. In this case the DSE shall not hold an entry or subentry (or a shadow-copy of an entry or subentry) or a DSA-shared attribute.

Glue DSEs arise in the DSA information model to represent names that are known by a DSA as a consequence of holding information associated with other names. For example, consider the cross reference depicted in Figure 16. The DSA holding this cross reference also "knows" (in the sense described in 21.3) the names that are superior to the context prefix name associated with the cross reference. When no other information is associated with such superior names, they are represented in the DSA information model by glue DSEs.

22.3.2 Naming Contexts

A naming context consists of a context prefix, a subtree of zero or more entries subordinate to the context prefix (the root of the subtree), and, if there are naming contexts subordinate to it, subordinate and/or non-specific subordinate references sufficient to constitute full subordinate knowledge.

A context prefix is represented by a DSE of type **cp**. If the context prefix corresponds to an entry, the DSE type includes **entry**. If it corresponds to an alias, the DSE type includes **alias**. If the context prefix corresponds to an administrative point, the DSE type includes **admPoint**.

The subtree of entries and subentries subordinate to the context prefix is represented by DSEs as described in 22.1 to 22.4.

The representation of the subordinate knowledge of the naming context is represented by DSEs as described in 22.2.2.

A replicated area (a shadow-copy of all or part of a naming context) is represented as above except that the DSE type includes **shadow** in each DSE for which user or operational attributes are received from the shadow supplier. In the case of incomplete replicated areas, DSEs of type **glue** may occur to represent a bridge between the separate pieces of the shadowed information. No user or operational attributes are associated with these (or any) glue DSEs.

22.3.3 Example

Figure 16 illustrates an example of the mapping of a portion of the DIT (that corresponding to a naming context) onto the information tree of a DSA. In addition to the naming context information itself, the DSA's root DSE containing its superior reference (this is not the DSA information tree for a first level DSA), a glue DSE and a DSE representing a reference (either a cross reference or an immediate superior reference) to an immediately superior naming context are also depicted.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

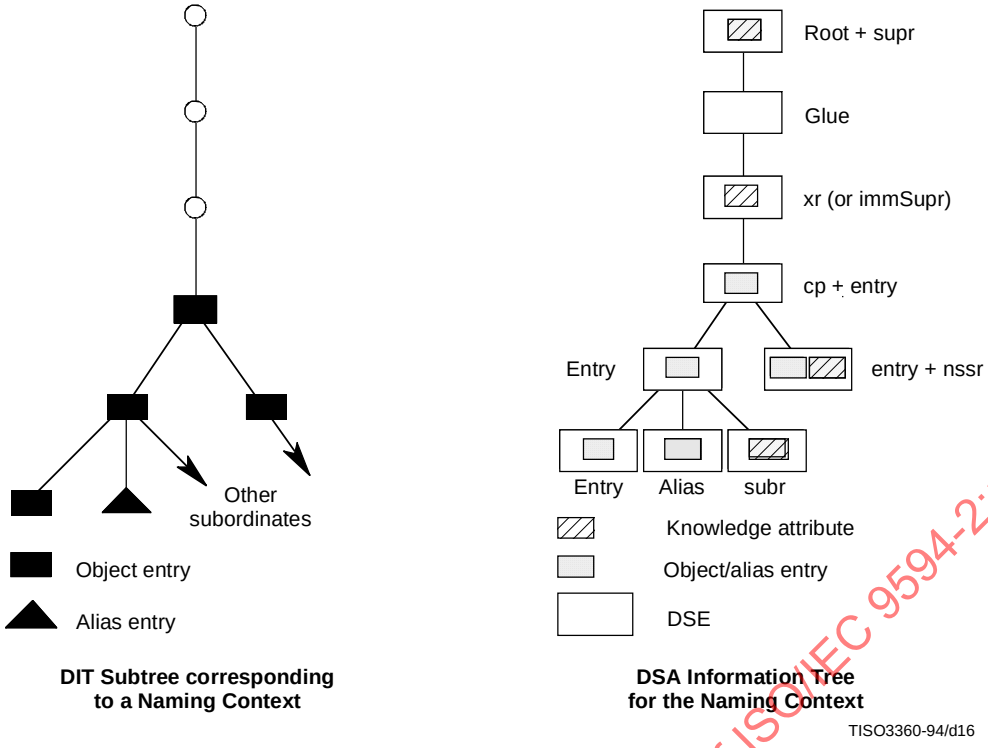


Figure 16 – DSEs for a Naming Context

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

SECTION 10 – DSA OPERATIONAL FRAMEWORK

23 Overview

23.1 Definitions

For the purposes of this Directory Specification, the following definitions apply:

23.1.1 directory operational framework: Provides the framework from which specific operational models concerned with particular aspects (e.g. shadowing or creating a naming context) of the operation of the components of the Directory (DSAs) may be derived by application of the framework. It factors out common elements which are present in all interactions between Directory components.

23.1.2 operational binding: A mutual understanding between two DSAs that, once established, expresses their "agreement" subsequently to engage in some sort of interaction.

23.1.3 operational binding type: A particular type of operational binding specified for some distinct purpose, that expresses the "agreement" of two DSAs to engage in specific types of interaction (e.g. shadowing).

23.1.4 operational binding instance: An operational binding of a specific type between two DSAs.

23.1.5 operational binding establishment: The process of establishing an operational binding instance.

23.1.6 operational binding modification: The process of modifying an operational binding instance.

23.1.7 operational binding termination: The process of terminating an operational binding instance.

23.1.8 operational binding management: The process of establishing, terminating or modifying an instance of an operational binding. This management may be achieved via information exchanges defined by Directory Specifications, via exchanges defined in other Specifications, or by other means.

23.1.9 cooperative state: With respect to a second DSA, the state of a DSA for which an operational binding instance has been established and has not been terminated.

23.1.10 non-cooperative state: With respect to a second DSA, the state of a DSA prior to the establishment or after the termination of an operational binding instance.

23.2 Introduction

The Directory Specifications define application protocol information exchanges and associated DSA procedures that define the distributed operation of the Directory. Clauses 23 through 26 define a DSA operational framework which models certain common elements in these information exchanges and procedures.

Two DSAs interact in a cooperative manner because, in addition to their technical capacity to exchange information and perform procedures associated with these exchanges, each has been configured to accept certain interactions with the other.

These clauses are concerned with the expression of a common framework for the specification of the structure of the elements of the cooperation between two DSAs.

One objective of this framework is that it be sufficiently general to account for all of the forms of DSA cooperation to be defined in this and future editions of the Directory Specifications. The framework is used within the Directory Specifications to define shadowing and hierarchical operational binding types.

24 Operational bindings

24.1 General

This clause is concerned with the definition of a general framework, the DSA operational framework, within which the specification of the nature of the cooperative interactions of components of the Directory (DSAs) may be structured in order to achieve a commonly agreed objective.

The general framework factors out common features which characterize all interactions between DSAs. By applying the DSA operational framework to specific aspects of cooperative interaction between DSAs, the resulting specifications will be both concise and consistent so that the overall number of mechanisms a DSA must support will be reduced.

The mutual understanding between two DSAs that, once established, expresses their "agreement" subsequently to engage in some sort of interaction is termed an *operational binding*. Two DSAs may share as many operational binding instances of a specific type as are required.

The DSA operational framework provides a common approach to the definition of an *operational binding type*. An operational binding type is a particular type of operational binding specified for some distinct purpose, that expresses the "agreement" of two DSAs to engage in specific types of interaction (e.g. shadowing). This interaction allows operations from a well-defined set to be invoked by one or the other party to the agreement.

Two particular DSAs that have reached such an "agreement" share an operational binding instance of a specific operational binding type. They are said to be in the *cooperative state* of that instance of an operational binding type.

Prior to the establishment or after the termination of an operational binding instance, two DSAs are said to be in the *non-cooperative state*.

Operational binding management is the process of establishing, terminating or modifying an instance of an operational binding. This management may be achieved via information exchanges defined by Directory Specifications, via exchanges defined in other Specifications, or by other means.

These general concepts are depicted in Figure 17.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

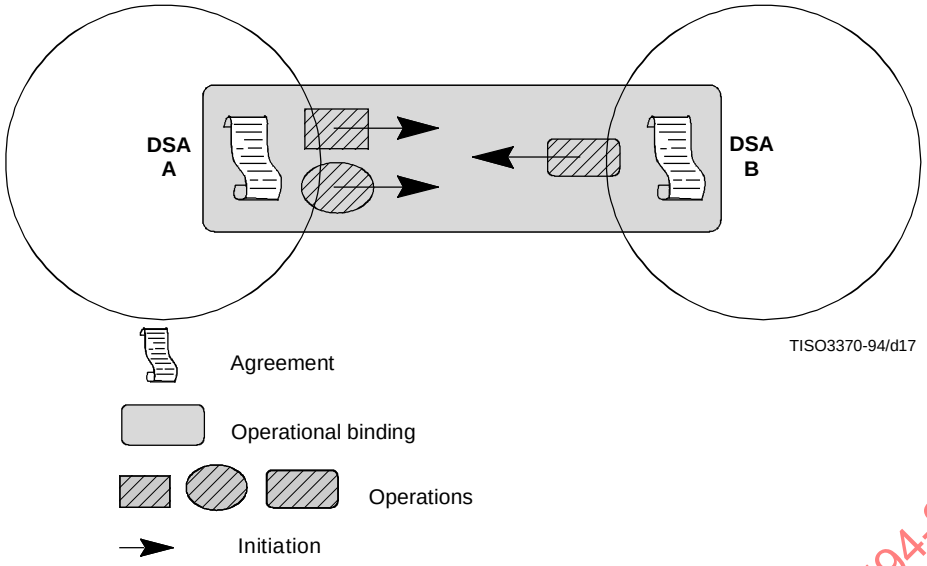


Figure 17 – An operational binding

24.2 Application of the operational framework

The application of the DSA operational framework to define an operational binding type is concerned with the following basic elements:

a) two DSAs;

b) an "agreement" of the service that one DSA will provide to another DSA;

c) a set of one or more operations, together with the accompanying procedures a DSA shall follow, through which the service can be realized;

d) a specification of the DSA interactions needed to manage the agreement.

The relationship of these basic elements is expressed by an operational binding. An operational binding comprises the set of these basic elements that are involved to represent the abstract agreement in technical terms. It represents the environment, governed by an "agreement", in which one DSA provides a defined service to the other (and vice versa).

24.2.1 Two DSAs

The DSA operational framework provides a structure within which the interaction of one DSA with another and the procedures they consequently execute may be specified.

The two DSAs may each play an identical role in the operational binding, in which case both DSAs may manage the operational binding, both DSAs may invoke the same operations on each other, and both DSAs are constrained to follow the same set of procedures. This is termed a symmetric operational binding.

Alternatively, each DSA may play a different role in the operational binding, so that different sets of operations and procedures apply to each DSA. Either or both of the DSAs may be involved in managing the operational binding. This is termed an asymmetric operational binding.

24.2.2 The agreement

An "agreement" is a mutual understanding reached between the administrative authorities of two DSAs about a service that shall be provided by one DSA to the other (and/or vice versa). The "agreement" is initially negotiated by the administrative authorities of the DSAs by means outside of the scope of the Directory Specifications.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Parameters of this "agreement" can be formalized by the recording in a DSA of an ASN.1 data type for use in a protocol exchange in the management of the operational binding. In this way both DSAs reach a mutual understanding of the service that each is providing to the other.

24.2.3 Operations

Operations are the basic medium that DSAs use to interact. A pair of DSAs will pass on one or more operations between themselves, in order to provide the agreed to service.

Whilst a DSA may be technically capable of supporting a large number of operations, it may only be willing to cooperate with another DSA in the processing of a small number of these operations, or in the processing of operations that only have particular values set for certain parameters.

The definition of an operational binding type requires the enumeration of the operations that can be exchanged. It also allows restrictions to be placed on the values of parameters defined within the operations.

24.2.4 Management of the agreement

The framework provides generic operations for managing an instance of an operational binding. These operations provide for the establishment, modification and termination of an operational binding.

The application of the framework to the specification of a particular operational binding type requires the initiator of each of the three management operations to be specified and also requires the procedures to be defined for each of establishment, modification and termination. Whenever a management operation is applied to an operational binding of the specified type, the DSA shall follow the corresponding procedure.

24.3 States of cooperation

The generic operational model defines two states of cooperation, as governed by an instance of a particular operational binding type, between two DSAs as seen by one DSA with respect to the other DSA and three transitions between these states. Each identified instance of an operational binding type shared by two DSAs has its own states of cooperation. The states of cooperation are:

a) *Non-cooperative state*: A particular identified instance of an operational binding type has not been established or has been terminated between the two DSAs. The interaction between the two DSAs (with respect to the identified instance of an operational binding type) is not defined. A DSA contacted by another with whom it is in a non-cooperative state may, for example, refuse to engage in any interaction at all, or it may be prepared to service the request.

b) *Cooperative state*: There is an instance of an operational binding of the type in question between the two DSAs. Their cooperative behaviour is governed by the definition of the operational binding type and its specific parameters and associated procedures.

The transitions between these two states of cooperation may be invoked in two ways: by standardized protocol interactions or by other means.

The interactions between two DSAs to manage an instance of an operational binding (e.g. to establish and terminate a shadowing agreement) are distinct from their potential interactions as governed by the binding (e.g. the interaction to update a unit of replication).

The state transitions are as follows:

a) The *establishment* transition creates an instance of an operational binding of a particular type between two DSAs, resulting in the movement from the non-cooperative to the cooperative state.

b) The *termination* transition destroys an instance of an operational binding of a particular type between two DSAs, resulting in the movement from the cooperative to the non-cooperative state.

c) The *modification* transition modifies the parameters of an instance of an operational binding between two DSAs, resulting in the movement from the cooperative state to the cooperative state.

These generic states and transitions are illustrated in Figure 18.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

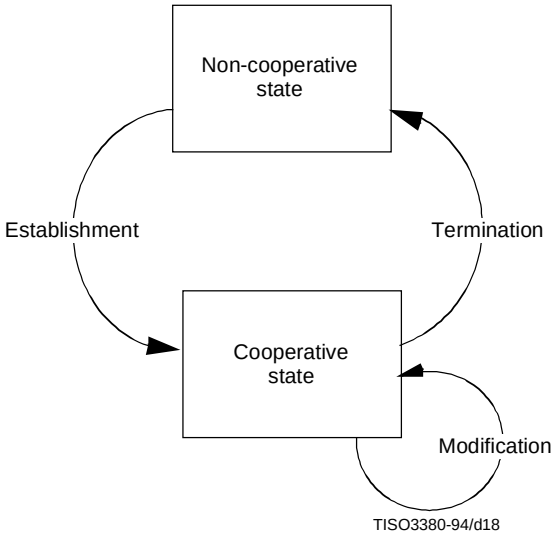


Figure 18 – States of cooperation

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

25 Operational binding specification and management

25.1 Operational binding type specification

When applying the framework to define a specific type of operational binding, the following characteristics of the type shall be specified:

a) *Symmetry*

A specification of the respective roles of the DSAs that are party to the operational binding.

Operational bindings may be symmetric, in which case the role of one DSA is interchangeable with the other and both DSAs exhibit the same external interactions. They may also be asymmetric, in which case each DSA plays a distinct role and both DSAs exhibit different external interactions. In this latter case, the Directory operational framework distinguishes the two abstract roles as "ROLE-A" and "ROLE-B".

Each of the abstract roles "ROLE-A" and "ROLE-B" have to be associated with a concrete role with defined semantics (e.g. "ROLE-A" as shadow supplier, "ROLE-B" as shadow consumer).

b) *Agreement*

A definition of the semantics and representation of the components of the "agreement". This information parameterizes the specific instance of an operational binding between two DSAs.

c) *Initiator*

A definition which of the two abstract roles "ROLE-A" and "ROLE-B" is allowed to initiate the establishment, modification or termination of an operational binding of this type.

d) *Management procedures*

A set of procedures that a DSA shall follow when the operational binding of this type is established, modified or terminated.

e) *Type identification*

This identifies the type of DSA interaction that is determined by the operational binding. These identifiers are object identifier values.

f) *Application-contexts, operations and procedures*

This identifies the set of application-contexts whose operations (or a subset thereof) may be employed during the cooperative phase of the operational binding.

For each operation referenced by the operational binding type, a description of the procedures to be followed by a DSA if the operation is invoked is required (this may be done by reference to another part of these Directory Specifications).

For those operational bindings that are to be managed using the generic operational binding management operations provided in this clause, the binding type shall be specified using the three information object classes **OPERATIONAL-BINDING**, **OP-BIND-COOP** and **OP-BIND-ROLE** defined in this clause.

25.2 Operational binding management

In general, the management of an operational binding requires initially the establishment of an operational binding instance. This may optionally be followed by one or more modifications to some or all of the parameters of the initial agreement, and finally may involve the termination of the operational binding instance. The precise details of how an instance may be managed are defined during the definition of the operational binding type. This type definition requires the specification of:

a) the initiator of each of the management operations (this can be either, both, or neither of the two DSAs);

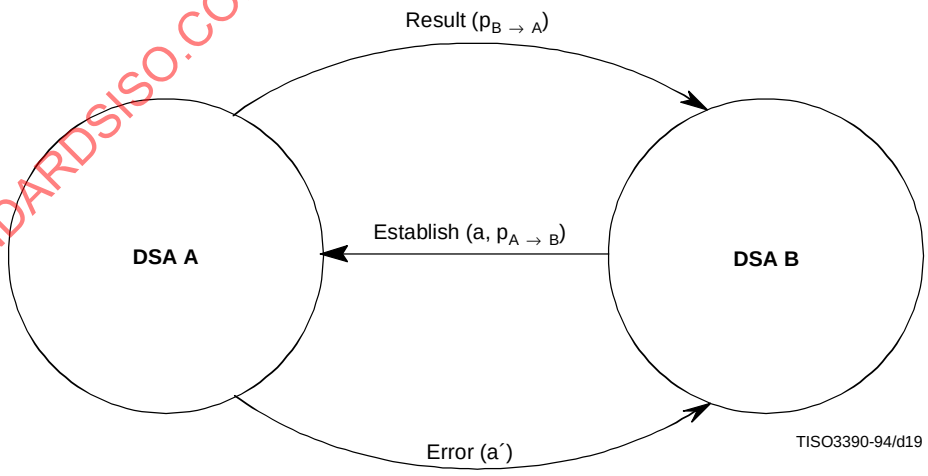
b) the parameters for each of the management operations; and

c) the procedures that each DSA must follow for each of the management operations.

During the establishment of an operational binding instance, an operational binding instance identifier (binding id) is created. This identifier, when combined with the distinguished names of the two DSAs involved in the operational binding, will form a unique identifier for the binding instance. All management operations subsequent to the establishment of the operational binding instance will use the binding id to identify which operational binding instance is being modified or terminated.

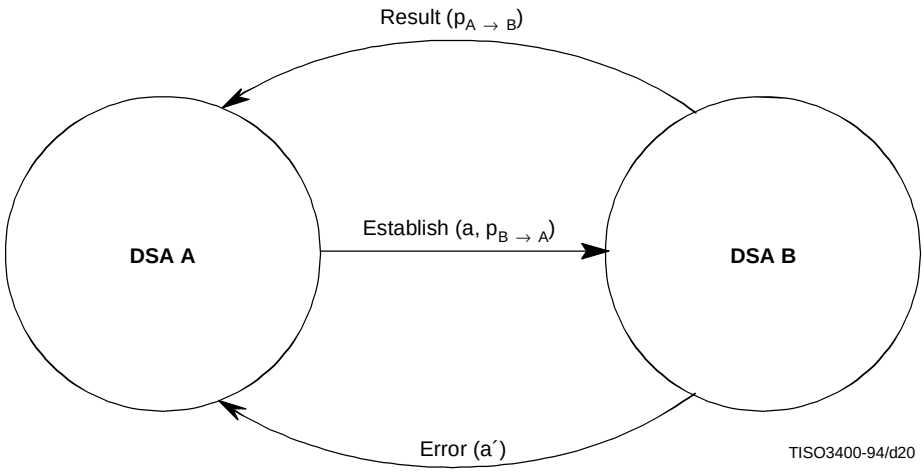
The initiator of the establish operation always transfers the parameters of the "agreement" to the second DSA. In addition, the initiator may also transfer some establishment parameters which are specific to its role in the operational binding. If the responding DSA is willing to enter into the operational binding, it may return in the result establishment parameters which are specific to its role. If the responding DSA is unwilling to enter into the operational binding, it shall return an error, which may optionally contain an agreement with a revised set of parameters. This is depicted in Figure 19 in the case where Role A and in Figure 20 in the case where Role B is the initiator of the establish operation.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998



a Agreement
b Establishment parameter

Figure 19 – DSA with Role A initiating establishment



a Agreement
b Establishment parameter

Figure 20 – DSA with Role B initiating establishment

25.3 Operational binding specification templates

For the definition of a specific type of operational binding, the following three ASN.1 information object classes may be used as templates. They allow those parts of the operational binding type that can be formalized to be specified by the use of ASN.1. Other aspects of the operational binding type, such as the procedures a DSA has to follow when an operational binding is established or terminated have to be specified by some other means (this can be done in a manner similar to the informal description of the DSA procedures during the name resolution process described in ITU-T Rec. X.518 | ISO/IEC 9594-4).

25.3.1 Operational binding information object class

```
OPERATIONAL-BINDING ::= CLASS {
        &Agreement,
        &Cooperation        OP-BINDING-COOP,
        &both               OP-BIND-ROLE OPTIONAL,
        &roleA              OP-BIND-ROLE OPTIONAL,
        &roleB              OP-BIND-ROLE OPTIONAL,
        &id                 OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
        AGREEMENT                   &Agreement
        APPLICATION CONTEXTS        &Cooperation
        [ SYMMETRIC                 &both ]
        [ ASYMMETRIC
                [ ROLE-A            &roleA ]
                ROLE-B              &roleB ] ]
        ID                          &id }
```

The OPERATIONAL-BINDING information object class serves as a specification template for an operational binding type. A variable notation is defined for this class to simplify its use as a template. The correspondence between the definition of an operational binding type and the fields of the variable notation is as follows:

a) The ASN.1 type of the agreement parameter that is used for this type of operational binding is that referenced by the "AGREEMENT" field.

b) The application contexts and the operations of these application-contexts that are employed within the cooperation phase of an operational binding instance of the defined type are those enumerated following the "APPLICATION-CONTEXTS" field. All operations of a listed application-context are selected unless the optional "APPLIES TO" field is present and followed by a list of references to operations that are selected from the application context. This list is an object class set composed of instances of the OPERATION information object class.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

c) The class of the operational binding is defined by the "SYMMETRIC" or "ASYMMETRIC" fields. In the case of a symmetric operational binding, the term "SYMMETRIC" is followed by a single information object of class OP-BIND-ROLE that is valid for both roles of the operational binding. In the case of an asymmetric operational binding, the term "ASYMMETRIC" is followed by two information objects of class OP-BIND-ROLE, one referenced by the subfield "ROLE-A" and the other by "ROLE-B".

d) The object identifier value that serves to identify this type of operational binding is defined by the "ID" field.

25.3.2 Operational binding cooperation information object class

```
OP-BINDING-COOP  ::=  CLASS {
        &applContext      APPLICATION-CONTEXT,
        &Operations       OPERATION OPTIONAL }
WITH SYNTAX {
        &applContext
        [ APPLIES TO      &Operations ] }
```

The OP-BIND-COOP information object class serves as a specification template for the identification of the operations of a named application context, some aspect of which is determined by the operational binding. An instance of this class is meaningful only within the context of a particular operational binding type. A variable notation is defined for this class to simplify its use as a template. The correspondence between the definition of an operational binding type and the fields of the variable notation is as follows:

a) The **applContext** field identifies an application context, some or all of whose operations are in some way determined by an operational binding.

b) The "APPLIES TO" field, if present, identifies the particular operations to which the operational binding applies. If the field is absent, the operational binding applies to all the operations of the application-context.

25.3.3 Operational binding role information object class

```
OP-BIND-ROLE  ::=  CLASS {
        &establish                  BOOLEAN  DEFAULT FALSE,
        &EstablishParam                      OPTIONAL,
        &modify                     BOOLEAN  DEFAULT FALSE,
        &ModifyParam                         OPTIONAL,
        &terminate                  BOOLEAN  DEFAULT FALSE,
        &TerminateParam                      OPTIONAL }
WITH SYNTAX {
        [ ESTABLISHMENT-INITIATOR       &establish ]
        [ ESTABLISHMENT-PARAMETER       &EstablishParam ]
        [ MODIFICATION-INITIATOR        &modify ]
        [ MODIFICATION-PARAMETER        &ModifyParam ]
        [ TERMINATION-INITIATOR         &terminate ]
        [ TERMINATION-PARAMETER         &TerminateParam ] }
```

The **OP-BIND-ROLE** information object class serves as a specification template for roles of an operational binding type. An instance of this class is meaningful only within the context of a particular operational binding type. A variable notation is defined for this class to simplify its use as a template. The correspondence between the definition of an operational binding role and the fields of the variable notation is as follows:

a) The "ESTABLISHMENT INITIATOR" field indicates whether the DSA assuming the defined role may initiate the establishment of an operational binding of a particular type.

b) The "ESTABLISHMENT PARAMETER" field defines the ASN.1 type exchanged by a DSA assuming the defined role when an instance of the operational binding type is established.

c) The "MODIFICATION INITIATOR" field indicates whether the DSA assuming the defined role may initiate the modification of an operational binding of a particular type.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

d) The "MODIFICATION PARAMETER" field defines the ASN.1 type exchanged by a DSA assuming the defined role when an instance of the operational binding type is modified.

e) The "TERMINATION INITIATOR" field indicates whether the DSA assuming the defined role may terminate the establishment of an operational binding of a particular type.

f) The "TERMINATION PARAMETER" field defines the ASN.1 type exchanged by a DSA assuming the defined role when an instance of the operational binding type is terminated.

26 Operations for operational binding management

This clause defines a set of operations that can be used to establish, modify and terminate operational bindings of various types. These operations are generic in the way that they can be used to manage operational bindings of any type. The specification of these operations makes use of the definitions provided for a certain type of operational binding by application of the **OPERATIONAL-BINDING** information object class template.

NOTE – By using this facility, arbitrary types of operational bindings may be managed. These operations (together with the associated application-context) provide a means of extensibility concerning DSA interactions. New types of operational bindings may be defined in the future which extend the functionality that is provided between DSAs.

26.1 Application-context definition

The set of operations for managing operational binding instances can be used for the definition of an application-context in the following two ways:

1) An application-context may be constructed containing only the operations for operational binding management. An application-context for generic operational binding management is defined in ITU-T Rec. X.519 | ISO/IEC 9594-5.

   The operations that may be exchanged during the cooperative phase of the operational binding form one or more separate application-contexts.

2) The set of operations can be imported into the module used to define a specific application-context. The operational binding management operations can then be used together with the operations of the cooperative phase within a single application-context.

NOTE – The first approach is useful in the case where a specialized component of a DSA wants to use an association solely for managing the set of operational bindings of that DSA, and it is not prepared to accept any of the operations defined for the co-operative phase (e.g. updateShadow).

26.2 Establish Operational Binding operation

The Establish Operational Binding operation allows establishment of an operational binding instance of a predefined type, between two DSAs. This is achieved through the transfer of the establishment parameters and the terms of agreement which were defined in the definition of the operational binding type. The arguments of the operation may be signed, encrypted, or signed and encrypted (see 15.3) by the requestor. If so requested, the responder may sign, encrypt, or sign and encrypt the results.

In the case of a symmetrical operational binding, either of the two DSAs may take the initiative to establish an operational binding instance of the predefined type.

In the case of an asymmetrical operational binding, either the DSA assuming "ROLE-A" or "ROLE-B" establishes the operational binding, depending on the specific definition of the operational binding type.

```
establishOperationalBinding  OPERATION  ::= {
        ARGUMENT        EstablishOperationalBindingArgument
        RESULT          EstablishOperationalBindingResult
        ERRORS          { operationalBindingError | securityError | serviceError }
        CODE            id-op-establishOperationalBinding }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
EstablishOperationalBindingArgument ::= OPTIONALLY-PROTECTED { SEQUENCE {
        bindingType             [0]   OPERATIONAL-BINDING.&id ({OpBindingSet}),
        bindingID               [1]   OperationalBindingID OPTIONAL,
        accessPoint             [2]   AccessPoint,
            -- symmetric, Role A initiates, or Role B initiates --
        initiator CHOICE {
            symmetric                 [3]   OPERATIONAL-BINDING.&both.&EstablishParam
                                              ({OpBindingSet}{@bindingType}),
            roleA-initiates           [4]   OPERATIONAL-BINDING.&roleA.&EstablishParam
                                              ({OpBindingSet}{@bindingType}),
            roleB-initiates           [5]   OPERATIONAL-BINDING.&roleB.&EstablishParam
                                              ({OpBindingSet}{@bindingType}) } OPTIONAL,
        agreement               [6]   OPERATIONAL-BINDING.&Agreement
                                          ({OpBindingSet}{@bindingType}),
        valid                   [7]   Validity DEFAULT { },
        securityParameters      [8]   SecurityParameters OPTIONAL },
        DIRQOP.&dopEstablishOpBindArg-QOP{@dirqop} }

OpBindingSet    OPERATIONAL-BINDING ::= {
        shadowOperationalBinding |
        hierarchicalOperationalBinding |
        nonSpecificHierarchicalOperationalBinding }

OperationalBindingID ::= SEQUENCE {
        identifier    INTEGER,
        version       INTEGER }
```

The component **bindingType** states which type of operational binding is to be established. Operational binding types are defined by the use of the **OPERATIONAL-BINDING** information object class template which assigns an object identifier value to the operational binding type. The **bindingType** is taken from the "ID" field of one of the instances of an operational binding type referenced by **OpBindingSet**. This set is a parameter of **EstablishOperationalBindingArgument**, a parameterized type.

The initiating DSA may assign an identification to the operational binding instance via the **bindingID** component. If **bindingID** is absent within the operation argument, the responding DSA shall assign an ID to the operational binding instance and return it in the **bindingID** component of the **establishOperationalBindingResult**. In either case, when establishing an operational binding, both the **identifier** and **version** components of the **OperationalBindingID** value must be assigned and issued by the DSA making the assignment.

The component **accessPoint** specifies the access point of the initiator for subsequent interactions.

The role that the DSA issuing the Establish Operational Binding operation assumes is indicated by the **CHOICE** type with the options **symmetric**, **roleA-initiates**, and **roleB-initiates**. The **CHOICE** option governs the particular establishment parameters employed by the initiating and responding DSAs. The semantics of the roles are defined as part of the definition of the operational binding type. The ASN.1 type of the **CHOICE** is determined by the "ESTABLISHMENT PARAMETER" of the initiator's **OP-BIND-ROLE** information object class template. The **CHOICE** type is omitted if establishment of the operational binding type requires no establishment parameter from the initiator.

The component **agreement** contains the terms of agreement governing the operational binding instance. Its actual content depends on the type of operational binding to be established. The ASN.1 type for this parameter is defined by the "AGREEMENT" field of the **OPERATIONAL-BINDING** information object class template of the operational binding type.

The duration that the operational binding instance shall exist is defined in **valid**. The starting time of the existence of the operational binding instance is specified in **validFrom** and the time that the operational binding instance is terminated is given in **validUntil**.

```
Validity ::= SEQUENCE {
        validFrom   [0]   CHOICE {
            now           [0]   NULL,
            time          [1]   UTCTime } DEFAULT now : NULL,
        validUntil  [1]   CHOICE {
            explicitTermination       [0]   NULL,
            time                      [1]   UTCTime } DEFAULT explicitTermination : NULL }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

If the Establish Operational Binding operation succeeds, the following result is returned and, may be signed, encrypted, or signed and encrypted (see 15.3) by the responder.

```
EstablishOperationalBindingResult ::= OPTIONALLY-PROTECTED { SEQUENCE {
    bindingType         [0]  OPERATIONAL-BINDING.&id ({OpBindingSet}),
    bindingID           [1]  OperationalBindingID OPTIONAL,
    accessPoint         [2]  AccessPoint,
    -- symmetric, Role A replies , or Role B replies --
    initiator CHOICE {
        symmetric       [3]  OPERATIONAL-BINDING.&both.&EstablishParam
                             ({OpBindingSet}{@bindingType}),
        roleA-replies   [4]  OPERATIONAL-BINDING.&roleA.&EstablishParam
                             ({OpBindingSet}{@bindingType}),
        roleB-replies   [5]  OPERATIONAL-BINDING.&roleB.&EstablishParam
                             ({OpBindingSet}{@bindingType}) } OPTIONAL,
    COMPONENTS OF       CommonResults },
    DIRQOP.&dopEstablishOpBindRes-QOP{@dirqop} }
```

The **bindingType** component is contained within the result to indicate the type of operational binding for use within the **CHOICE** element. Its value is the same as that provided by the establishment initiator and is taken from the "ID" field of one of the instances of an operational binding type referenced by **OpBindingSet**. This set is a parameter of **EstablishOperationalBindingResult**, a parameterized type.

The identification of the established operational binding instance may be returned in **bindingID**. It shall be used to identify this operational binding instance in any subsequent Modify or Terminate Operational Binding operation, and may be used in any other operation that is executed within the cooperative phase of the established operational binding instance.

The component **accessPoint** specifies the access point of the responder for subsequent interactions.

The initiating DSA may assign an identification to the operational binding instance via the **bindingID** component. If **bindingID** is absent within the operation argument, the responding DSA shall assign an ID to the operational binding instance and return it in the **bindingID** component of the **establishOperationalBindingResult**.

The role that the DSA replying to the Establish Operational Binding operation assumes is indicated by the **CHOICE** type with the options **symmetric**, **roleA-initiates** and **roleB-initiates**. The semantics of the roles are defined as part of the definition of the operational binding type. The ASN.1 type of the **CHOICE** is determined by the "ESTABLISHMENT PARAMETER" of the responder's **OP-BIND-ROLE** information object class template. The **CHOICE** type is omitted if establishment of the operational binding type requires no establishment parameter from the responder.

26.3 Modify Operational Binding operation

The Modify Operational Binding operation is used to modify an established operational binding. The right to modify is indicated by the "MODIFICATION INITIATOR" field(s) within the definition of the operational binding type using the **OP-BIND-ROLE** and **OPERATIONAL-BINDING** information object class templates.

The components of an operational binding that can be modified are the content of the agreement for the operational binding and its period of validity. Further, a modification parameter can be specified by the initiating role. The arguments of the operation may be signed, encrypted, or signed and encrypted (see 15.3) by the requestor. If so requested, the responder may sign, encrypt, or sign and encrypt the result.

```
modifyOperationalBinding OPERATION ::= {
    ARGUMENT        ModifyOperationalBindingArgument
    RESULT          ModifyOperationalBindingResult
    ERRORS          { operationalBindingError | securityError | serviceError }
    CODE            id-op-modifyOperationalBinding }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
ModifyOperationalBindingArgument ::= OPTIONALLY-PROTECTED { SEQUENCE {
        bindingType         [0]   OPERATIONAL-BINDING.&id ({OpBindingSet}),
        bindingID           [1]   OperationalBindingID,
        accessPoint         [2]   AccessPoint OPTIONAL,
        -- symmetric, Role A initiates, or Role B initiates --
        initiator CHOICE {
                symmetric           [3]   OPERATIONAL-BINDING.&both.&ModifyParam
                                          ({OpBindingSet}{@bindingType}),
                roleA-initiates     [4]   OPERATIONAL-BINDING.&roleA.&ModifyParam
                                          ({OpBindingSet}{@bindingType}),
                roleB-initiates     [5]   OPERATIONAL-BINDING.&roleB.&ModifyParam
                                          ({OpBindingSet}{@bindingType}) } OPTIONAL,
        newBindingID        [6]   OperationalBindingID,
        newAgreement        [7]   OPERATIONAL-BINDING.&Agreement
                                  ({OpBindingSet}{@bindingType}) OPTIONAL,
        valid               [8]   Validity OPTIONAL,
        securityParameters
                            [9]   SecurityParameters OPTIONAL },
        DIRQOP.&dopModifyOpBindArg-QOP{@dirqop} }
```

The component **bindingType** states which type of operational binding is to be modified. The **bindingType** is taken from the "ID" field of one of the instances of an operational binding type referenced by **OpBindingSet**. This set is a parameter of **ModifyOperationalBindingArgument**, a parameterized type.

The identification of the operational binding instance to be modified is given by **bindingID**. The revised identifier of the operational binding instance is given by **newBindingID**. The **version** component of **newBindingID** must be greater than that of **bindingID**.

The optional component **accessPoint** is present if the initiator's access point for subsequent interactions is to be changed.

The role that the DSA issuing the Modify Operational Binding operation assumes is indicated by the **CHOICE** type with the options **symmetric**, **roleA-initiates** and **roleB-initiates**. The semantics of the roles are defined as part of the definition of the operational binding type. The ASN.1 type of the **CHOICE** is determined by the "MODIFICATION PARAMETER" of the initiator's **OP-BIND-ROLE** information object class template. The **CHOICE** type is omitted if modification of the operational binding type requires no modification parameter from the initiator.

The component **newAgreement**, if present, contains the modified terms of agreement governing the operational binding instance. The ASN.1 type for this parameter is defined by the "AGREEMENT" field of the **OPERATIONAL-BINDING** information object class template of the operational binding type. If **newAgreement** is not present, the parameters of the agreement are not changed by the operation.

The optional **valid** component may be used to indicate a revised period of validity for the altered agreement. If the **valid** component is absent, the **validFrom** component is presumed to have the value **now** and the **validUntil** component is assumed to be unchanged. If the **validFrom** component is present and refers to an instant of time in the future, the current agreement remains in effect until that time.

If the Modify Operational Binding operation succeeds, the following result is returned and, may be signed, encrypted, or signed and encrypted (see 15.3) by the responder:

```
ModifyOperationalBindingResult ::= CHOICE {
        null            [0]   NULL,
        protected       [1]   PROTECTED { SEQUENCE {
                newBindingID            OperationalBindingID,
                bindingType             OPERATIONAL-BINDING.&id ({OpBindingSet}),
                newAgreement            OPERATIONAL-BINDING.&Agreement
                                              ({OpBindingSet}{@bindingType}),
                valid                   Validity OPTIONAL,
                COMPONENTS OF           CommonResults },
        DIRQOP.&dopModifyOpBindRes-QOP{@dirqop} } }
```

It is not possible for the responding DSA to return the modification parameter defined for its role to the modification initiator.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

26.4 Terminate Operational Binding operation

The Terminate Operational Binding operation is used to request the termination of an established operational binding instance. The right to request termination is indicated by the "TERMINATION INITIATOR" field(s) within the definition of the operational binding type using the **OP-BIND-ROLE** and **OPERATIONAL-BINDING** information object class templates. The arguments of the operation may be signed, encrypted, or signed and encrypted (see 15.3) by the requestor. If so requested, the responder may sign, encrypt, or sign and encrypt the result.

```
terminateOperationalBinding  OPERATION  ::=  {
        ARGUMENT        TerminateOperationalBindingArgument
        RESULT          TerminateOperationalBindingResult
        ERRORS          { operationalBindingError | securityError | serviceError }
        CODE            id-op-terminateOperationalBinding }

TerminateOperationalBindingArgument  ::=  OPTIONALLY-PROTECTED { SEQUENCE {
        bindingType             [0]   OPERATIONAL-BINDING.&id ({OpBindingSet}),
        bindingID               [1]   OperationalBindingID,
        -- symmetric, Role A initiates, or Role B initiates --
        initiator CHOICE {
                symmetric               [2]   OPERATIONAL-BINDING.&both.&TerminateParam
                                              ({OpBindingSet}{@bindingType}),
                roleA-initiates         [3]   OPERATIONAL-BINDING.&roleA.&TerminateParam
                                              ({OpBindingSet}{@bindingType}),
                roleB-initiates         [4]   OPERATIONAL-BINDING.&roleB.&TerminateParam
                                              ({OpBindingSet}{@bindingType})}  OPTIONAL,
        terminateAt             [5]   UTCTime OPTIONAL,
        securityParameters      [6]   SecurityParameters OPTIONAL },
        DIRQOP.&dopTermOpBindArg-QOP{@dirqop} }
```

The component **bindingType** states which type of operational binding is to be terminated. The **bindingType** is taken from the "ID" field of one of the instances of an operational binding type referenced by **OpBindingSet**. This set is a parameter of **TerminateOperationalBindingArgument**, a parameterized type.

The identification of the operational binding instance to be terminated is given by **bindingID**. The **version** component present in the **bindingID** is ignored.

The role that the DSA issuing the Terminate Operational Binding operation assumes is indicated by the **CHOICE** type with the options **symmetric**, **roleA-initiates** and **roleB-initiates**. The semantics of the roles are defined as part of the definition of the operational binding type. The ASN.1 type of the **CHOICE** is determined by the "TERMINATION PARAMETER" of the initiator's **OP-BIND-ROLE** information object class template. The **CHOICE** type is omitted if termination of the operational binding type requires no termination parameter from the initiator.

If the operational binding is not to be terminated immediately, a delayed termination time can defined in **terminateAt**.

If the Terminate Operational Binding operation succeeds, the following result is returned and, may be signed, encrypted, or signed and encrypted (see 15.3) by the responder:

```
TerminateOperationalBindingResult  ::=  CHOICE  {
        null                    [0]   NULL,
        protected               [1]   PROTECTED { SEQUENCE {
                        bindingID               OperationalBindingID,
                        bindingType             OPERATIONAL-BINDING.&id ({OpBindingSet}),
                        terminateAt             GeneralizedTime OPTIONAL,
                        COMPONENTS OF           CommonResults },
        DIRQOP.&dopTermOpBindRes-QOP{@dirqop} } }
```

It is not possible for the responding DSA to return the termination parameter defined for its role to the termination initiator.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

26.5 Operational Binding Error

An Operational Binding Error reports a problem related to the usage of operations for management of operational bindings. The parameter of the *error* may be signed, encrypted, or signed and encrypted (see 15.3) by the responder.

```
operationalBindingError  ERROR  ::=  {
        PARAMETER       OPTIONALLY-PROTECTED {
                                OpBindingErrorParam,
                                DIRQOP.&dirErrors-QOP{@dirqop} }
        CODE            id-err-operationalBindingError }

OpBindingErrorParam  ::=  SEQUENCE {
        problem                 [0]     ENUMERATED {
                invalidID                       (0),
                duplicateID                     (1),
                unsupportedBindingType          (2),
                notAllowedForRole               (3),
                parametersMissing               (4),
                roleAssignment                  (5),
                invalidStartTime                (6),
                invalidEndTime                  (7),
                invalidAgreement                (8),
                currentlyNotDecidable           (9),
                modificationNotAllowed          (10) },
        bindingType             [1]     OPERATIONAL-BINDING.&id ({OpBindingSet}) OPTIONAL,
        agreementProposal       [2]     OPERATIONAL-BINDING.&Agreement
                                        ({OpBindingSet}{@bindingType}) OPTIONAL,
        retryAt                 [3]     UTCTime OPTIONAL,
        COMPONENTS OF                   CommonResults }
```

The values of **problem** have the following meanings:

a) **invalidID**: The operational binding ID given in the request is not known by the receiving DSA or is in the wrong state for the requested operation.

b) **duplicateID**: The operational binding ID given in the establishment request already exists at the responder. This may be caused by a prior attempt to establish an operational binding instance when the result was lost and initiator has repeated the establishment request.

c) **unsupportedBindingType**: The requested operational binding type is not supported by the DSA.

d) **notAllowedForRole**: A management operation on the operational binding instance has been requested which is not allowed for the role that the requestor plays (e.g. a Terminate Operational Binding operation has been issued by a DSA that takes a role which is not allowed to initiate the termination of the operational binding instance).

e) **parametersMissing**: Any required establishment or termination parameters that are defined for the type of operational binding are missing.

f) **roleAssignment**: The requested role assignment for an asymmetric operational binding instance has not been accepted.

g) **invalidStartTime**: The specified starting time for the operational binding instance has not been accepted.

h) **invalidEndTime**: The specified termination time for the operational binding instance has not been accepted.

i) **invalidAgreement**: The terms of agreement for the requested operational binding instance have not been accepted. The terms of agreement that would be accepted by the responding DSA can be returned in **agreementProposal**.

j) **currentlyNotDecidable**: The DSA is not able to decide on-line about the establishment or modification of the requested operational binding instance. A time when the request should be repeated can be given in **retryAt**.

k) **modificationNotAllowed**: The Modify Operational Binding operation is rejected since modification is not permitted for this binding instance.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

The **bindingType** component shall be the same as that transmitted by the invoker of the failed operational binding management operation.

The **agreementProposal** component shall only be used in response to an **EstablishOperationalBinding** operation to propose a revised set of agreement parameters as described in 25.2.

The **retryAt** component shall be used only in conjunction with the **problem** value **currentlyNotDecidable** to indicate a time when the **EstablishOperationalBinding** or **ModifyOperationalBinding** operation should be retried.

The **CommonResults** component (see 7.4 of ITU-T Rec. X.511 | ISO/IEC 9594-3) includes **SecurityParameters**. The **SecurityParameters** component (see 7.10 of ITU-T Rec. X.511 | ISO/IEC 9594-3) shall be included in the **CommonResults** if the parameter of the error is to be signed by the responder.

26.6 Operational Binding Management Bind and Unbind

The **DSAOperationalBindingManagementBind** and **DSAOperationalBindingManagementUnBind** operations, defined in 26.6.1 and 26.6.2, are used by a DSA at the beginning and end of a particular period of operational binding management activity.

Protection for the **dSAOperationalBindingManagementBind** and **dSAOperationalBindingManagementUnbind** shall be equivalent to the protection applied to the **DSABind** and **DSAUnbind** operations.

NOTE – The credentials required for authentication may be carried by the Security Exchange Service Element (see ITU-T Rec. X.519 | ISO/IEC 9594-5) in which case they are not present in the bind arguments or results.

26.6.1 DSA Operational Binding Management Bind

A **dSAOperationalBindingManagementBind** operation is used to begin a period of operational binding management.

dSAOperationalBindingManagementBind OPERATION ::= directoryBind

The components of the **dSAOperationalManagementBind** are identical to their counterparts in **directoryBind** (see ITU-T Rec. X.511 | ISO/IEC 9594-3) with the following differences.

NOTE – The credentials required for authentication may be carried by the Security Exchange Service Element (see ITU-T Rec. X.519 | ISO/IEC 9594-5) in which case they are not present in the bind arguments or results.

26.6.1.1 Initiator Credentials

The **Credentials** of the **DirectoryBindArgument** allows information identifying the AE-Title of the initiating DSA to be sent to the responding DSA. The AE-title shall be in the form of a Directory Distinguished Name.

26.6.1.2 Responder Credentials

The **Credentials** of the **DirectoryBindResult** allows information identifying the AE-Title of the responding DSA to be sent to the initiating DSA. The AE-title shall be in the form of a Distinguished Name.

26.6.2 DSA Operational Binding Management Unbind

A **dSAOperationalManagementUnbind** operation is used to end a period of providing operational binding management.

dSAOperationalBindingManagementUnbind OPERATION ::= directoryUnbind

There are no arguments, results or errors.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Annex A

Object identifier usage

(This annex forms an integral part of this Recommendation | International Standard)

This annex documents the upper reaches of the object identifier subtree in which all of the object identifiers assigned in the Directory Specifications 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-itu-t ds(5) module(1) usefulDefinitions(0) 3}
DEFINITIONS ::=
BEGIN

-- EXPORTS All --

-- The types and values defined in this module are exported for use in the other ASN.1 modules contained
-- within the Directory Specifications, and for the use of other applications which will use them to access
-- Directory services. Other applications may use them for their own purposes, but this will not constrain
-- extensions and modifications needed to maintain or improve the Directory service.

ID          ::=    OBJECT IDENTIFIER

ds    ID    ::=    {joint-iso-itu-t ds(5)}
-- categories of information object --
module                          ID    ::=    {ds 1}
serviceElement                  ID    ::=    {ds 2}
applicationContext              ID    ::=    {ds 3}
attributeType                   ID    ::=    {ds 4}
attributeSyntax                 ID    ::=    {ds 5}
objectClass                     ID    ::=    {ds 6}
-- attributeSet                 ID    ::=    {ds 7}
algorithm                       ID    ::=    {ds 8}
abstractSyntax                  ID    ::=    {ds 9}
-- object                       ID    ::=    {ds 10}
-- port                         ID    ::=    {ds 11}
dsaOperationalAttribute         ID    ::=    {ds 12}
matchingRule                    ID    ::=    {ds 13}
knowledgeMatchingRule           ID    ::=    {ds 14}
nameForm                        ID    ::=    {ds 15}
group                           ID    ::=    {ds 16}
subentry                        ID    ::=    {ds 17}
operationalAttributeType        ID    ::=    {ds 18}
operationalBinding              ID    ::=    {ds 19}
schemaObjectClass               ID    ::=    {ds 20}
schemaOperationalAttribute      ID    ::=    {ds 21}
administrativeRoles             ID    ::=    {ds 23}
accessControlAttribute          ID    ::=    {ds 24}
rosObject                       ID    ::=    {ds 25}
contract                        ID    ::=    {ds 26}
package                         ID    ::=    {ds 27}
accessControlSchemes            ID    ::=    {ds 28}
certificateExtension            ID    ::=    {ds 29}
managementObject                ID    ::=    {ds 30}
attributeValueContext           ID    ::=    {ds 31}
securityExchange                ID    ::=    {ds 32}
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

-- modules --

```
usefulDefinitions                ID  ::=  {module usefulDefinitions(0) 3}
informationFramework             ID  ::=  {module informationFramework(1) 3}
directoryAbstractService         ID  ::=  {module directoryAbstractService(2) 3}
distributedOperations            ID  ::=  {module distributedOperations(3) 3}
protocolObjectIdentifiers        ID  ::=  {module protocolObjectIdentifiers(4) 3}
selectedAttributeTypes           ID  ::=  {module selectedAttributeTypes(5) 3}
selectedObjectClasses            ID  ::=  {module selectedObjectClasses(6) 3}
authenticationFramework          ID  ::=  {module authenticationFramework(7) 3}
algorithmObjectIdentifiers       ID  ::=  {module algorithmObjectIdentifiers(8) 3}
directoryObjectIdentifiers       ID  ::=  {module directoryObjectIdentifiers(9) 3}
upperBounds                      ID  ::=  {module upperBounds(10) 3}
dap                              ID  ::=  {module dap(11) 3}
dsp                              ID  ::=  {module dsp(12) 3}
distributedDirectoryOIDs         ID  ::=  {module distributedDirectoryOIDs(13) 3}
directoryShadowOIDs              ID  ::=  {module directoryShadowOIDs(14) 3}
directoryShadowAbstractService   ID  ::=  {module directoryShadowAbstractService(15) 3}
disp                             ID  ::=  {module disp(16) 3}
dop                              ID  ::=  {module dop(17) 3}
opBindingManagement              ID  ::=  {module opBindingManagement(18) 3}
opBindingOIDs                    ID  ::=  {module opBindingOIDs(19) 3}
hierarchicalOperationalBindings  ID  ::=  {module hierarchicalOperationalBindings(20) 3}
dsaOperationalAttributeTypes     ID  ::=  {module dsaOperationalAttributeTypes(22) 3}
schemaAdministration             ID  ::=  {module schemaAdministration(23) 3}
basicAccessControl               ID  ::=  {module basicAccessControl(24) 3}
directoryOperationalBindingTypes ID  ::=  {module directoryOperationalBindingTypes(25) 3}
certificateExtensions            ID  ::=  {module certificateExtensions(26) 0}
directoryManagement              ID  ::=  {module directoryManagement(27) 1}
enhancedSecurity                 ID  ::=  {module enhancedSecurity (28) 1}
directorySecurityExchanges       ID  ::=  {module directorySecurityExchanges (29) 1}
```

-- synonyms --

```
id-oc          ID  ::=  objectClass
id-at          ID  ::=  attributeType
id-as          ID  ::=  abstractSyntax
id-mr          ID  ::=  matchingRule
id-nf          ID  ::=  nameForm
id-sc          ID  ::=  subentry
id-oa          ID  ::=  operationalAttributeType
id-ob          ID  ::=  operationalBinding
id-doa         ID  ::=  dsaOperationalAttribute
id-kmr         ID  ::=  knowledgeMatchingRule
id-soc         ID  ::=  schemaObjectClass
id-soa         ID  ::=  schemaOperationalAttribute
id-ar          ID  ::=  administrativeRoles
id-aca         ID  ::=  accessControlAttribute
id-ac          ID  ::=  applicationContext
id-rosObject   ID  ::=  rosObject
id-contract    ID  ::=  contract
id-package     ID  ::=  package
id-acScheme    ID  ::=  accessControlSchemes
id-ce          ID  ::=  certificateExtension
id-mgt         ID  ::=  managementObject
id-avc         ID  ::=  attributeValueContext
id-se          ID  ::=  securityExchange
```

-- obsolete module identifiers --

```
--  usefulDefinition     ID  ::=  {module 0}
--  informationFramework ID  ::=  {module 1}
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
--      directoryAbstractService                ID      ::=     {module 2}
--      distributedOperations                   ID      ::=     {module 3}
--      protocolObjectIdentifiers               ID      ::=     {module 4}
--      selectedAttributeTypes                  ID      ::=     {module 5}
--      selectedObjectClasses                   ID      ::=     {module 6}
--      authenticationFramework                 ID      ::=     {module 7}
--      algorithmObjectIdentifiers              ID      ::=     {module 8}
--      directoryObjectIdentifiers              ID      ::=     {module 9}
--      upperBounds                             ID      ::=     {module 10}
--      dap                                     ID      ::=     {module 11}
--      dsp                                     ID      ::=     {module 12}
--      distributedDirectoryObjectIdentifiers   ID      ::=     {module 13}

-- unused module identifiers --

--      directoryShadowOIDs                     ID      ::=     {module 14}
--      directoryShadowAbstractService          ID      ::=     {module 15}
--      disp                                    ID      ::=     {module 16}
--      dop                                     ID      ::=     {module 17}
--      opBindingManagement                     ID      ::=     {module 18}
--      opBindingOIDs                           ID      ::=     {module 19}
--      hierarchicalOperationalBindings         ID      ::=     {module 20}
--      dsaOperationalAttributeTypes            ID      ::=     {module 22}
--      schemaAdministration                    ID      ::=     {module 23}
--      basicAccessControl                      ID      ::=     {module 24}
--      operationalBindingOIDs                  ID      ::=     {module 25}

END
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Annex B

Information Framework in ASN.1

(This annex forms an integral part of this Recommendation | International Standard)

This annex provides a summary of all of the ASN.1 type, value and macro definitions contained in this Directory Specification. The definitions form the ASN.1 module **InformationFramework**.

```
InformationFramework {joint-iso-itu-t ds(5) module(1) informationFramework(1) 3}
DEFINITIONS ::=
BEGIN

-- EXPORTS All --

-- The types and values defined in this module are exported for use in the other ASN.1 modules contained
-- within the Directory Specifications, and for the use of other applications which will use them to access
-- Directory services. Other applications may use them for their own purposes, but this will not constrain
-- extensions and modifications needed to maintain or improve the Directory service.

IMPORTS
        id-oc, id-at, id-mr, id-oa, id-sc, id-ar, id-nf, selectedAttributeTypes, directoryAbstractService
                FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 3}

        commonName, generalizedTimeMatch, generalizedTimeOrderingMatch, booleanMatch,
        objectIdentifierFirstComponentMatch
                FROM SelectedAttributeTypes selectedAttributeTypes

        OPTIONALLY-SIGNED, TypeAndContextAssertion
                FROM DirectoryAbstractService directoryAbstractService ;

-- attribute data types --

Attribute  ::=  SEQUENCE {
        type                    ATTRIBUTE.&id ({SupportedAttributes}),
        values                  SET SIZE (0 .. MAX) OF ATTRIBUTE.&Type ({SupportedAttributes}{@type}),
        valuesWithContext       SET SIZE (1 .. MAX) OF SEQUENCE {
        value                   ATTRIBUTE.&Type ({SupportedAttributes}{@type}),
        contextList             SET SIZE (1 .. MAX) OF Context } OPTIONAL }

AttributeType  ::=  ATTRIBUTE.&id

AttributeValue  ::=  ATTRIBUTE.&Type

Context  ::=  SEQUENCE {
        contextType       CONTEXT.&id ({SupportedContexts}),
        contextValues     SET SIZE (1..MAX) OF CONTEXT.&Type ({SupportedContexts}{@contextType}),
        fallback          BOOLEAN DEFAULT FALSE }

AttributeValueAssertion  ::=  SEQUENCE {
        type                    ATTRIBUTE.&id ({SupportedAttributes}),
        assertion               ATTRIBUTE.&equality-match.&AssertionType ({SupportedAttributes}{@type}),
        assertedContexts        CHOICE {
                allContexts             [0]  NULL,
                selectedContexts        [1]  SET OF ContextAssertion } OPTIONAL }

ContextAssertion  ::=  SEQUENCE {
        contextType             CONTEXT.&id({SupportedContexts}),
        contextValues           SET SIZE (1..MAX) OF
                        CONTEXT.&Assertion ({SupportedContexts}{@contextType})}
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
-- Definition of the following information object set is deferred, perhaps to standardized
-- profiles or to protocol implementation conformance statements. The set is required to
-- specify a table constraint on the values component of Attribute, the value component
-- of AttributeTypeAndValue, and the assertion component of AttributeValueAssertion.

SupportedAttributes     ATTRIBUTE       ::=     { objectClass | aliasedEntryName, ... }


-- Definition of the following information object set is deferred, perhaps to standardized
-- profiles or to protocol implementation conformance statements. The set is required to
-- specify a table constraint on the context specifications

SupportedContexts  CONTEXT  ::=  { ... }

-- naming data types --

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

RDNSequence  ::=  SEQUENCE OF RelativeDistinguishedName

DistinguishedName  ::=  RDNSequence

RelativeDistinguishedName  ::=  SET SIZE (1..MAX) OF AttributeTypeAndDistinguishedValue

AttributeTypeAndDistinguishedValue  ::=  SEQUENCE  {
        type                    ATTRIBUTE.&id ({SupportedAttributes}),
        value                   ATTRIBUTE.&Type({SupportedAttributes}{@type}),
        primaryDistinguished    BOOLEAN DEFAULT TRUE,
        valuesWithContext       SET SIZE (1 .. MAX) OF SEQUENCE {
                distingAttrValue        ATTRIBUTE.&Type ({SupportedAttributes}{@type}) OPTIONAL,
                contextList             SET SIZE (1 .. MAX) OF Context } OPTIONAL }

-- subtree data types --

SubtreeSpecification  ::=  SEQUENCE {
        base                    [0]     LocalName DEFAULT { },
                                        COMPONENTS OF  ChopSpecification,
        specificationFilter     [4]     Refinement OPTIONAL }
        -- empty sequence specifies whole administrative area

LocalName  ::=  RDNSequence

ChopSpecification  ::=  SEQUENCE  {
        specificExclusions      [1]     SET OF CHOICE {
                chopBefore              [0]     LocalName,
                chopAfter               [1]     LocalName } OPTIONAL,
        minimum                 [2]     BaseDistance DEFAULT 0,
        maximum                 [3]     BaseDistance OPTIONAL }

BaseDistance  ::=  INTEGER (0 .. MAX)

Refinement  ::=  CHOICE {
        item            [0]     OBJECT-CLASS.&id,
        and             [1]     SET OF Refinement,
        or              [2]     SET OF Refinement,
        not             [3]     Refinement }


-- OBJECT-CLASS information object class specification --

OBJECT-CLASS  ::=     CLASS {
        &Superclasses           OBJECT-CLASS OPTIONAL,
        &kind                   ObjectClassKind DEFAULT structural,
        &MandatoryAttributes    ATTRIBUTE OPTIONAL,
        &OptionalAttributes     ATTRIBUTE OPTIONAL,
        &id                     OBJECT IDENTIFIER UNIQUE }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
WITH SYNTAX {
        [ SUBCLASS OF           &Superclasses ]
        [ KIND                  &kind ]
        [ MUST CONTAIN          &MandatoryAttributes ]
        [ MAY CONTAIN           &OptionalAttributes ]
        ID                      &id }

ObjectClassKind  ::=  ENUMERATED {
        abstract      (0),
        structural    (1),
        auxiliary     (2) }


-- object classes --

top  OBJECT-CLASS  ::=  {
        KIND                abstract
        MUST CONTAIN        { objectClass }
        ID                  id-oc-top }

alias  OBJECT-CLASS  ::=  {
        SUBCLASS OF         { top }
        MUST CONTAIN        { aliasedEntryName }
        ID                  id-oc-alias }


-- ATTRIBUTE information object class specification --

ATTRIBUTE  ::=  CLASS  {
        &derivation                     ATTRIBUTE OPTIONAL,
        &Type                           OPTIONAL, -- either &Type or &derivation required --
        &equality-match                 MATCHING-RULE OPTIONAL,
        &ordering-match                 MATCHING-RULE OPTIONAL,
        &substrings-match               MATCHING-RULE OPTIONAL,
        &single-valued                  BOOLEAN DEFAULT FALSE,
        &collective                     BOOLEAN DEFAULT FALSE,


        -- operational extensions --
        &no-user-modification           BOOLEAN DEFAULT FALSE,
        &usage                          AttributeUsage DEFAULT userApplications,
        &id                             OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
        [ SUBTYPE OF                    &derivation ]
        [ WITH SYNTAX                   &Type ]
        [ EQUALITY MATCHING RULE        &equality-match ]
        [ ORDERING MATCHING RULE        &ordering-match ]
        [ SUBSTRINGS MATCHING RULE      &substrings-match ]
        [ SINGLE VALUE                  &single-valued ]
        [ COLLECTIVE                    &collective ]
        [ NO USER MODIFICATION          &no-user-modification ]
        [ USAGE                         &usage ]
        ID                              &id }

AttributeUsage  ::=  ENUMERATED {
        userApplications        (0),
        directoryOperation      (1),
        distributedOperation    (2),
        dSAOperation            (3) }


-- attributes --

objectClass  ATTRIBUTE  ::=  {
        WITH SYNTAX                     OBJECT IDENTIFIER
        EQUALITY MATCHING RULE          objectIdentifierMatch
        ID                              id-at-objectClass }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
aliasedEntryName  ATTRIBUTE ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        SINGLE VALUE                    TRUE
        ID                              id-at-aliasedEntryName }


-- MATCHING-RULE information object class specification --

MATCHING-RULE  ::=  CLASS  {
        &AssertionType  OPTIONAL,
        &id             OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX  {
        [ SYNTAX        &AssertionType ]
        ID              &id }

-- matching rules --

objectIdentifierMatch  MATCHING-RULE  ::=  {
        SYNTAX    OBJECT IDENTIFIER
        ID        id-mr-objectIdentifierMatch }

distinguishedNameMatch  MATCHING-RULE  ::=  {
        SYNTAX    DistinguishedName
        ID        id-mr-distinguishedNameMatch }

-- NAME-FORM information object class specification --

NAME-FORM  ::=  CLASS  {
        &namedObjectClass       OBJECT-CLASS,
        &MandatoryAttributes    ATTRIBUTE,
        &OptionalAttributes     ATTRIBUTE OPTIONAL,
        &id                     OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX  {
        NAMES                   &namedObjectClass
        WITH ATTRIBUTES         &MandatoryAttributes
        [ AND OPTIONALLY        &OptionalAttributes ]
        ID                      &id }


-- STRUCTURE-RULE class and DIT structure rule data types --

STRUCTURE-RULE  ::=  CLASS {
        &nameForm               NAME-FORM,
        &SuperiorStructureRules STRUCTURE-RULE OPTIONAL,
        &id                     RuleIdentifier }
WITH SYNTAX {
        NAME FORM               &nameForm
        [ SUPERIOR RULES        &SuperiorStructureRules ]
        ID                      &id }

DITStructureRule  ::=  SEQUENCE {
        ruleIdentifier          RuleIdentifier ,
                                -- must be unique within the scope of the subschema
        nameForm                NAME-FORM.&id,
        superiorStructureRules  SET OF RuleIdentifier OPTIONAL }

RuleIdentifier  ::=  INTEGER


-- CONTENT-RULE class and DIT content rule data types --

CONTENT-RULE  ::=  CLASS {
        &structuralClass        OBJECT-CLASS.&id    UNIQUE,
        &Auxiliaries            OBJECT-CLASS        OPTIONAL,
        &Mandatory              ATTRIBUTE           OPTIONAL,
        &Optional               ATTRIBUTE           OPTIONAL,
        &Precluded              ATTRIBUTE           OPTIONAL }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
WITH SYNTAX {
        STRUCTURAL OBJECT-CLASS      &structuralClass
        [ AUXILIARY OBJECT-CLASSES   &Auxiliaries ]
        [ MUST CONTAIN               &Mandatory ]
        [ MAY CONTAIN                &Optional ]
        [ MUST-NOT CONTAIN           &Precluded ] }

DITContentRule ::= SEQUENCE {
        structuralObjectClass  OBJECT-CLASS.&id,
        auxiliaries            SET OF OBJECT-CLASS.&id  OPTIONAL,
        mandatory              [1]  SET OF ATTRIBUTE.&id  OPTIONAL,
        optional               [2]  SET OF ATTRIBUTE.&id  OPTIONAL,
        precluded              [3]  SET OF ATTRIBUTE.&id  OPTIONAL }

CONTEXT ::= CLASS {
        &Type,
        &Assertion        OPTIONAL,
        &id               OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
        WITH SYNTAX       &Type
        [ ASSERTED AS     &Assertion ]
        ID                &id }

DITContextUse ::= SEQUENCE {
        attributeType                ATTRIBUTE.&id,
        mandatoryContexts     [1]    SET OF CONTEXT.&id OPTIONAL,
        optionalContexts      [2]    SET OF CONTEXT.&id OPTIONAL }

DIT-CONTEXT-USE-RULE ::= CLASS {
        &attributeType               ATTRIBUTE.&id  UNIQUE,
        &Mandatory                   CONTEXT        OPTIONAL,
        &Optional                    CONTEXT        OPTIONAL }
WITH SYNTAX {
        ATTRIBUTE TYPE               &attributeType
        [ MANDATORY CONTEXTS         &Mandatory ]
        [ OPTIONAL CONTEXTS          &Optional ] }


-- system schema information objects --

-- object classes --

subentry OBJECT-CLASS ::= {
        SUBCLASS OF    { top }
        KIND           structural
        MUST CONTAIN   { commonName | subtreeSpecification }
        ID             id-sc-subentry }

subentryNameForm NAME-FORM ::= {
        NAMES                subentry
        WITH ATTRIBUTES      { commonName }
        ID                   id-nf-subentryNameForm }

accessControlSubentry OBJECT-CLASS ::= {
        KIND         auxiliary
        ID           id-sc-accessControlSubentry }

collectiveAttributeSubentry OBJECT-CLASS ::= {
        KIND         auxiliary
        ID           id-sc-collectiveAttributeSubentry }

contextAssertionSubentry OBJECT-CLASS ::= {
        KIND           auxiliary
        MUST CONTAIN   {contextAssertionDefaults}
        ID             id-sc-contextAssertionSubentry }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
-- attributes --

createTimestamp  ATTRIBUTE  ::= {
        WITH SYNTAX                     GeneralizedTime
                        -- as per 41.3 b) or c) of ITU-T Rec. X.680 | ISO/IEC 8824-1
        EQUALITY MATCHING RULE          generalizedTimeMatch
        ORDERING MATCHING RULE          generalizedTimeOrderingMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-createTimestamp }

modifyTimestamp  ATTRIBUTE  ::= {
        WITH SYNTAX                     GeneralizedTime
                        -- as per 41.3 b) or c) of ITU-T Rec. X.680 | ISO/IEC 8824-1
        EQUALITY MATCHING RULE          generalizedTimeMatch
        ORDERING MATCHING RULE          generalizedTimeOrderingMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-modifyTimestamp }

subschemaTimestamp  ATTRIBUTE  ::= {
        WITH SYNTAX                     GeneralizedTime
                        -- as per 41.3 b) or c) of ITU-T Rec.X. 680 | ISO/IEC 8824-1
        EQUALITY MATCHING RULE          generalizedTimeMatch
        ORDERING MATCHING RULE          generalizedTimeOrderingMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-subschemaTimestamp }

creatorsName  ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-creatorsName }

modifiersName  ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-modifiersName }

subschemaSubentryList  ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-subschemaSubentryList }

accessControlSubentryList  ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-accessControlSubentryList }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
collectiveAttributeSubentryList  ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-collectiveAttributeSubentryList }

contextDefaultSubentryList  ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-contextDefaultSubentryList }

hasSubordinates ATTRIBUTE ::= {
        WITH SYNTAX                     BOOLEAN
        EQUALITY MATCHING RULE          booleanMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-hasSubordinates }

administrativeRole  ATTRIBUTE  ::= {
        WITH SYNTAX                     OBJECT-CLASS.&id
        EQUALITY MATCHING RULE          objectIdentifierMatch
        USAGE                           directoryOperation
        ID                              id-oa-administrativeRole }

subtreeSpecification  ATTRIBUTE  ::= {
        WITH SYNTAX     SubtreeSpecification
        SINGLE VALUE    TRUE
        USAGE           directoryOperation
        ID              id-oa-subtreeSpecification }

collectiveExclusions  ATTRIBUTE  ::= {
        WITH SYNTAX                     OBJECT IDENTIFIER
        EQUALITY MATCHING RULE          objectIdentifierMatch
        USAGE                           directoryOperation
        ID                              id-oa-collectiveExclusions }

contextAssertionDefaults  ATTRIBUTE  ::= {
        WITH SYNTAX                     TypeAndContextAssertion
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-oa-contextAssertionDefault }


-- object identifier assignments --


-- object classes --

id-oc-top                       OBJECT IDENTIFIER    ::=    {id-oc 0}
id-oc-alias                     OBJECT IDENTIFIER    ::=    {id-oc 1}


-- attributes --

id-at-objectClass               OBJECT IDENTIFIER    ::=    {id-at 0}
id-at-aliasedEntryName          OBJECT IDENTIFIER    ::=    {id-at 1}


-- matching rules --

id-mr-objectIdentifierMatch     OBJECT IDENTIFIER    ::=    {id-mr 0}
id-mr-distinguishedNameMatch    OBJECT IDENTIFIER    ::=    {id-mr 1}
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
-- operational attributes --

id-oa-excludeAllCollectiveAttributes    OBJECT IDENTIFIER  ::=  {id-oa 0}
id-oa-createTimestamp                  OBJECT IDENTIFIER  ::=  {id-oa 1}
id-oa-modifyTimestamp                  OBJECT IDENTIFIER  ::=  {id-oa 2}
id-oa-creatorsName                     OBJECT IDENTIFIER  ::=  {id-oa 3}
id-oa-modifiersName                    OBJECT IDENTIFIER  ::=  {id-oa 4}
id-oa-administrativeRole               OBJECT IDENTIFIER  ::=  {id-oa 5}
id-oa-subtreeSpecification             OBJECT IDENTIFIER  ::=  {id-oa 6}
id-oa-collectiveExclusions             OBJECT IDENTIFIER  ::=  {id-oa 7}
id-oa-subschemaTimestamp               OBJECT IDENTIFIER  ::=  {id-oa 8}
id-oa-hasSubordinates                  OBJECT IDENTIFIER  ::=  {id-oa 9}
id-oa-subschemaSubentryList            OBJECT IDENTIFIER  ::=  {id-oa 10}
id-oa-accessControlSubentryList        OBJECT IDENTIFIER  ::=  {id-oa 11}
id-oa-collectiveAttributeSubentryList  OBJECT IDENTIFIER  ::=  {id-oa 12}
id-oa-contextDefaultSubentryList       OBJECT IDENTIFIER  ::=  {id-oa 13}
id-oa-contextAssertionDefault          OBJECT IDENTIFIER  ::=  {id-oa 14}

-- subentry classes --

id-sc-subentry                         OBJECT IDENTIFIER  ::=  {id-sc 0}
id-sc-accessControlSubentry            OBJECT IDENTIFIER  ::=  {id-sc 1}
id-sc-collectiveAttributeSubentry      OBJECT IDENTIFIER  ::=  {id-sc 2}
id-sc-contextAssertionSubentry         OBJECT IDENTIFIER  ::=  {id-sc 3}

--  Name forms --

id-nf-subentryNameForm                 OBJECT IDENTIFIER  ::=  {id-nf 16}

-- administrative roles --

id-ar-autonomousArea                   OBJECT IDENTIFIER  ::=  {id-ar 1}
id-ar-accessControlSpecificArea        OBJECT IDENTIFIER  ::=  {id-ar 2}
id-ar-accessControlInnerArea           OBJECT IDENTIFIER  ::=  {id-ar 3}
id-ar-subschemaAdminSpecificArea       OBJECT IDENTIFIER  ::=  {id-ar 4}
id-ar-collectiveAttributeSpecificArea  OBJECT IDENTIFIER  ::=  {id-ar 5}
id-ar-collectiveAttributeInnerArea     OBJECT IDENTIFIER  ::=  {id-ar 6}
id-ar-contextDefaultSpecificArea       OBJECT IDENTIFIER  ::=  {id-ar 7}

END
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Annex C

SubSchema Administration Schema in ASN.1

(This annex forms an integral part of this Recommendation | International Standard)

This annex contains the ASN.1 type, value and information object definitions for subschema administration in the form of an ASN.1 module, **SchemaAdministration**.

```
SchemaAdministration {joint-iso-itu-t ds(5) module(1) schemaAdministration(23) 3}
DEFINITIONS ::=
BEGIN

-- EXPORTS All --
-- The types and values defined in this module are exported for use in the other ASN.1 modules contained
-- within the Directory Specifications, and for the use of other applications which will use them to access
-- Directory services. Other applications may use them for their own purposes, but this will not constrain
-- extensions and modifications needed to maintain or improve the Directory service.

IMPORTS
        informationFramework, selectedAttributeTypes, upperBounds, id-soc, id-soa
                FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 3}

        OBJECT-CLASS, ATTRIBUTE, MATCHING-RULE, DITStructureRule, DITContentRule,
        ObjectClassKind, AttributeUsage, NAME-FORM, CONTEXT, objectIdentifierMatch
                FROM InformationFramework informationFramework

        DirectoryString {}, integerFirstComponentMatch, integerMatch,
        objectIdentifierFirstComponentMatch
                FROM SelectedAttributeTypes selectedAttributeTypes

        ub-schema
                FROM UpperBounds upperBounds;

-- types --

DITStructureRuleDescription  ::=  SEQUENCE {
        COMPONENTS OF       DITStructureRule,
        name                [1]     SET OF DirectoryString { ub-schema } OPTIONAL,
        description                 DirectoryString { ub-schema } OPTIONAL,
        obsolete                    BOOLEAN DEFAULT FALSE }

DITContentRuleDescription  ::=  SEQUENCE {
        COMPONENTS OF               DITContentRule,
        name                [4]     SET OF DirectoryString { ub-schema } OPTIONAL,
        description                 DirectoryString { ub-schema }OPTIONAL,
        obsolete                    BOOLEAN DEFAULT FALSE }

MatchingRuleDescription  ::=  SEQUENCE {
        identifier                  MATCHING-RULE.&id,
        name                        SET OF DirectoryString { ub-schema }    OPTIONAL,
        description                 DirectoryString { ub-schema }           OPTIONAL,
        obsolete                    BOOLEAN                                 DEFAULT FALSE,
        information         [0]     DirectoryString { ub-schema } }
                                    -- describes the ASN.1 syntax

AttributeTypeDescription  ::=  SEQUENCE  {
        identifier                  ATTRIBUTE.&id,
        name                        SET OF DirectoryString { ub-schema }    OPTIONAL,
        description                 DirectoryString { ub-schema }           OPTIONAL,
        obsolete                    BOOLEAN                                 DEFAULT FALSE,
        information         [0]     AttributeTypeInformation }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
AttributeTypeInformation ::= SEQUENCE {
        derivation              [0]   ATTRIBUTE.&id OPTIONAL,
        equalityMatch           [1]   MATCHING-RULE.&id              OPTIONAL,
        orderingMatch           [2]   MATCHING-RULE.&id              OPTIONAL,
        substringsMatch         [3]   MATCHING-RULE.&id              OPTIONAL,
        attributeSyntax         [4]   DirectoryString { ub-schema }OPTIONAL,
        multi-valued            [5]   BOOLEAN                        DEFAULT TRUE,
        collective              [6]   BOOLEAN                        DEFAULT FALSE,
        userModifiable          [7]   BOOLEAN                        DEFAULT TRUE,
        application                   AttributeUsage                 DEFAULT  userApplication }

ObjectClassDescription ::= SEQUENCE {
        identifier                    OBJECT-CLASS.&id,
        name                          SET OF DirectoryString { ub-schema }   OPTIONAL,
        description                   DirectoryString { ub-schema }          OPTIONAL,
        obsolete                      BOOLEAN                                DEFAULT FALSE,
        information             [0]   ObjectClassInformation }

ObjectClassInformation ::= SEQUENCE {
        subclassOf                    SET OF OBJECT-CLASS.&id        OPTIONAL,
        kind                          ObjectClassKind                DEFAULT structural,
        mandatories             [3]   SET OF ATTRIBUTE.&id           OPTIONAL,
        optionals               [4]   SET OF ATTRIBUTE.&id           OPTIONAL }

NameFormDescription ::= SEQUENCE {
        identifier                    NAME-FORM.&id,
        name                          SET OF DirectoryString { ub-schema } OPTIONAL,
        description                   DirectoryString { ub-schema } OPTIONAL,
        obsolete                      BOOLEAN DEFAULT FALSE,
        information             [0]   NameFormInformation }

NameFormInformation ::= SEQUENCE {
        subordinate                   OBJECT-CLASS.&id,
        namingMandatories             SET OF ATTRIBUTE.&id,
        namingOptionals SET OF ATTRIBUTE.&id OPTIONAL }

MatchingRuleUseDescription ::= SEQUENCE {
        identifier                    MATCHING-RULE.&id,
        name                          SET OF DirectoryString { ub-schema }   OPTIONAL,
        description                   DirectoryString { ub-schema }          OPTIONAL,
        obsolete                      BOOLEAN                                DEFAULT FALSE,
        information             [0]   SET OF ATTRIBUTE.&id }

ContextDescription ::= SEQUENCE {
        identifier                    CONTEXT.&id,
        name                          SET OF DirectoryString {ub-schema}     OPTIONAL,
        description                   DirectoryString { ub-schema }          OPTIONAL,
        obsolete                      BOOLEAN                                DEFAULT FALSE,
        information             [0]   ContextInformation }

ContextInformation ::= SEQUENCE {
        syntax                        DirectoryString { ub-schema } ,
        assertionSyntax               DirectoryString { ub-schema } OPTIONAL }

DITContextUseDescription ::= SEQUENCE {
        identifier                    ATTRIBUTE.&id,
        name                          SET OF DirectoryString { ub-schema }   OPTIONAL,
        description                   DirectoryString { ub-schema }          OPTIONAL,
        obsolete                      BOOLEAN                                DEFAULT FALSE,
        information             [0]   DITContextUseInformation }

DITContextUseInformation ::= SEQUENCE {
        mandatoryContexts       [1]   SET OF CONTEXT.&id OPTIONAL,
        optionalContexts        [2]   SET OF CONTEXT.&id  OPTIONAL }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
-- object classes --

subschema  OBJECT-CLASS  ::= {
        KIND                auxiliary
        MAY CONTAIN         {
                            dITStructureRules |
                            nameForms |
                            dITContentRules |
                            objectClasses |
                            attributeTypes |
                            contextTypes |
                            dITContextUse |
                            matchingRules |
                            matchingRuleUse }
        ID                  id-soc-subschema }

-- attributes --

dITStructureRules ATTRIBUTE  ::= {
        WITH SYNTAX                     DITStructureRuleDescription
        EQUALITY MATCHING RULE          integerFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-dITStructureRule }

dITContentRules  ATTRIBUTE  ::= {
        WITH SYNTAX                     DITContentRuleDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-dITContentRules }

matchingRules  ATTRIBUTE  ::= {
        WITH SYNTAX                     MatchingRuleDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-matchingRules }

attributeTypes ATTRIBUTE ::= {
        WITH SYNTAX                     AttributeTypeDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-attributeTypes }

objectClasses ATTRIBUTE  ::= {
        WITH SYNTAX                     ObjectClassDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-objectClasses }

nameForms  ATTRIBUTE  ::= {
        WITH SYNTAX                     NameFormDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-nameForms }

matchingRuleUse ATTRIBUTE  ::= {
        WITH SYNTAX                     MatchingRuleUseDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-matchingRuleUse }

structuralObjectClass ATTRIBUTE ::= {
        WITH SYNTAX                     OBJECT IDENTIFIER
        EQUALITY MATCHING RULE          objectIdentifierMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-soa-structuralObjectClass }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
governingStructureRule ATTRIBUTE  ::= {
        WITH SYNTAX                     INTEGER
        EQUALITY MATCHING RULE          integerMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-soa-governingStructureRule }

contextTypes ATTRIBUTE  ::= {
        WITH SYNTAX                     ContextDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-contextTypes }

dITContextUse ATTRIBUTE  ::= {
        WITH SYNTAX                     DITContextUseDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-dITContextUse }


-- object identifier assignments --

-- schema object classes --

id-soc-subschema                OBJECT IDENTIFIER   ::=   {id-soc 1}

-- schema operational attributes --

id-soa-dITStructureRule         OBJECT IDENTIFIER   ::=   {id-soa 1}
id-soa-dITContentRules          OBJECT IDENTIFIER   ::=   {id-soa 2}
id-soa-matchingRules            OBJECT IDENTIFIER   ::=   {id-soa 4}
id-soa-attributeTypes           OBJECT IDENTIFIER   ::=   {id-soa 5}
id-soa-objectClasses            OBJECT IDENTIFIER   ::=   {id-soa 6}
id-soa-nameForms                OBJECT IDENTIFIER   ::=   {id-soa 7}
id-soa-matchingRuleUse          OBJECT IDENTIFIER   ::=   {id-soa 8}
id-soa-structuralObjectClass    OBJECT IDENTIFIER   ::=   {id-soa 9}
id-soa-governingStructureRule   OBJECT IDENTIFIER   ::=   {id-soa 10}
id-soa-contextTypes             OBJECT IDENTIFIER   ::=   {id-soa 11}
id-soa-dITContextUse            OBJECT IDENTIFIER   ::=   {id-soa 12}

END
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Annex D

Basic Access Control in ASN.1

(This annex forms an integral part of this Recommendation | International Standard)

This annex provides a summary of all of the ASN.1 type and value definitions for Basic Access Control. The definitions form the ASN.1 module **BasicAccessControl**.

```
BasicAccessControl {joint-iso-itu-t ds(5) module(1) basicAccessControl(24) 3}
DEFINITIONS ::=
BEGIN

-- EXPORTS All --

-- The types and values defined in this module are exported for use in the other ASN.1 modules contained
-- within the Directory Specifications, and for the use of other applications which will use them to access
-- Directory services. Other applications may use them for their own purposes, but this will not constrain
-- extensions and modifications needed to maintain or improve the Directory service.

IMPORTS
        id-aca, id-acScheme, informationFramework, upperBounds, selectedAttributeTypes,
                FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 3}

        ATTRIBUTE, AttributeType, DistinguishedName, ContextAssertion, SubtreeSpecification,
        MATCHING-RULE, objectIdentifierMatch
                FROM InformationFramework informationFramework

        ub-tag
                FROM UpperBounds upperBounds
        UniqueIdentifier, NameAndOptionalUID, directoryStringFirstComponentMatch,
        DirectoryString
                FROM SelectedAttributeTypes selectedAttributeTypes;

-- types --

ACIItem ::= SEQUENCE {
        identificationTag               DirectoryString { ub-tag },
        precedence                      Precedence,
        authenticationLevel             AuthenticationLevel,
        itemOrUserFirst                 CHOICE {
            itemFirst           [0]     SEQUENCE {
                protectedItems              ProtectedItems,
                itemPermissions             SET OF ItemPermission },
            userFirst           [1]     SEQUENCE {
                userClasses                 UserClasses,
                userPermissions             SET OF UserPermission } } }

Precedence ::= INTEGER (0..255)

ProtectedItems ::= SEQUENCE {
        entry                           [0]  NULL                           OPTIONAL,
        allUserAttributeTypes           [1]  NULL                           OPTIONAL,
        attributeType                   [2]  SET OF AttributeType           OPTIONAL,
        allAttributeValues              [3]  SET OF AttributeType           OPTIONAL,
        allUserAttributeTypesAndValues  [4]  NULL                           OPTIONAL,
        attributeValue                  [5]  SET OF AttributeTypeAndValue   OPTIONAL,
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
        selfValue                   [6]   SET OF AttributeType          OPTIONAL,
        rangeOfValues               [7]   Filter                        OPTIONAL,
        maxValueCount               [8]   SET OF MaxValueCount          OPTIONAL,
        maxImmSub                   [9]   INTEGER                       OPTIONAL,
        restrictedBy                [10]  SET OF RestrictedValue        OPTIONAL,
        contexts                    [11]  SET OF ContextAssertion       OPTIONAL }

MaxValueCount ::= SEQUENCE {
        type            AttributeType,
        maxCount        INTEGER }

RestrictedValue ::= SEQUENCE {
        type            AttributeType,
        valuesIn        AttributeType }

UserClasses ::= SEQUENCE {
        allUsers        [0]   NULL                          OPTIONAL,
        thisEntry       [1]   NULL                          OPTIONAL,
        name            [2]   SET OF NameAndOptionalUID     OPTIONAL,
        userGroup       [3]   SET OF NameAndOptionalUID     OPTIONAL,
                              -- dn component must be the name of an
                              -- entry of GroupOfUniqueNames
        subtree         [4]   SET OF SubtreeSpecification   OPTIONAL }

ItemPermission ::= SEQUENCE {
        precedence              Precedence OPTIONAL,
                              -- defaults to precedence in ACIItem --
        userClasses             UserClasses,
        grantsAndDenials        GrantsAndDenials }

UserPermission ::= SEQUENCE {
        precedence              Precedence OPTIONAL,
                              -- defaults to precedence in ACIItem
        protectedItems          ProtectedItems,
        grantsAndDenials        GrantsAndDenials }

AuthenticationLevel ::= CHOICE {
        basicLevelsSEQUENCE {
            level               ENUMERATED { none (0), simple (1), strong (2) },
            localQualifier      INTEGER OPTIONAL },
        other           EXTERNAL }

GrantsAndDenials ::= BIT STRING {
        -- permissions that may be used in conjunction
        -- with any component of ProtectedItems

        grantAdd                (0),
        denyAdd                 (1),
        grantDiscloseOnError    (2),
        denyDiscloseOnError     (3),
        grantRead               (4),
        denyRead                (5),
        grantRemove             (6),
        denyRemove              (7),
        -- permissions that may be used only in conjunction
        -- with the entry component
        grantBrowse             (8),
        denyBrowse              (9),
        grantExport             (10),
        denyExport              (11),
        grantImport             (12),
        denyImport              (13),
        grantModify             (14),
        denyModify              (15),
        grantRename             (16),
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
        denyRename              (17),
        grantReturnDN           (18),
        denyReturnDN            (19),
        -- permissions that may be used in conjunction
        -- with any component, except entry, of ProtectedItems
        grantCompare            (20),
        denyCompare             (21),
        grantFilterMatch        (22),
        denyFilterMatch         (23) }

AttributeTypeAndValue ::= SEQUENCE {
        type        ATTRIBUTE.&id ({SupportedAttributes}),
        value       ATTRIBUTE.&Type({SupportedAttributes}{@type}) }


-- attributes --

accessControlScheme ATTRIBUTE ::= {
        WITH SYNTAX                     OBJECT IDENTIFIER
        EQUALITY MATCHING RULE          objectIdentifierMatch
        SINGLE VALUE                    TRUE
        USAGE                           directoryOperation
        ID                              id-aca-accessControlScheme }

prescriptiveACI ATTRIBUTE ::= {
        WITH SYNTAX                     ACIItem
        EQUALITY MATCHING RULE          directoryStringFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-aca-prescriptiveACI }

entryACI ATTRIBUTE ::= {
        WITH SYNTAX                     ACIItem
        EQUALITY MATCHING RULE          directoryStringFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-aca-entryACI }

subentryACI ATTRIBUTE ::= {
        WITH SYNTAX                     ACIItem
        EQUALITY MATCHING RULE          directoryStringFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-aca-subentryACI }

-- object identifier assignments --

-- attributes --

id-aca-accessControlScheme              OBJECT IDENTIFIER   ::=   { id-aca 1 }
id-aca-prescriptiveACI                  OBJECT IDENTIFIER   ::=   { id-aca 4 }
id-aca-entryACI                         OBJECT IDENTIFIER   ::=   { id-aca 5 }
id-aca-subentryACI                      OBJECT IDENTIFIER   ::=   { id-aca 6 }

-- access control schemes –

basicAccessControlScheme                OBJECT IDENTIFIER   ::=   { id-acScheme 1 }
simplifiedAccessControlScheme           OBJECT IDENTIFIER   ::=   { id-acScheme 2 }
rule-based-access-control               OBJECT IDENTIFIER   ::=   { id-acScheme 3 }
rule-and-basic-access-control           OBJECT IDENTIFIER   ::=   { id-acScheme 4 }
rule-and-simple-access-control          OBJECT IDENTIFIER   ::=   { id-acScheme 5 }

END
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Annex E

DSA Operational Attribute Types in ASN.1

(This annex forms an integral part of this Recommendation | International Standard)

This annex includes all of the ASN.1 type and value definitions contained in clauses 10 through 20 in the form of an ASN.1 module, **DSAOperationalAttributeTypes**.

```
DSAOperationalAttributeTypes {joint-iso-itu-t ds(5) module(1) dsaOperationalAttributeTypes(22) 3}
DEFINITIONS ::=
BEGIN

-- EXPORTS All --

-- The types and values defined in this module are exported for use in the other ASN.1 modules contained
-- within the Directory Specifications, and for the use of other applications which will use them to access
-- Directory services. Other applications may use them for their own purposes, but this will not constrain
-- extensions and modifications needed to maintain or improve the Directory service.

IMPORTS
        id-doa, id-kmr, informationFramework, distributedOperations, opBindingManagement,
        selectedAttributeTypes
                FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 3 }

        ATTRIBUTE, MATCHING-RULE, Name
                FROM InformationFramework informationFramework

        OperationalBindingID
                FROM OperationalBindingManagement opBindingManagement

        AccessPoint, MasterAndShadowAccessPoints
                FROM DistributedOperations distributedOperations

        bitStringMatch
                FROM SelectedAttributeTypes selectedAttributeTypes  ;

-- data types --

DSEType  ::=  BIT STRING {
        root            (0),        -- root DSE --
        glue            (1),        -- represents knowledge of a name only --
        cp              (2),        -- context prefix --
        entry           (3),        -- object entry --
        alias           (4),        -- alias entry --
        subr            (5),        -- subordinate reference --
        nssr            (6),        -- non-specific subordinate reference --
        supr            (7),        -- superior reference --
        xr              (8),        -- cross reference --
        admPoint        (9),        -- administrative point --
        subentry        (10),       -- subentry --
        shadow          (11),       -- shadow copy --
        immSupr         (13),       -- immediate superior reference --
        rhob            (14),       -- rhob information --
        sa              (15),       -- subordinate reference to alias entry --
        dsSubentry      (16) }      -- DSA-Specific subentry --

SupplierOrConsumer  ::=  SET  {
        COMPONENTS OF           AccessPoint,        -- supplier or consumer --
        agreementID     [3]     OperationalBindingID }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
SupplierInformation  ::=  SET {
        COMPONENTS OF                    SupplierOrConsumer,  -- supplier --
        supplier-is-master        [4]    BOOLEAN DEFAULT TRUE,
        non-supplying-master      [5]    AccessPoint OPTIONAL }

ConsumerInformation  ::=  SupplierOrConsumer        -- consumer --

SupplierAndConsumers  ::=  SET {
        COMPONENTS OF                    AccessPoint,        -- supplier --
        consumers                 [3]    SET OF AccessPoint }


-- attribute types --

dseType  ATTRIBUTE  ::=  {
        WITH SYNTAX                        DSEType
        EQUALITY MATCHING RULE             bitStringMatch
        SINGLE VALUE                       TRUE
        NO USER MODIFICATION               TRUE
        USAGE                              dSAOperation
        ID                                 id-doa-dseType }

myAccessPoint  ATTRIBUTE  ::=  {
        WITH SYNTAX                        AccessPoint
        EQUALITY MATCHING RULE             accessPointMatch
        SINGLE VALUE                       TRUE
        NO USER MODIFICATION               TRUE
        USAGE                              dSAOperation
        ID                                 id-doa-myAccessPoint }

superiorKnowledge  ATTRIBUTE  ::=  {
        WITH SYNTAX                        AccessPoint
        EQUALITY MATCHING RULE             accessPointMatch
        SINGLE VALUE                       TRUE
        NO USER MODIFICATION               TRUE
        USAGE                              dSAOperation
        ID                                 id-doa-superiorKnowledge }

specificKnowledge  ATTRIBUTE  ::=  {
        WITH SYNTAX                        MasterAndShadowAccessPoints
        EQUALITY MATCHING RULE             masterAndShadowAccessPointsMatch
        SINGLE VALUE                       TRUE
        NO USER MODIFICATION               TRUE
        USAGE                              distributedOperation
        ID                                 id-doa-specificKnowledge }

nonSpecificKnowledge  ATTRIBUTE  ::=  {
        WITH SYNTAX                        MasterAndShadowAccessPoints
        EQUALITY MATCHING RULE             masterAndShadowAccessPointsMatch
        NO USER MODIFICATION               TRUE
        USAGE                              distributedOperation
        ID                                 id-doa-nonSpecificKnowledge }

supplierKnowledge  ATTRIBUTE  ::=  {
        WITH SYNTAX                        SupplierInformation
        EQUALITY MATCHING RULE             supplierOrConsumerInformationMatch
        NO USER MODIFICATION               TRUE
        USAGE                              dSAOperation
        ID                                 id-doa-supplierKnowledge }

consumerKnowledge  ATTRIBUTE  ::=  {
        WITH SYNTAX                        ConsumerInformation
        EQUALITY MATCHING RULE             supplierOrConsumerInformationMatch
        NO USER MODIFICATION               TRUE
        USAGE                              dSAOperation
        ID                                 id-doa-consumerKnowledge }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
secondaryShadows  ATTRIBUTE  ::= {
        WITH SYNTAX                     SupplierAndConsumers
        EQUALITY MATCHING RULE          supplierAndConsumersMatch
        NO USER MODIFICATION            TRUE
        USAGE                           dSAOperation
        ID                              id-doa-secondaryShadows }


-- matching rules --

accessPointMatch  MATCHING-RULE  ::= {
        SYNTAX      Name
        ID          id-kmr-accessPointMatch }

masterAndShadowAccessPointsMatch  MATCHING-RULE  ::= {
        SYNTAX      SET OF Name
        ID          id-kmr-masterShadowMatch }

supplierOrConsumerInformationMatch  MATCHING-RULE  ::= {
        SYNTAX      SET {
                ae-title                [0]     Name,
                agreement-identifier    [2]     INTEGER }
        ID          id-kmr-supplierConsumerMatch }

supplierAndConsumersMatch  MATCHING-RULE  ::= {
        SYNTAX      Name
        ID          id-kmr-supplierConsumersMatch }

-- object identifier assignments --

-- dsa operational attributes --

id-doa-dseType                  OBJECT IDENTIFIER  ::=  {id-doa 0}
id-doa-myAccessPoint            OBJECT IDENTIFIER  ::=  {id-doa 1}
id-doa-superiorKnowledge        OBJECT IDENTIFIER  ::=  {id-doa 2}
id-doa-specificKnowledge        OBJECT IDENTIFIER  ::=  {id-doa 3}
id-doa-nonSpecificKnowledge     OBJECT IDENTIFIER  ::=  {id-doa 4}
id-doa-supplierKnowledge        OBJECT IDENTIFIER  ::=  {id-doa 5}
id-doa-consumerKnowledge        OBJECT IDENTIFIER  ::=  {id-doa 6}
id-doa-secondaryShadows         OBJECT IDENTIFIER  ::=  {id-doa 7}

-- knowledge matching rules --

id-kmr-accessPointMatch         OBJECT IDENTIFIER  ::=  {id-kmr 0}
id-kmr-masterShadowMatch        OBJECT IDENTIFIER  ::=  {id-kmr 1}
id-kmr-supplierConsumerMatch    OBJECT IDENTIFIER  ::=  {id-kmr 2}
id-kmr-supplierConsumersMatch   OBJECT IDENTIFIER  ::=  {id-kmr 3}

END
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Annex F

Operational Binding Management in ASN.1

(This annex forms an integral part of this Recommendation | International Standard)

This annex includes all of the ASN.1 type, value and information object class definitions regarding Operational Bindings relevant to this Directory Specification in the form of the ASN.1 module **OperationalBindingManagement**.

```
OperationalBindingManagement {joint-iso-itu-t ds(5) module(1) opBindingManagement(18) 3}
DEFINITIONS ::=
BEGIN

-- EXPORTS All --

-- The types and values defined in this module are exported for use in the other ASN.1 modules contained
-- within the Directory Specifications, and for the use of other applications which will use them to access
-- Directory services. Other applications may use them for their own purposes, but this will not constrain
-- extensions and modifications needed to maintain or improve the Directory service.

IMPORTS
        directoryShadowAbstractService, hierarchicalOperationalBindings,
        dop, directoryAbstractService, distributedOperations
                FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 3}

        shadowOperationalBinding
                FROM DirectoryShadowAbstractService directoryShadowAbstractService

        hierarchicalOperationalBinding, nonSpecificHierarchicalOperationalBinding
                FROM HierarchicalOperationalBindings hierarchicalOperationalBindings

        OPERATION, ERROR
                FROM Remote-Operations-Information-Objects
        {joint-iso-itu-t remote-operations(4) informationObjects(5) version1(0)}

        APPLICATION-CONTEXT
                FROM Remote-Operations-Information-Objects-extensions {joint-iso-itu-t
        remote-operations(4) informationObjects-extensions(8) version1(0)}

        id-op-establishOperationalBinding, id-op-modifyOperationalBinding,
        id-op-terminateOperationalBinding, id-err-operationalBindingError
                FROM DirectoryOperationalBindingManagementProtocol dop

        directoryBind, directoryUnbind, securityError, CommonResults, SecurityParameters
                FROM DirectoryAbstractService directoryAbstractService

        OPTIONALLY-PROTECTED, DIRQOP
                FROM EnhancedSecurity enhancedSecurity

        AccessPoint
                FROM DistributedOperations distributedOperations;

-- bind and unbind --

dSAOperationalBindingManagementBind OPERATION       ::=   directoryBind

dSAOperationalBindingManagementUnbind OPERATION  ::=      directoryUnbind

-- operations, arguments and results --

establishOperationalBinding  OPERATION  ::= {
        ARGUMENT        EstablishOperationalBindingArgument
        RESULT          EstablishOperationalBindingResult
        ERRORS          {operationalBindingError | securityError}
        CODE            id-op-establishOperationalBinding }
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
EstablishOperationalBindingArgument ::= OPTIONALLY-PROTECTED { SEQUENCE {
        bindingType             [0]   OPERATIONAL-BINDING.&id ({OpBindingSet}),
        bindingID               [1]   OperationalBindingID OPTIONAL,
        accessPoint             [2]   AccessPoint,
            -- symmetric, Role A initiates, or Role B initiates --
        initiator CHOICE {
            symmetric                 [3]   OPERATIONAL-BINDING.&both.&EstablishParam
                                            ({OpBindingSet}{@bindingType}),
            roleA-initiates           [4]   OPERATIONAL-BINDING.&roleA.&EstablishParam
                                            ({OpBindingSet}{@bindingType}),
            roleB-initiates           [5]   OPERATIONAL-BINDING.&roleB.&EstablishParam
                                            ({OpBindingSet}{@bindingType}) } OPTIONAL,
        agreement               [6]   OPERATIONAL-BINDING.&Agreement
                                        ({OpBindingSet}{@bindingType}),
        valid                   [7]   Validity DEFAULT { },
        securityParameters      [8]   SecurityParameters OPTIONAL },
        DIRQOP.&dopEstablishOpBindArg-QOP{@dirqop} }


OperationalBindingID ::= SEQUENCE {
        identifier    INTEGER,
        version       INTEGER }


Validity ::= SEQUENCE {
        validFrom   [0]   CHOICE {
            now           [0]   NULL,
            time          [1]   UTCTime } DEFAULT now : NULL,
        validUntil  [1]   CHOICE {
            explicitTermination       [0]   NULL,
            time                      [1]   UTCTime } DEFAULT explicitTermination : NULL }


EstablishOperationalBindingResult ::= OPTIONALLY-PROTECTED { SEQUENCE {
        bindingType       [0]   OPERATIONAL-BINDING.&id ({OpBindingSet}),
        bindingID         [1]   OperationalBindingID OPTIONAL,
        accessPoint       [2]   AccessPoint,
        -- symmetric, Role A replies , or Role B replies --
        initiator CHOICE {
            symmetric         [3]   OPERATIONAL-BINDING.&both.&EstablishParam
                                    ({OpBindingSet}{@bindingType}),
            roleA-replies     [4]   OPERATIONAL-BINDING.&roleA.&EstablishParam
                                    ({OpBindingSet}{@bindingType}),
            roleB-replies     [5]   OPERATIONAL-BINDING.&roleB.&EstablishParam
                                    ({OpBindingSet}{@bindingType}) } OPTIONAL,
        COMPONENTS OF             CommonResults },
        DIRQOP.&dopEstablishOpBindRes-QOP{@dirqop} }


modifyOperationalBinding OPERATION ::= {
        ARGUMENT        ModifyOperationalBindingArgument
        RESULT          ModifyOperationalBindingResult
        ERRORS          { operationalBindingError | securityError }
        CODE            id-op-modifyOperationalBinding }


ModifyOperationalBindingArgument ::= OPTIONALLY-PROTECTED { SEQUENCE {
        bindingType       [0]   OPERATIONAL-BINDING.&id ({OpBindingSet}),
        bindingID         [1]   OperationalBindingID,
        accessPoint       [2]   AccessPoint OPTIONAL,
        -- symmetric, Role A initiates, or Role B initiates --
        initiator CHOICE {
            symmetric         [3]   OPERATIONAL-BINDING.&both.&ModifyParam
                                    ({OpBindingSet}{@bindingType}),
            roleA-initiates   [4]   OPERATIONAL-BINDING.&roleA.&ModifyParam
                                    ({OpBindingSet}{@bindingType}),
            roleB-initiates   [5]   OPERATIONAL-BINDING.&roleB.&ModifyParam
                                    ({OpBindingSet}{@bindingType}) } OPTIONAL,
        newBindingID      [6]   OperationalBindingID,
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
        newAgreement    [7]    OPERATIONAL-BINDING.&Agreement
                               ({OpBindingSet}{@bindingType})  OPTIONAL,
        valid           [8]    Validity OPTIONAL,
        securityParameters
                        [9]    SecurityParameters OPTIONAL },
        DIRQOP.&dopModifyOpBindArg-QOP{@dirqop} }

ModifyOperationalBindingResult  ::=  CHOICE  {
        null            [0]    NULL,
        protected       [1]    PROTECTED { SEQUENCE {
                newBindingID            OperationalBindingID,
                bindingType             OPERATIONAL-BINDING.&id ({OpBindingSet}),
                newAgreement            OPERATIONAL-BINDING.&Agreement
                                            ({OpBindingSet}{@bindingType}),
                valid                   Validity OPTIONAL,
                COMPONENTS OF           CommonResults },
        DIRQOP.&dopModifyOpBindRes-QOP{@dirqop} } }

terminateOperationalBinding  OPERATION  ::=  {
        ARGUMENT        TerminateOperationalBindingArgument
        RESULT          TerminateOperationalBindingResult
        ERRORS          {operationalBindingError | securityError}
        CODE            id-op-terminateOperationalBinding }

TerminateOperationalBindingArgument  ::=  OPTIONALLY-PROTECTED { SEQUENCE {
        bindingType             [0]    OPERATIONAL-BINDING.&id ({OpBindingSet}),
        bindingID               [1]    OperationalBindingID,
        -- symmetric, Role A initiates, or Role B initiates --
        initiator CHOICE {
                symmetric               [2]    OPERATIONAL-BINDING.&both.&TerminateParam
                                               ({OpBindingSet}{@bindingType}),
                roleA-initiates         [3]    OPERATIONAL-BINDING.&roleA.&TerminateParam
                                               ({OpBindingSet}{@bindingType}),
                roleB-initiates         [4]    OPERATIONAL-BINDING.&roleB.&TerminateParam
                                               ({OpBindingSet}{@bindingType})}  OPTIONAL,
        terminateAt             [5]    UTCTime OPTIONAL,
        securityParameters      [6]    SecurityParameters OPTIONAL },
        DIRQOP.&dopTermOpBindArg-QOP{@dirqop} }

TerminateOperationalBindingResult  ::=  CHOICE {
        null            [0]    NULL,
        protected       [1]    PROTECTED { SEQUENCE {
                bindingID               OperationalBindingID,
                bindingType             OPERATIONAL-BINDING.&id ({OpBindingSet}),
                terminateAt             GeneralizedTime OPTIONAL,
                COMPONENTS OF           CommonResults },
        DIRQOP.&dopTermOpBindRes-QOP{@dirqop} } }

-- errors and parameters --

operationalBindingError  ERROR  ::=  {
        PARAMETER       OPTIONALLY-PROTECTED {
                            OpBindingErrorParam,
                            DIRQOP.&dirErrors-QOP{@dirqop} }
        CODE            id-err-operationalBindingError }

OpBindingErrorParam  ::=  SEQUENCE {
        problem                 [0]    ENUMERATED {
                invalidID                       (0),
                duplicateID                     (1),
                unsupportedBindingType          (2),
                notAllowedForRole               (3),
                parametersMissing               (4),
                roleAssignment                  (5),
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

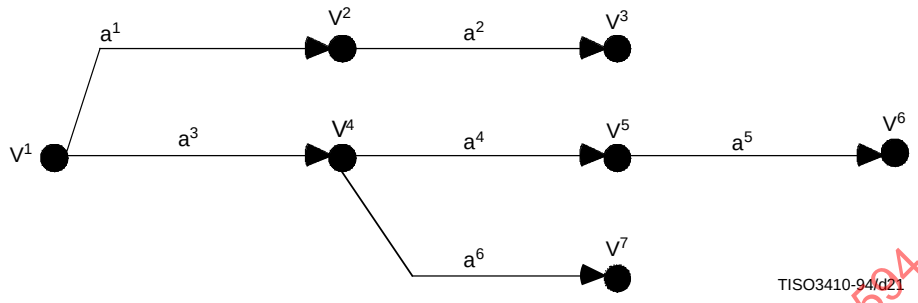
```
                invalidStartTime            (6),
                invalidEndTime              (7),
                invalidAgreement            (8),
                currentlyNotDecidable       (9),
                modificationNotAllowed      (10) },
        bindingType             [1]   OPERATIONAL-BINDING.&id ({OpBindingSet}) OPTIONAL,
        agreementProposal       [2]   OPERATIONAL-BINDING.&Agreement
                                      ({OpBindingSet}{@bindingType}) OPTIONAL,
        retryAt                 [3]   UTCTime OPTIONAL,
        COMPONENTS OF                 CommonResults }


-- information object classes --

OPERATIONAL-BINDING  ::=  CLASS {
        &Agreement,
        &Cooperation      OP-BINDING-COOP,
        &both             OP-BIND-ROLE OPTIONAL,
        &roleA            OP-BIND-ROLE OPTIONAL,
        &roleB            OP-BIND-ROLE OPTIONAL,
        &id               OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
        AGREEMENT                 &Agreement
        APPLICATION CONTEXTS      &Cooperation
        [ SYMMETRIC               &both ]
        [ ASYMMETRIC
                [ ROLE-A          &roleA ]
                [ ROLE-B          &roleB ] ]
        ID                        &id }

OP-BINDING-COOP  ::=  CLASS {
        &applContext      APPLICATION-CONTEXT,
        &Operations       OPERATION OPTIONAL }
WITH SYNTAX {
        &applContext
        [ APPLIES TO      &Operations ] }

OP-BIND-ROLE  ::=  CLASS {
        &establish                    BOOLEAN  DEFAULT FALSE,
        &EstablishParam                        OPTIONAL,
        &modify                       BOOLEAN  DEFAULT FALSE,
        &ModifyParam                           OPTIONAL,
        &terminate                    BOOLEAN  DEFAULT FALSE,
        &TerminateParam                        OPTIONAL }
WITH SYNTAX {
        [ ESTABLISHMENT-INITIATOR       &establish ]
        [ ESTABLISHMENT-PARAMETER       &EstablishParam ]
        [ MODIFICATION-INITIATOR        &modify ]
        [ MODIFICATION-PARAMETER        &ModifyParam ]
        [ TERMINATION-INITIATOR         &terminate ]
        [ TERMINATION-PARAMETER         &TerminateParam ] }

OpBindingSet     OPERATIONAL-BINDING  ::=  {
        shadowOperationalBinding  |
        hierarchicalOperationalBinding  |
        nonSpecificHierarchicalOperationalBinding }

END
```

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Annex G

The Mathematics of Trees

(This annex does not form an integral part of this Recommendation | International Standard)



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 $(a^1, a^2, ..., a^n)$ $(n \geq 1)$ 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.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Annex H

Name Design Criteria

(This annex does not form an integral part of this Recommendation | International Standard)

The information framework is very general, and allows for arbitrary variety of entries and attributes within the DIT. Since, as defined there, names are closely related to paths through the DIT, this means that arbitrary variety in names is possible. This annex suggests criteria to be considered in the design of names. The appropriate criteria have been used in the design of the recommended name forms which are to be found in ITU-T Rec. X.521 | ISO/IEC 9594-7. It is suggested that the criteria also be used, where appropriate, in designing the names for objects to which the recommended name forms do not apply.

Presently, only one criterion is addressed; that of user-friendliness.

NOTE – Not all names need to be user-friendly.

The remainder of this annex discusses the concept of user friendliness applied to names.

Names with which human beings must deal directly should be user-friendly. A user-friendly name is one that takes the human user's point of view, not the computer's. It is one that is easy for people to deduce, remember and understand, rather than one that is easy for computers to interpret.

The goal of user-friendliness can be stated somewhat more precisely in terms of the following two principles:

– A human being usually should be able to correctly guess an object's user-friendly name on the basis of information about the object that he naturally possesses. For example, one should be able to guess a business person's name given only the information about her casually acquired through normal business association.

– When a object's name is ambiguously specified, the Directory should recognize that fact rather than conclude that the name identifies one particular object. For example, where two people have the same last name, the last name alone should be considered as inadequate identification of either party.

The following subgoals follow from the goal of user-friendliness:

a) Names should not artificially remove natural ambiguities. For example, if two people share the last name "Jones", neither should be required to answer to "WJones" or "Jones2". Instead, the naming convention should provide a user-friendly means of discriminating between the entities. For example, it might require first name and middle initial in addition to last name.

b) Names should admit common abbreviations and common variations in spelling. For example, if one is employed by the Conway Steel Corporation and the name of one's employer figures in one's name, any of the names "Conway Steel Corporation", "Conway Steel Corp.", "Conway Steel", and "CSC" should suffice to identify the organization in question.

c) In certain cases, alias names can be used: to direct the search for a particular entry, in order to be more user-friendly, or to reduce the scope of a search. The following example demonstrates the use of an alias name for such a purpose: As shown in Figure H.1, the branch office in Osaka can also be identified with the name { C = Japan, L = Osaka, O = ABC, OU = Osaka-branch }.

d) If names are multi-part, both the number of mandatory parts and the number of optional parts should be relatively small and thus easy to remember.

e) If names are multi-part, the precise order in which those parts appear should generally be immaterial.

f) User-friendly names should not involve computer addresses.

g) In certain cases, contexts can be used to provide alternative names. For example, as shown in Figure H.2, the person Jones can be identified by {O = "XYZ", OU = "Research", CN = "Jones"} when the context is Language = English, and {O = "XYZ", OU = "Recherche", CN = "Jones"} when the context is Language = French.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

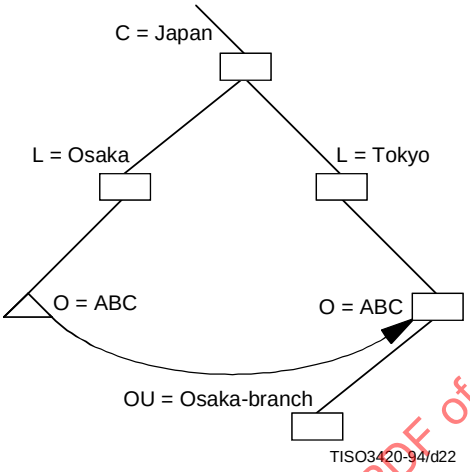


Figure H.1 – Aliasing example

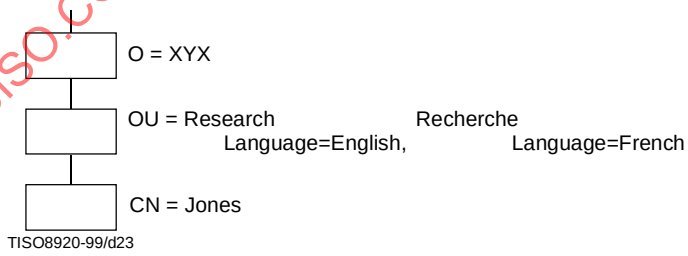


Figure H.2 – Example of context variations of a name

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Annex I

Examples of various aspects of schema

(This annex does not form an integral part of this Recommendation | International Standard)

I.1 Example of an Attribute Hierarchy

Figure I.1 shows a simple hierarchy of values of a generic **telephoneNumber** attribute, values of which are represented as contained in the outer set. Two specific attribute types are derived from the generic type, **workTelephoneNumber** and **homeTelephoneNumber**. Values of these types are represented as contained in the inner sets.

A value of type **homeTelephoneNumber** is contained in both the inner set representing **homeTelephoneNumber** and the outer set representing **telephoneNumber**, but not the inner set representing **workTelephoneNumber** values.

A DIT structure rule could be defined which permits entries to contain values of all three types shown in Figure I.1. Another rule could be defined permitting entries to contain only values of type telephoneNumber.

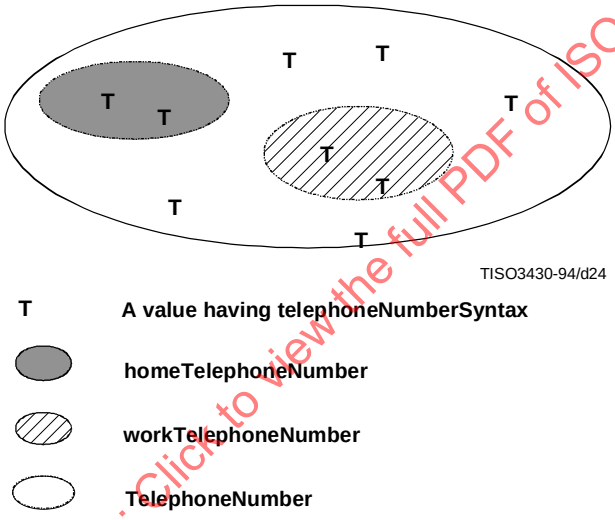


Figure I.1 – Hierarchy of Telephone Number Attribute Values

I.2 Example of a Subtree Specification

The following is an example illustrating the specification of subtrees. Consider the portion of the DIT represented in Figure I.2.

Subtree 1 and subtree 2 are specified with respect to the administrative point having name a. The identifiers b1, c2, d3, etc. represent local name values with respect to the administrative point a.

Subtree 1 may be specified as:

```
subtree1 SubtreeSpecification ::= {
                    specificExclusions { chopBefore b1 } }
```

Subtree 2 may be specified as:

```
subtree2 SubtreeSpecification      ::= {
                    base b1 }
```

Suppose that the entries identified in the figure with local names e1, e2, etc., represent organizational person entries. A subtree refinement could be specified to include all of these entries in the administrative area as:

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

```
subtree-refinement1 SubtreeSpecification ::= {
        specificationFilter
                item            id-oc-organizationalPerson }
```

This could be further refined to include only the organizational persons in subtree 2 as:

```
subtree2-refinement SubtreeSpecification::=     {
        base                        b1,
        specificationFilter
                item            id-oc-organizationalPerson }
```

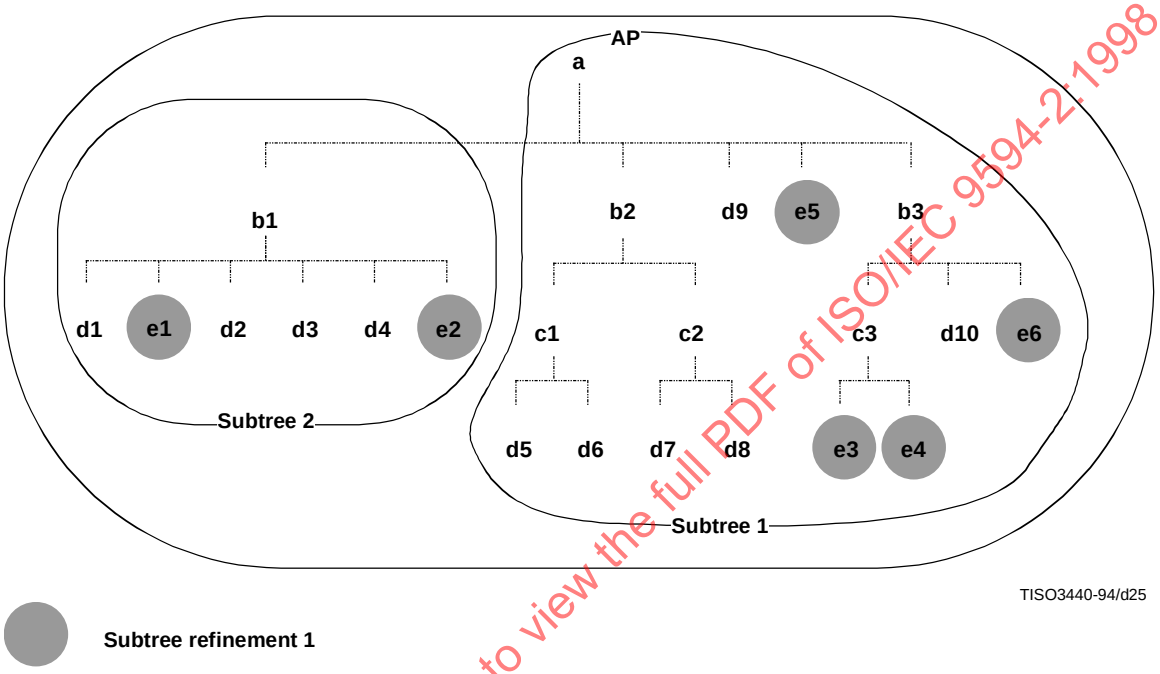


Figure I.2 – Subtree Specification Example

I.3 Schema Specification

I.3.1 Object Classes and Name Forms

The following object classes, defined in ITU-T Rec. X.521 | ISO/IEC 9594-7, are used within a particular subschema administrative area:

– **organization**;

– **organizationalUnit**;

– **organizationalPerson**.

A name form is not required for the administrative entry, which will be the only entry in the subschema of object class **organization**. The following name forms, defined in ITU-T Rec. X.521 | ISO/IEC 9594-7, are used to include entries of class organizationalUnit and organizationalPerson:

– **orgNameForm**;

– **orgUnitNameForm**;

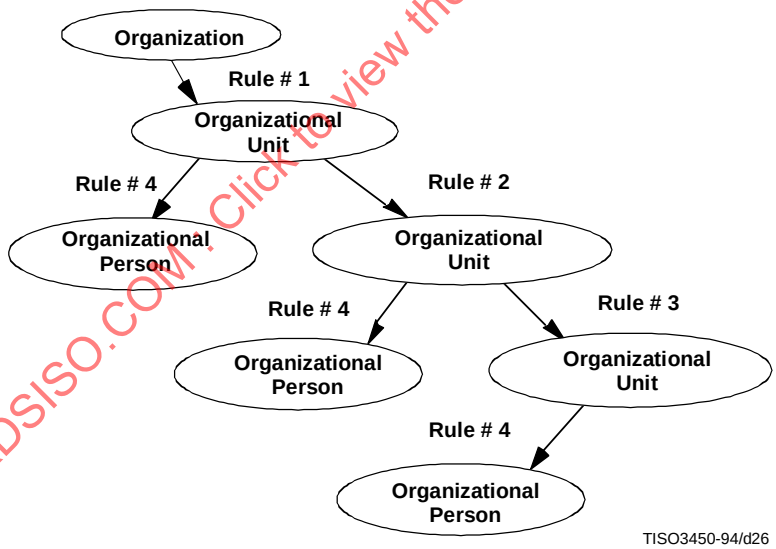
– **orgPersonNameForm**.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

I.3.2 DIT Structure Rules

The following structure rules are defined to specify a tree structure as shown in Figure I.3. Figure I.3 illustrates which rule may be used to add entries at the various points in the DIT.

```
rule-0 STRUCTURE-RULE::= {
        NAME FORM                   orgNameForm
        ID                          0 }

rule-1 STRUCTURE-RULE::= {
        NAME FORM                   orgUnitNameForm
        SUPERIOR RULES              { rule-0 }
        ID                          1 }

rule-2 STRUCTURE-RULE::= {
        NAME FORM                   orgUniNameForm
        SUPERIOR RULES              { rule-1 }
        ID                          2 }

rule-3 STRUCTURE-RULE::= {
        NAME FORM                   orgUniNameForm
        SUPERIOR RULES              { rule-2 }
        ID                          3 }

rule-4 STRUCTURE-RULE::= {
        NAME FORM                   orgPersonNameForm
        SUPERIOR RULES              { rule-1, rule-2, rule-3 }
        ID                          4 }
```



STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 9594-2:1998

Figure I.3 – Example Subschema

I.4 DIT content rules

The subschema administrator has the following two requirements to add supplemental information to entries in the subschema administrative area:

– all **organizationalPerson** and **organizationalUnit** entries should have the **organizationalTelephoneNumber** attribute. This attribute should be returned when the Directory is queried for telephoneNumbers;

– all **organizationalPerson** entries will have the new attribute manager.