

INTERNATIONAL STANDARD

ISO/IEC
9594-2

Second edition
1995-09-15

Information technology — Open Systems Interconnection — The Directory: Models

*Technologies de l'information — Interconnexion de systèmes ouverts
(OSI) — L'Annuaire: Modèles*



Reference number
ISO/IEC 9594-2:1995(E)
AVA Documents
Document Sharing Hub

Contents

	Page
SECTION 1 – GENERAL.....	1
1 Scope.....	1
2 Normative references	2
2.1 Identical Recommendations International Standards	2
2.2 Paired Recommendations International Standards equivalent in technical content	2
3 Definitions	3
3.1 OSI Reference Model Definitions.....	3
3.2 Basic directory definitions	3
3.3 Distributed operation definitions	3
3.4 Replication definitions	3
4 Abbreviations	4
5 Conventions.....	4
SECTION 2 – OVERVIEW OF THE DIRECTORY MODELS	5
6 Directory Models	5
6.1 Definitions.....	5
6.2 The Directory and its Users	5
6.3 Directory and DSA Information Models	6
6.4 Directory Administrative Authority Model	7
SECTION 3 – MODEL OF DIRECTORY USER INFORMATION	8
7 Directory Information Base.....	8
7.1 Definitions.....	8
7.2 Objects	8
7.3 Directory Entries.....	9
7.4 The Directory Information Tree (DIT)	9

8	Directory Entries	10
8.1	Definitions.....	10
8.2	Overall Structure	11
8.3	Object Classes	11
8.4	Attribute Types	13
8.5	Attribute Values	13
8.6	Attribute Type Hierarchies.....	14
8.7	Matching Rules	14
8.8	Entry Collections	16
9	Names.....	17
9.1	Definitions.....	17
9.2	Names in General.....	17
9.3	Relative Distinguished Names	18
9.4	Distinguished Names	18
9.5	Alias Names	19
SECTION 4 – DIRECTORY ADMINISTRATIVE MODEL		19
10	Directory Administrative Authority model	19
10.1	Definitions.....	19
10.2	Overview	20
10.3	Policy	20
10.4	Specific administrative authorities	21
10.5	Administrative areas and administrative points	21
10.6	DIT Domain policies.....	24
10.7	DMD policies.....	24
SECTION 5 – MODEL OF DIRECTORY ADMINISTRATIVE AND OPERATIONAL INFORMATION		25
11	Model of Directory Administrative and Operational Information	25
11.1	Definitions.....	25
11.2	Overview	25
11.3	Subtrees.....	26
11.4	Operational attributes	28
11.5	Entries	29
11.6	Subentries.....	29
11.7	Information model for collective attributes.....	31
SECTION 6 – THE DIRECTORY SCHEMA		31
12	Directory Schema.....	31
12.1	Definitions.....	31
12.2	Overview	32
12.3	Object class definition.....	33
12.4	Attribute type definition	35
12.5	Matching rule definition.....	38
12.6	DIT structure definition	39
12.7	DIT content rule definition	41

13	Directory System Schema	42
13.1	Overview	42
13.2	System schema supporting the administrative and operational information model.....	43
13.3	System schema supporting the administrative model	44
13.4	System schema supporting general administrative and operational requirements	44
13.5	System schema supporting access control	45
13.6	System schema supporting the collective attribute model	45
13.7	Maintenance of system schema.....	45
14	Directory schema administration.....	46
14.1	Overview	46
14.2	Policy objects	46
14.3	Policy parameters.....	46
14.4	Policy procedures.....	47
14.5	Subschema modification procedures	47
14.6	Entry addition and modification procedures	47
14.7	Subschema policy attributes.....	48
SECTION 7 – SECURITY		51
15	Security model	51
15.1	Definitions.....	51
15.2	Security policies	52
16	Basic Access Control	53
16.1	Scope and application	53
16.2	Basic Access Control model	53
16.3	Access control administrative areas	56
16.4	Representation of Access Control Information.....	58
16.5	The ACI operational attributes.....	62
16.6	Protecting the ACI	63
16.7	Access control and Directory operations	63
16.8	Access Control Decision Function.....	63
16.9	Simplified Access Control	64
SECTION 8 – DSA MODELS		65
17	DSA Models.....	65
17.1	Definitions.....	65
17.2	Directory Functional Model.....	65
17.3	Directory Distribution Model.....	66
SECTION 9 – DSA INFORMATION MODEL		68
18	Knowledge	68
18.1	Definitions.....	68
18.2	Introduction.....	68
18.3	Knowledge References	69
18.4	Minimum Knowledge	71
18.5	First Level DSAs.....	72

19	Basic Elements of the DSA Information Model.....	72
19.1	Definitions.....	72
19.2	Introduction.....	72
19.3	DSA-Specific Entries and their Names.....	73
19.4	Basic Elements.....	74
20	Representation of DSA Information	75
20.1	Representation of Directory User and Operational Information.....	76
20.2	Representation of Knowledge References	76
20.3	Representation of Names and Naming Contexts	82
SECTION 10 – DSA OPERATIONAL FRAMEWORK		83
21	Overview	83
21.1	Definitions.....	83
21.2	Introduction.....	84
22	Operational bindings	84
22.1	General	84
22.2	Application of the operational framework	85
22.3	States of cooperation.....	86
23	Operational binding specification and management	87
23.1	Operational binding type specification	87
23.2	Operational binding management	88
23.3	Operational binding specification templates.....	89
24	Operations for operational binding management	91
24.1	Application-context definition	91
24.2	Establish Operational Binding operation	91
24.3	Modify Operational Binding operation.....	93
24.4	Terminate Operational Binding operation	94
24.5	Operational Binding Error	95
24.6	Operational Binding Management Bind and Unbind.....	96
Annex A – Object identifier usage		97
Annex B – Information Framework in ASN.1		99
Annex C – SubSchema Administration Schema in ASN.1		104
Annex D – Basic Access Control in ASN.1		107
Annex E – DSA Operational Attribute Types in ASN.1		110
Annex F – Operational Binding Management in ASN.1.....		113
Annex G – The Mathematics of Trees.....		117
Annex H – Name Design Criteria.....		118
Annex I – Examples of various aspects of schema		120
Annex J – Overview of Basic Access Control Permissions		124
Annex K – Example of Basic Access Control.....		127
Annex L – DSE Type Combinations.....		144
Annex M – Modelling of knowledge		146
Annex N – Alphabetical index of definitions.....		151
Annex O – Amendments and corrigenda		153

Foreword

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

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

International Standard ISO/IEC 9594-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 21, *Open systems interconnection, data management and open distributed processing*, in collaboration with ITU-T. The identical text is published as ITU-T Recommendation X.501.

Implementors should note that a defect resolution process exists and that corrections may be applied to this part of ISO/IEC 9594 in the form of technical corrigenda. A list of approved technical corrigenda for this part of ISO/IEC 9594 can be obtained from the subcommittee secretariat. Published technical corrigenda are available from your national standards organization.

This second edition technically revises and enhances ISO/IEC 9594-2:1990. It also incorporates technical corrigendum 1:1991 and technical corrigendum 2:1992. Implementations may still claim conformance to the first edition of this part of ISO/IEC 9594. However, at some point, the first edition will no longer be supported (i.e. reported defects will no longer be resolved). It is recommended that implementations conform to this second edition as soon as possible.

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*

Annexes A to F form an integral part of this part of ISO/IEC 9594. Annexes G to O are for information only.

Introduction

This Recommendation | International Standard, together with the other Recommendation | 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 second edition technically revises and enhances, but does not replace, the first edition of this Recommendation | International Standard. Implementations may still claim conformance to the first edition.

This second edition specifies version 1 of the Directory service and protocols. The first edition also specifies version 1. Differences between the services and between the protocols defined in the two editions are accommodated using the rules of extensibility defined in this edition of X.519 | ISO/IEC 9594-5.

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 | 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.

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, lists the amendments and defect reports that have been incorporated to form this edition of this Recommendation | International Standard.

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 two flexible, specific access control schemes which are suitable for a wide variety of applications and styles of use. The security model is concerned solely with control of access to the Directory information, not control of access to the DSA application-entity holding the information.

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.

2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard part. 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 editions 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.500 (1993) | ISO/IEC 9594-1:1995, *Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services.*
- ITU-T Recommendation X.509 (1993) | ISO/IEC 9594-8:1995, *Information technology – Open Systems Interconnection – The Directory: Authentication framework.*
- ITU-T Recommendation X.511 (1993) | ISO/IEC 9594-3:1995, *Information technology – Open Systems Interconnection – The Directory: Abstract service definition.*
- ITU-T Recommendation X.518 (1993) | ISO/IEC 9594-4:1995, *Information technology – Open Systems Interconnection – The Directory: Procedures for distributed operation.*
- ITU-T Recommendation X.519 (1993) | ISO/IEC 9594-5:1995, *Information technology – Open Systems Interconnection – The Directory: Protocol specifications.*
- ITU-T Recommendation X.520 (1993) | ISO/IEC 9594-6:1995, *Information technology – Open Systems Interconnection – The Directory: Selected attribute types.*
- ITU-T Recommendation X.521 (1993) | ISO/IEC 9594-7:1995, *Information technology – Open Systems Interconnection – The Directory: Selected object classes.*
- ITU-T Recommendation X.525 (1993) | ISO/IEC 9594-9:1995, *Information technology – Open Systems Interconnection – The Directory: Replication.*
- ITU-T Recommendation X.680 (1994) | ISO/IEC 8824-1:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*
- ITU-T Recommendation X.681 (1994) | ISO/IEC 8824-2:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.*
- ITU-T Recommendation X.682 (1994) | ISO/IEC 8824-3:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*
- ITU-T Recommendation X.683 (1994) | ISO/IEC 8824-4:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications.*
- ITU-T Recommendation X.811¹⁾ | ISO/IEC 10181-1:...¹⁾, *Information technology – Open Systems Interconnection – Security Frameworks in Open Systems: Authentication framework.*
- ITU-T Recommendation X.812¹⁾ | ISO/IEC 10181-2:...¹⁾, *Information technology – Open Systems Interconnection – Security Frameworks in Open Systems – Access control framework.*
- ITU-T Recommendation X.813¹⁾ | ISO/IEC 10181-3:...¹⁾, *Information technology – Open Systems Interconnection – Security Frameworks in Open Systems – Non-repudiation framework.*

2.2 Paired Recommendations | International Standards equivalent in technical content

- CCITT Recommendation X.200 (1988), *Reference Model of Open Systems Interconnection for CCITT Applications.*
- ISO 7498:1984, *Information processing systems – Open Systems Interconnection – Basic Reference Model.*
- 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.*

¹⁾ Presently at the stage of draft.

3 Definitions

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

3.1 OSI Reference Model Definitions

The following terms are defined in CCITT Rec. X.200 | ISO 7498:

- 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 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*;
- 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

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 1
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 Shadow 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 1993.

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 and all parts of ISO/IEC 9594.

This Directory Specification uses the term “1988 edition systems” to refer to systems conforming to the previous (1988) edition of the Directory Specifications, i.e. the 1988 edition of the series of ITU-T X.500 Recommendations and the ISO/IEC 9594:1990 edition. Systems conforming to the current Directory Specifications are referred to as “1993 edition systems”.

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 1 – 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 2 – 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.

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 17.2.

NOTES

1 Some open systems may provide a centralized DUA function retrieving information for the actual users (application-processes, persons, etc.). This is transparent to the Directory.

2 The DUA functions and a DSA can be within the same open system, and it is an implementation choice whether to make one or more DUAs visible within the OSI Environment as application-entities.

3 A DUA may exhibit local behavior and structure which is outside the scope of 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.

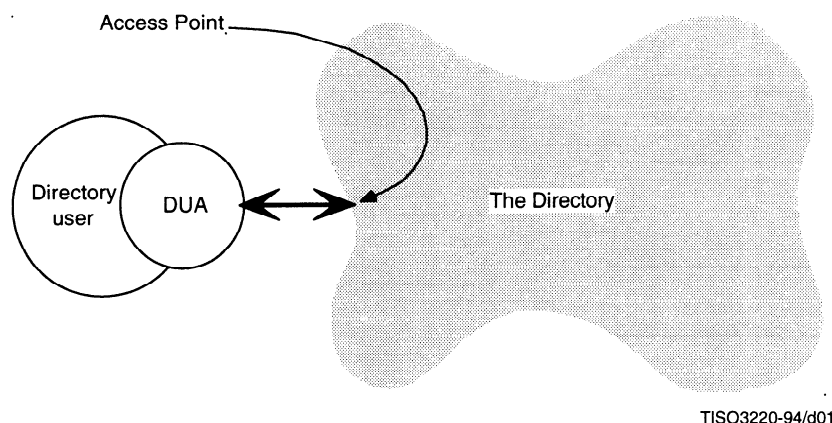


Figure 1 – Access to the Directory

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 18 through 20.

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 – 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.

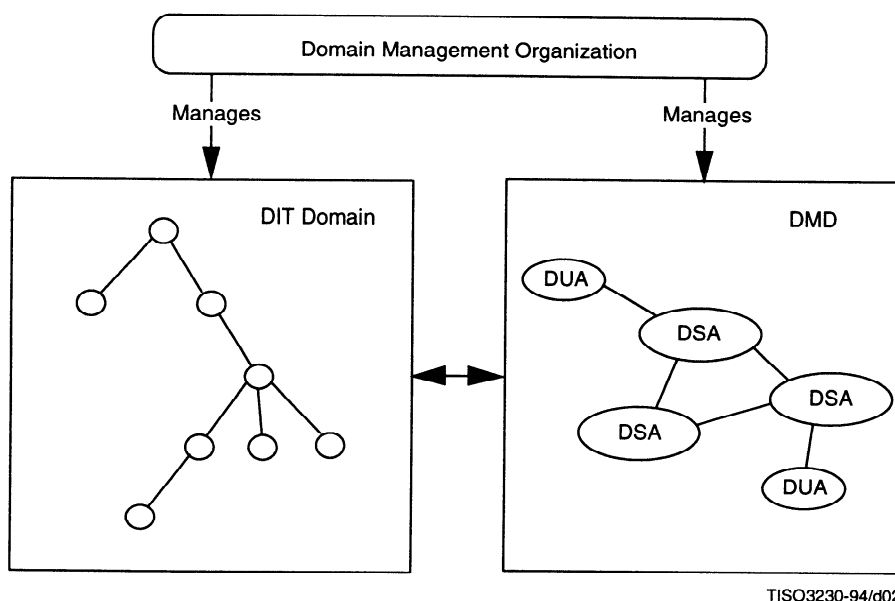


Figure 2 – Directory Management

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 – 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 super classes 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).

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).

NOTES

1 The ‘world’ is generally that of telecommunications and information processing or some part thereof.

2 The objects known to the Directory may not correspond exactly with the set of ‘real’ things in the world. For example, a real-world person may be regarded as two different objects, a business person and a residential person, as far as the Directory is concerned. The mapping is not defined in 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.

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

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

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

8.1.14 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.

8.1.15 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.

8.1.16 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.

8.1.17 operational attribute: An attribute representing operational and/or administrative information.

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

8.1.19 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*.

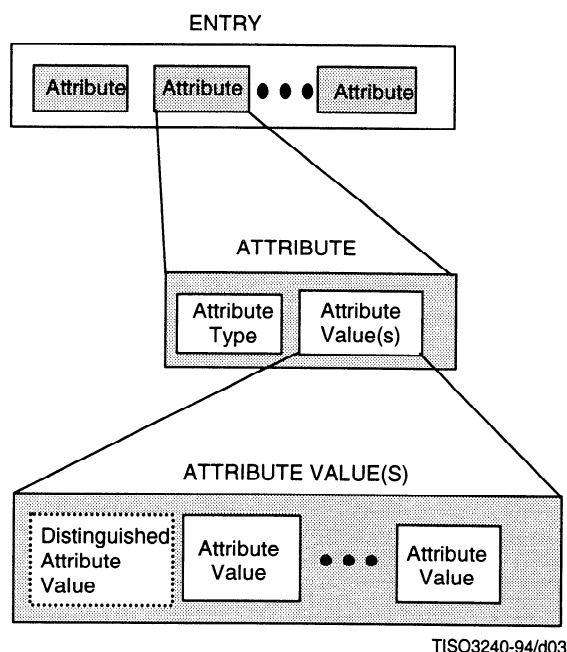


Figure 3 – Structure of an entry

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.

```
Attribute ::= SEQUENCE {
    type      ATTRIBUTE.&id ({ SupportedAttributes }),
    values     SET SIZE (1 .. MAX) OF ATTRIBUTE.&TYPE ({ SupportedAttributes}{@type}})
```

NOTE 2 – Attribute types and attribute values are described in 8.4. and 8.5 respectively.

An attribute may be designated as single valued or multi-valued. The Directory shall ensure that single valued attributes have only one value.

8.3 Object Classes

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

- describing and categorising 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.

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 CCITT 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.

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

At most one of the values of an attribute may be designated as a *distinguished value*, in which case the attribute value appears in the relative distinguished name (see 9.3) of the entry.

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.

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.

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 Matching Rules

8.7.1 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.

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.

8.7.2 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 and an attribute value:

AttributeValueAssertion := SEQUENCE {

type	ATTRIBUTE.&id ({SupportedAttributes}),
assertion	ATTRIBUTE.&equality-match.&AssertionType ({SupportedAttributes}@type)}

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.

An AVA is:

- a) undefined, if any of the following hold:
 - 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, one of whose values matches that value;
- c) false, otherwise.

8.7.3 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.

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.

8.7.4 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.

8.7.5 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.

8.8 Entry Collections

8.8.1 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.

8.8.2 Collective Attributes

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

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 the 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:

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

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

distinguished name (of an entry): The name of an entry which is formed from the sequence of the RDNs of the entry and each of its superior entries. Every object entry, alias entry and subentry has precisely one distinguished name.

(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).

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

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

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

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

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

RDNSequence ::= SEQUENCE OF
RelativeDistinguishedName

DistinguishedName ::= RDNSequence

NOTE – 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 a *relative distinguished name (RDN)*. An RDN of an object or alias entry consists of a set of attribute type and value pairs, each of which matches, using the equality matching rule, a distinct distinguished attribute value of the entry.

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.

RelativeDistinguishedName	::=	SET SIZE (1..MAX) OF AttributeTypeAndValue
AttributeTypeAndValue ::=		SEQUENCE
type		ATTRIBUTE.&id ({SupportedAttributes}),
value		{{ATTRIBUTE.&Type ({SupportedAttributes}){@type}}}

The set contains exactly one attribute type and value pair for each distinguished value in the entry; no two attribute type and value pairs can thus refer to the same attribute.

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

NOTE 2 – Frequently, an entry will contain a single distinguished value (and the RDN will thus comprise a single type and value pair); however, under certain circumstances (in order to differentiate), additional values (and hence attribute type and value pairs) may be used.

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

NOTE 3 – 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 4 – Changing the RDN of a non-leaf entry automatically changes the corresponding RDN of subordinate entries.

9.4 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.

NOTES

- 1 It is preferable that the distinguished names of objects which humans have to deal with be user-friendly.
- 2 ISO 7498-3 defines the concept of a primitive name. A distinguished name can be used as a primitive name for the object it identifies because:

a) it is unambiguous;

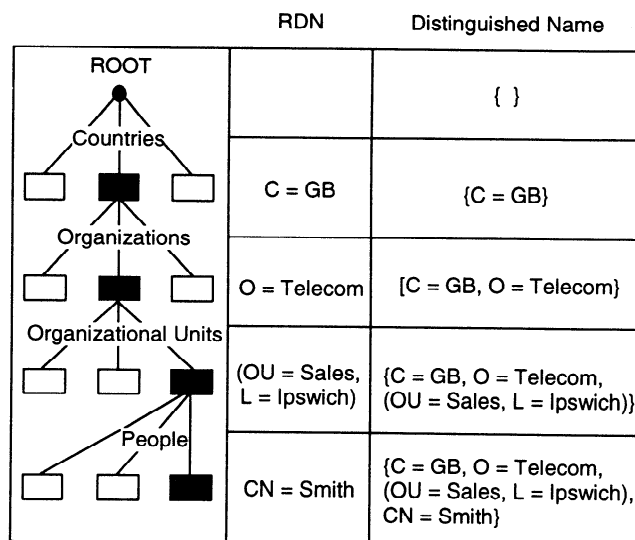
b) it is unique; and

c) the semantics of its internal structure (a sequence of RDNs) need not (but of course may) be understood by the user of the Directory.
- 3 Because only the object entry and its superiors are involved, distinguished names of objects can never involve alias entries.

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 the null sequence.

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



TISO3250-94/d04

Figure 4 – Determination of Distinguished Names

9.5 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.

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 – 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.

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.

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 characterised 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 modeling 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 18 through 20 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.

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 characterised 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 three kinds of specific administrative area:

- subschema administrative areas;

- access control administrative areas;
- collective-attribute administrative areas.

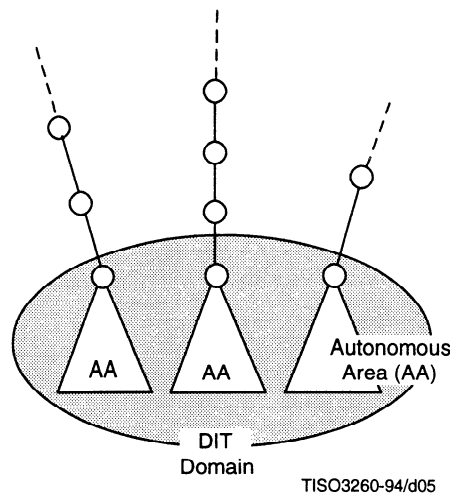


Figure 5 – A DIT Domain

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.

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 18 through 20 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.

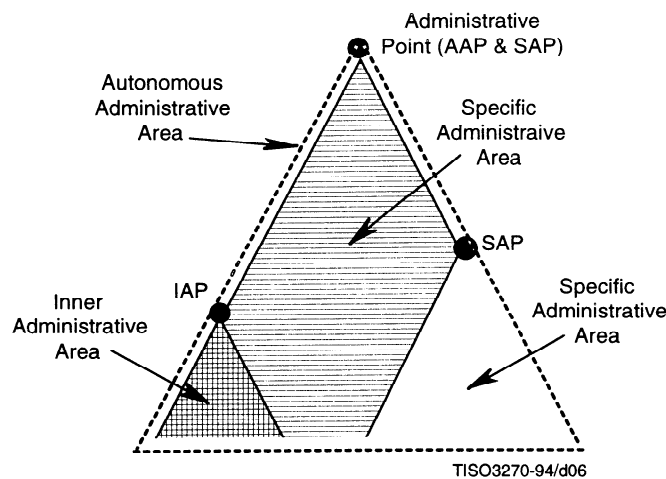


Figure 6 – Administrative Points and Areas

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:

- 1) Limiting the basic service provided to Directory (i.e. non-administrative) users to interrogation operations only, in accordance with CCITT Recommendation F.500.
- 2) Limiting the service provided to users accessing the DSA indirectly, via chaining, including distinctions based on whether the user request traversed a trusted path.
- 3) 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.

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 have been added in this edition of the Directory Specifications 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.

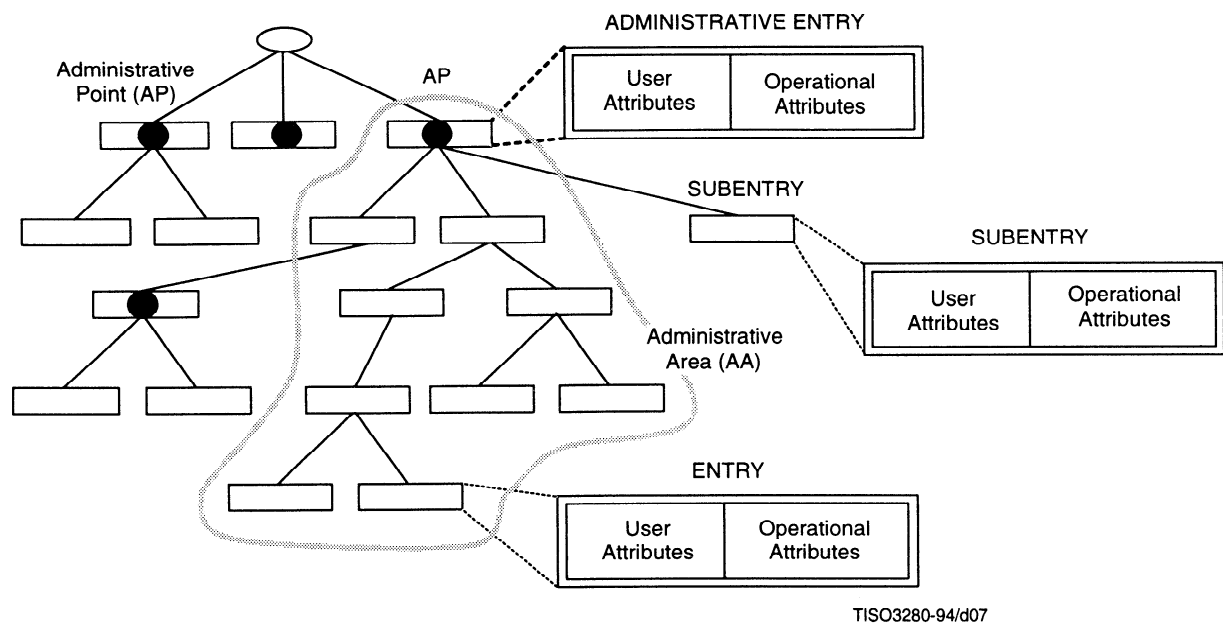


Figure 7 – Model of Directory Administrative and Operational Information

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.

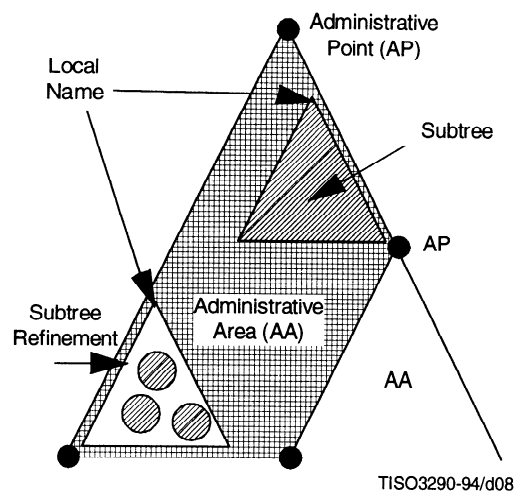


Figure 8 – Specification of Subtrees and Subtree Refinements within the context of Administrative Areas

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).

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 ::= RDNSSequence
```

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

11.3.4 ChopSpecification

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.

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 .

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.

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.521 | 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.

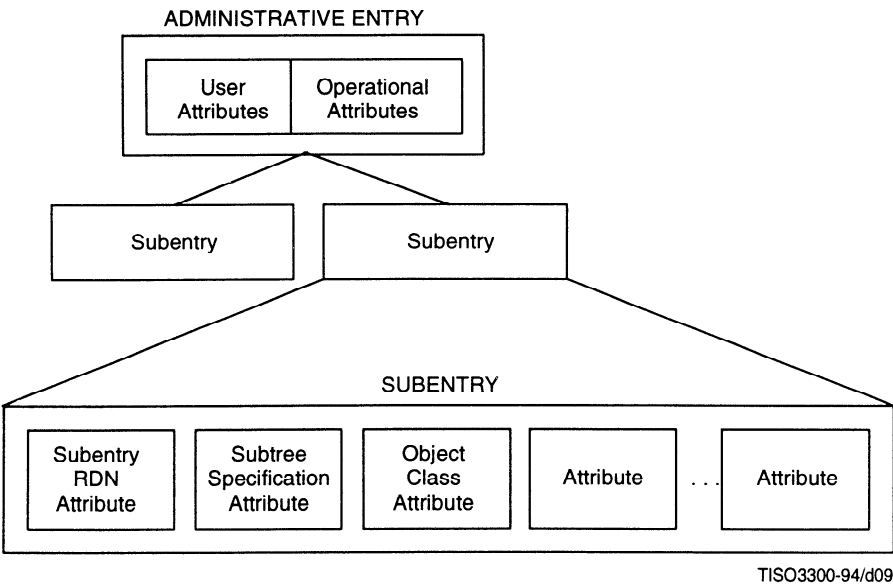


Figure 9 – Structure of a Subentry

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.

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.

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, 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.

12.1.5 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.

12.1.6 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.

12.1.7 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.

12.1.8 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 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.

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.

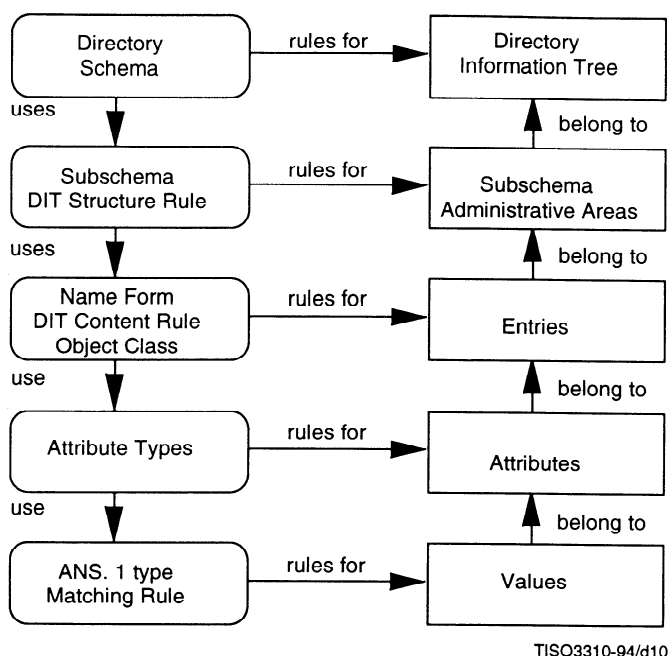


Figure 10 – Overview of Directory Schema

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.

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 and content rules. 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:

- &Superclasses** is the set of object classes which are its direct superclasses;
- &kind** is its kind;
- &MandatoryAttributes** is the set of attributes which entries of that class must contain;
- &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;
- &id** is the object identifier assigned to it.

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:

- optionally indicating that the attribute type is a subtype of a previously defined attribute type, its direct supertype;
- specifying the attribute syntax for the attribute type;
- optionally indicating the equality, ordering and/or substring matching rule(s) for the attribute type;
- indicating whether an attribute of this type shall have only one or may have more than one value;
- indicating whether the attribute type is operational or user;
- optionally indicating that a user attribute type is collective;
- optionally indicating that an operational attribute is not user modifiable;
- for operational attributes, indicating the application;
- 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.

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 an 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 }

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;
- 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;
- 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 AVAs;
- c) corresponding AVAs (i.e. those in corresponding RDNs and with identical attribute types) have attribute values which match for equality (in such a match, the attribute values take the same roles - i.e. as presented or target value - as the distinguished name which contains them in the overall match).

distinguishedNameMatch is an equality matching rule.

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 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.

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.

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
```

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:

- the **ruleIdentifier** component identifies the DIT structure rule uniquely within a subschema;
- the **nameForm** component of the DIT structure rule specifies the name form for entries governed by the DIT structure rule;
- 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 an autonomous 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 UNIQUE }
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:

- an indication of the structural object class to which it applies;
- optionally, an indication of the auxiliary object classes allowed for entries governed by the rule;
- 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;
- 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;
- 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.

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 ] }
```

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.