
Information technology — Open Systems Interconnection — The Directory: Models

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

Reference number
ISO/IEC 9594-2:2001(E)



PDF disclaimer

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

Adobe is a trademark of Adobe Systems Incorporated.

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

© ISO/IEC 2001

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

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

Published by ISO in 2002

Printed in Switzerland

CONTENTS

	<i>Page</i>
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.....	3
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.....	6
6 Directory Models	6
6.1 Definitions	6
6.2 The Directory and its Users	6
6.3 Directory and DSA Information Models	7
6.4 Directory Administrative Authority Model.....	8
SECTION 3 – MODEL OF DIRECTORY USER INFORMATION.....	9
7 Directory Information Base	9
7.1 Definitions	9
7.2 Objects.....	10
7.3 Directory Entries	10
7.4 The Directory Information Tree (DIT).....	10
8 Directory Entries.....	11
8.1 Definitions	11
8.2 Overall Structure	12
8.3 Object Classes	13
8.4 Attribute Types.....	15
8.5 Attribute Values	15
8.6 Attribute Type Hierarchies	15
8.7 Contexts.....	16
8.8 Matching Rules.....	17
8.9 Entry Collections.....	20
8.10 Compound entries and families of entries	21
9 Names.....	22
9.1 Definitions	22
9.2 Names in General	22
9.3 Relative Distinguished Names.....	23
9.4 Name Matching	24
9.5 Names returned during operations.....	24
9.6 Names held as attribute values or used as parameters	25
9.7 Distinguished Names.....	25
9.8 Alias Names	25
10 Hierarchical groups.....	26
10.1 Definitions	26
10.2 Hierarchical relationship	27

	<i>Page</i>
SECTION 4 – DIRECTORY ADMINISTRATIVE MODEL	28
11 Directory Administrative Authority model.....	28
11.1 Definitions	28
11.2 Overview	28
11.3 Policy.....	29
11.4 Specific administrative authorities	29
11.5 Administrative areas and administrative points.....	30
11.6 DIT Domain policies	32
11.7 DMD policies	32
SECTION 5 – MODEL OF DIRECTORY ADMINISTRATIVE AND OPERATIONAL INFORMATION.	34
12 Model of Directory Administrative and Operational Information	34
12.1 Definitions	34
12.2 Overview	34
12.3 Subtrees	35
12.4 Operational attributes	37
12.5 Entries.....	38
12.6 Subentries	38
12.7 Information model for collective attributes	39
12.8 Information model for context defaults	40
SECTION 6 – THE DIRECTORY SCHEMA.....	41
13 Directory Schema	41
13.1 Definitions	41
13.2 Overview	41
13.3 Object class definition	43
13.4 Attribute type definition	45
13.5 Matching rule definition	47
13.6 Relaxations and tightenings.....	49
13.7 DIT structure definition	56
13.8 DIT content rule definition	58
13.9 Context type definition	59
13.10 DIT Context Use definition	60
14 Directory System Schema.....	61
14.1 Overview	61
14.2 System schema supporting the administrative and operational information model.....	61
14.3 System schema supporting the administrative model.....	62
14.4 System schema supporting general administrative and operational requirements.....	62
14.5 System schema supporting access control.....	65
14.6 System schema supporting the collective attribute model	65
14.7 System schema supporting context assertion defaults	65
14.8 System schema supporting the service administration model	66
14.9 System schema supporting hierarchical groups.....	66
14.10 Maintenance of system schema	67
14.11 System schema for first-level subordinates	67
15 Directory schema administration	67
15.1 Overview	67
15.2 Policy objects	67
15.3 Policy parameters	68
15.4 Policy procedures	68
15.5 Subschema modification procedures	68
15.6 Entry addition and modification procedures.....	69
15.7 Subschema policy attributes	69

SECTION 7 – DIRECTORY SERVICE ADMINISTRATION	75
16 Service Administration Model.....	75
16.1 Definitions	75
16.2 Service-type/user-class model	75
16.3 Service specific administrative areas.....	76
16.4 Introduction to search-rules.....	77
16.5 Subfilters	77
16.6 Filter requirements.....	78
16.7 Attribute information selection based on search-rules.....	78
16.8 Access control aspects of search-rules	79
16.9 Contexts aspects of search-rules.....	79
16.10 Search-rule specification	79
16.11 Matching restriction definition	87
16.12 Search-validation function.....	87
SECTION 8 – SECURITY	89
17 Security model.....	89
17.1 Definitions	89
17.2 Security policies	89
17.3 Protection of Directory operations.....	90
18 Basic Access Control.....	94
18.1 Scope and application.....	94
18.2 Basic Access Control model.....	94
18.3 Access control administrative areas.....	96
18.4 Representation of Access Control Information	99
18.5 The ACI operational attributes	104
18.6 Protecting the ACI.....	104
18.7 Access control and Directory operations	105
18.8 Access Control Decision Function	105
18.9 Simplified Access Control.....	106
19 Rule-based Access Control.....	107
19.1 Scope and application.....	107
19.2 Rule-based Access Control model.....	107
19.3 Access control administrative areas.....	108
19.4 Security Label.....	108
19.5 Clearance.....	109
19.6 Access Control and Directory operations	109
19.7 Access Control Decision Function	110
19.8 Use of Rule-based and Basic Access Control.....	110
20 Cryptographic Protection in Storage.....	110
20.1 Data Integrity in Storage	110
20.2 Confidentiality of stored data	112
SECTION 9 – DSA MODELS	115
21 DSA Models	115
21.1 Definitions	115
21.2 Directory Functional Model	115
21.3 Directory Distribution Model.....	116
SECTION 10 – DSA INFORMATION MODEL.....	118
22 Knowledge.....	118
22.1 Definitions	118
22.2 Introduction	118
22.3 Knowledge References.....	119
22.4 Minimum Knowledge.....	121
22.5 First Level DSAs.....	122

	<i>Page</i>
23 Basic Elements of the DSA Information Model	122
23.1 Definitions	122
23.2 Introduction	122
23.3 DSA-Specific Entries and their Names	123
23.4 Basic Elements	124
24 Representation of DSA Information	126
24.1 Representation of Directory User and Operational Information.....	126
24.2 Representation of Knowledge References.....	127
24.3 Representation of Names and Naming Contexts	133
SECTION 11 – DSA OPERATIONAL FRAMEWORK	135
25 Overview	135
25.1 Definitions	135
25.2 Introduction	135
26 Operational bindings.....	135
26.1 General	135
26.2 Application of the operational framework.....	136
26.3 States of cooperation	137
27 Operational binding specification and management.....	138
27.1 Operational binding type specification.....	138
27.2 Operational binding management.....	139
27.3 Operational binding specification templates	140
28 Operations for operational binding management.....	142
28.1 Application-context definition.....	142
28.2 Establish Operational Binding operation.....	142
28.3 Modify Operational Binding operation	144
28.4 Terminate Operational Binding operation.....	145
28.5 Operational Binding Error.....	146
28.6 Operational Binding Management Bind and Unbind	147
Annex A – Object identifier usage	149
Annex B – Information Framework in ASN.1	152
Annex C – SubSchema Administration Schema in ASN.1	161
Annex D – Service Administration in ASN.1	165
Annex E – Basic Access Control in ASN.1	169
Annex F – DSA Operational Attribute Types in ASN.1	172
Annex G – Operational Binding Management in ASN.1	175
Annex H – Enhanced security	179
Annex I – The Mathematics of Trees.....	185
Annex J – Name Design Criteria	186
Annex K – Examples of various aspects of schema.....	188
K.1 Example of an Attribute Hierarchy	188
K.2 Example of a Subtree Specification.....	188
K.3 Schema Specification	189
K.4 DIT content rules.....	190
K.5 DIT context use	191
Annex L – Overview of Basic Access Control Permissions	192
L.1 Introduction	192
L.2 Permissions required for operations	192
L.3 Permissions affecting error.....	193
L.4 Entry level permissions	194
L.5 Entry level permissions	195

Annex M – Examples of Access Control	196
M.1 Introduction	196
M.2 Design principles for Basic Access Control	196
M.3 Introduction to example	197
M.4 Policy affecting the definition of specific and inner areas	197
M.5 Policy affecting the definition of DACDs	200
M.6 Policy expressed in prescriptiveACI attributes	202
M.7 Policy expressed in subentryACI attributes	209
M.8 Policy expressed in entryACI attributes	210
M.9 ACDF examples	210
M.10 Rule-based Access Control	212
Annex N – DSE Type Combinations	213
Annex O – Modelling of knowledge	215
Annex P – Names held as attribute values or used as parameters	220
Annex Q – Subfilters	221
Annex R – Compound entry name patterns and their use	222
Annex S – Alphabetical index of definitions	224
Annex T – Amendments and corrigenda	226

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.

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

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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

Users and implementors should note the existence of a “defect resolution” procedure in ISO/IEC JTC 1 to identify and correct errors in International Standards through the publication of Technical Corrigenda. Identical corrections are made to the corresponding ITU-T Recommendations through Corrigenda and may also be made in the form of Implementors' Guides. Details of Technical Corrigenda to International Standards are available on the ISO website; published Technical Corrigenda can be obtained via the ISO webstore or from the ISO and IEC national bodies. Corrigenda and Implementors' Guides to ITU-T Recommendations can be obtained from the ITU-T website.

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

This fourth edition of ISO/IEC 9594-2 constitutes a technical revision of the third edition (ISO/IEC 9594-2:1998), which is provisionally retained in order to support implementations based on the third edition. This edition also incorporates Corrigendum 1:2002 and Corrigendum 2:2002.

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: Public-key and attribute certificate frameworks*
- *Part 9: Replication*
- *Part 10: Use of systems management for administration of the Directory*

Annexes A to H form a normative part of this part of ISO/IEC 9594. Annexes I to T are for information only.

Introduction

This Recommendation | International Standard, together with the other Recommendations | International Standards, has been produced to facilitate the interconnection of information processing systems to provide directory services. A set of such systems, together with the directory information that 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 fourth edition technically revises and enhances, but does not replace, the third edition of this Recommendation | International Standard. Implementations may still claim conformance to the third edition. However, at some point, the third edition will not be supported (i.e. reported defects will no longer be resolved). It is recommended that implementations conform to this fourth edition as soon as possible.

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

The first and second editions specified only version 1. Most of the services and protocols specified in this edition are designed to function under version 1. However some enhanced services and protocols, e.g. signed errors, will not function unless all Directory entities involved in the operation have negotiated version 2. Whichever version has been negotiated, differences between the services and between the protocols defined in the four editions, except for those specifically assigned to version 2, are accommodated using the rules of extensibility defined in this edition of ITU-T Rec. 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 | parts of ISO/IEC 9594.

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

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

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

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

Annex F, 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.

ISO/IEC 9594-2:2001(E)

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

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

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

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

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

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

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

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

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

Annex Q, which is not an integral part of this Recommendation | International Standard, describes the concept of subfilters.

Annex R, which is not an integral part of this Recommendation | International Standard, describes recommendations and examples on how family members can be named.

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

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

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

- a) the Directory functional model describes how the Directory is manifested as a set of one or more components, each being a DSA;
- b) the Directory distribution model describes the principals according to which the DIB entries and entry-copies may be distributed among DSAs;
- c) the DSA information model describes the structure of the Directory user and operational information held in a DSA;
- d) the DSA operational framework describes the means by which the definition of specific forms of cooperation between DSAs to achieve particular objectives (e.g. shadowing) is structured.

2 Normative references

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

2.1 Identical Recommendations | International Standards

- ITU-T Recommendation X.200 (1994) | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*.
- ITU-T Recommendation X.500 (2001) | ISO/IEC 9594-1:2001, *Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services*.
- ITU-T Recommendation X.509 (2000) | ISO/IEC 9594-8:2001, *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks*.
- ITU-T Recommendation X.511 (2001) | ISO/IEC 9594-3:2001, *Information technology – Open Systems Interconnection – The Directory: Abstract service definition*.
- ITU-T Recommendation X.518 (2001) | ISO/IEC 9594-4:2001, *Information technology – Open Systems Interconnection – The Directory: Procedures for distributed operation*.
- ITU-T Recommendation X.519 (2001) | ISO/IEC 9594-5:2001, *Information technology – Open Systems Interconnection – The Directory: Protocol specifications*.
- ITU-T Recommendation X.520 (2001) | ISO/IEC 9594-6:2001, *Information technology – Open Systems Interconnection – The Directory: Selected attribute types*.
- ITU-T Recommendation X.521 (2001) | ISO/IEC 9594-7:2001, *Information technology – Open Systems Interconnection – The Directory: Selected object classes*.
- ITU-T Recommendation X.525 (2001) | ISO/IEC 9594-9:2001, *Information technology – Open Systems Interconnection – The Directory: Replication*.
- ITU-T Recommendation X.530 (2001) | ISO/IEC 9594-10:2001, *Information technology – Open Systems Interconnection – The Directory: Use of systems management for administration of the Directory*.
- CCITT Recommendation X.660 (1992) | ISO/IEC 9834-1:1993, *Information technology – Open Systems Interconnection – Procedures for the operation of OSI Registration Authorities: General procedures*.
- ITU-T Recommendation X.680 (1997) | ISO/IEC 8824-1:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*.
- ITU-T Recommendation X.681 (1997) | ISO/IEC 8824-2:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification*.
- ITU-T Recommendation X.682 (1997) | ISO/IEC 8824-3:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification*.
- ITU-T Recommendation X.683 (1997) | ISO/IEC 8824-4:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications*.
- ITU-T Recommendation X.803 (1994) | ISO/IEC 10745:1995, *Information technology – Open Systems Interconnection – Upper layers security model*.
- ITU-T Recommendation X.811 (1995) | ISO/IEC 10181-2:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Authentication framework*.
- ITU-T Recommendation X.812 (1995) | ISO/IEC 10181-3:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems – Access control framework*.
- ITU-T Recommendation X.813 (1996) | ISO/IEC 10181-4:1997, *Information technology – Open Systems Interconnection – Security frameworks for open systems – Non-repudiation framework*.
- ITU-T Recommendation X.830 (1995) | ISO/IEC 11586-1:1996, *Information technology – Open Systems Interconnection – Generic upper layers security: Overview, models and notation*.
- ITU-T Recommendation X.833 (1995) | ISO/IEC 11586-4:1996, *Information technology – Open Systems Interconnection – Generic upper layers security: Protecting transfer syntax specification*.

2.2 Paired Recommendations | International Standards equivalent in technical content

- CCITT Recommendation X.800 (1991), *Security architecture for Open Systems Interconnection for CCITT applications*.
ISO 7498-2:1989, *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture*.

3 Definitions

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

3.1 OSI Reference Model Definitions

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

- a) *application-context*;
- b) *application-entity*;
- c) *application-process*.

3.2 Basic directory definitions

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

- a) *Directory*;
- b) *Directory Access Protocol*;
- c) *Directory Information Base*;
- d) *Directory Operational Binding Management Protocol*;
- e) *Directory System Protocol*;
- f) *(Directory) user*.

3.3 Distributed operation definitions

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

- a) *access point*;
- b) *hierarchical operational binding*;
- c) *name resolution*;
- d) *non-specific hierarchical operational binding*;
- e) *relevant hierarchical operational binding*.

3.4 Replication definitions

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

- a) *cache-copy*;
- b) *consumer reference*;
- c) *entry-copy*;
- d) *master DSA*;
- e) *primary shadowing*;
- f) *replicated area*;
- g) *replication*;
- h) *secondary shadowing*;
- i) *shadow consumer*;
- 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 S for easy reference.

4 Abbreviations

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

ACDF	Access Control Decision Function
ACI	Access Control Information
ACIA	Access Control Inner Area
ACSA	Access Control Specific Area
ADDMD	Administration Directory Management Domain
ASN.1	Abstract Syntax Notation One
AVA	attribute value assertion
BER	(ASN.1) Basic Encoding Rules
DACD	Directory Access Control Domain
DAP	Directory Access Protocol
DIB	Directory Information Base
DISP	Directory Information Shadowing Protocol
DIT	Directory Information Tree
DMD	Directory Management Domain
DMO	Domain Management Organization
DOP	Directory Operational Binding Management Protocol
DSA	Directory System Agent
DSE	DSA-Specific Entry
DSP	Directory System Protocol
DUA	Directory User Agent
HOB	Hierarchical Operational Binding
NHOB	Non-specific Hierarchical Operational Binding
NSSR	Non-Specific Subordinate Reference
PRDMD	Private Directory Management Domain
RHOB	Relevant Hierarchical Operational Binding (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 "Rules for presentation of ITU-T | ISO/IEC common text" in the Guide for ITU-T and ISO/IEC JTC 1 Cooperation, October 1996.

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

This Directory Specification uses the term "1988 edition systems" to refer to systems conforming to the first edition of the Directory Specifications, i.e. the 1988 edition of the series of CCITT X.500 Recommendations and the ISO/IEC 9594:1990 edition. This Directory Specification uses the term "1993 edition systems" to refer to systems conforming to the second (1993) edition of the Directory Specifications, i.e. the 1993 edition of the series of ITU-T X.500 Recommendations and the ISO/IEC 9594:1995 edition. This Directory Specification uses the term "1997 edition systems" to refer to systems conforming to the third edition of the Directory Specifications, i.e. the 1997 edition of the series of ITU-T X.500 Recommendations and the ISO/IEC 9594:1998 edition. This Directory Specification uses the term "4th edition systems" to refer to systems conforming to this fourth edition of the Directory Specifications, i.e. the 2001 editions of ITU-T X.500, X.501, X.511, X.518, X.519, X.520, X.521, X.525, and X.530, the 2000 edition of ITU-T X.509, and parts 1-10 of the ISO/IEC 9594:2001 edition.

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

SECTION 2 – OVERVIEW OF THE DIRECTORY MODELS

6 Directory Models

6.1 Definitions

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

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

6.1.2 administration directory management domain (ADDMD): A DMD which is managed by an Administration.

NOTE – The term Administration denotes a public telecommunications administration or other organization offering public telecommunications services.

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

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

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

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

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

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

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

6.1.10 directory user agent (DUA): An OSI application process which represents a user in accessing the Directory.

NOTE – DUAs may also provide a range of local facilities to assist users compose queries and interpret the responses.

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

6.2 The Directory and its Users

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

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

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 21.2.

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

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

NOTE 3 – A DUA may exhibit local behaviour and structure which is outside the scope of envisaged Directory Specifications. For example, a DUA which represents a human directory user may provide a range of local facilities to assist its user to compose queries and interpret the responses.

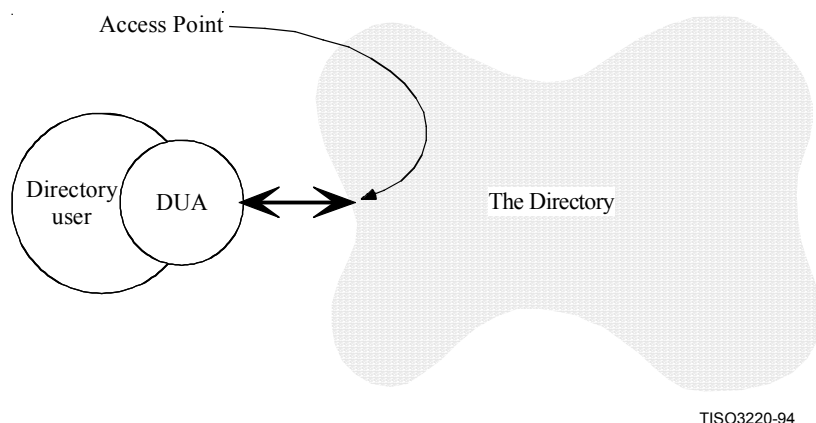


Figure 1 – Access to the Directory

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 22 through 23.

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 standardization 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 11.5). A DIT Domain shall not contain the root of the global DIT.
- The term DMD may also be used as a general term when both aspects of management are considered together.

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

NOTE 1 – A DMO may be an Administration (i.e. a public telecommunications administration or other organization offering public telecommunications services) in which case the managed DMD is said to be an Administration DMD (ADDMD); otherwise, it is a Private DMD (PRDMD). It should be recognized that the provision of support for private directory systems by ITU-T members falls within the framework of national regulations. Thus, the technical possibilities described may or may not be offered by an Administration which provides directory services. The internal operation and configuration of private DMDs is not within the scope of envisaged Directory Specifications.

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

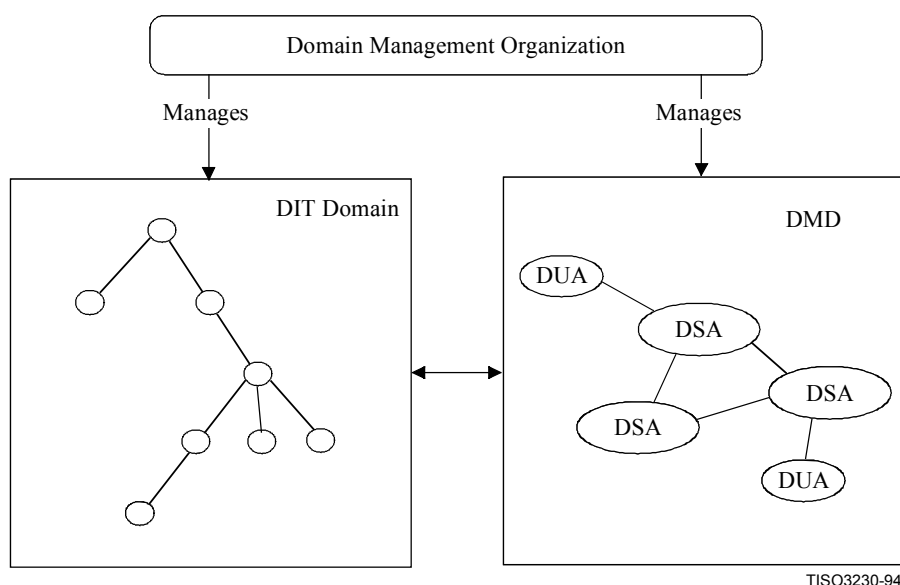


Figure 2 – Directory Management

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

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

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

A DMD may be assigned an object identifier (a DMD-id) for convenience in reference, for example in search-rules.

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 ancestor: The entry at the root of the hierarchy of family members that comprise a compound entry.

7.1.3 compound entry: A representation of an object in terms of family members that are hierarchically organized into one or more families of entries.

7.1.4 derived entry: Entry information in a search result containing attribute values obtained by performing a join on data that originated from more than one directory entry.

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

7.1.6 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.

7.1.7 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.

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

7.1.9 family: A hierarchical subset of family member entries that represents a particular class of information within a compound entry. The root of each family within a compound entry is the ancestor, but apart from the shared ancestor, families do not share common members. A family is distinguished from other families within a compound entry by having a common class (structural object class) for each family member that is immediately subordinate to the ancestor.

7.1.10 family member: A member of a hierarchical collection of entries that comprise a compound entry.

7.1.11 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.

7.1.12 immediately superior entry: Relative to a particular entry – an entry which is at the initial vertex of an arc in the DIT whose final vertex is that of the particular entry.

7.1.13 immediately superior 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.

7.1.14 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.

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

7.1.16 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.

7.1.17 related entries: A set of (directory) entries, each of which can be identified as holding information in the DIB about a particular real-world object of interest. Different entries in the set may contain different types of information about the real-world object, and may even contain conflicting information.

NOTE 1 – The value of information within a set of related entries depends on the reliability of the identification of each entry with the real-world.

NOTE 2 – It is possible, but not necessary, for related entries to exist in disjoint views of the directory and to have identical distinguished names. Similarly, it is possible for non-related entries to have identical distinguished names; however, it is recommended that identical distinguished names be used only for related entries.

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

7.1.19 subordinate: The converse of superior.

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

7.1.21 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).

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

NOTE 2 – The objects known to the Directory may not correspond exactly with the set of 'real' things in the world. For example, a real-world person may be regarded as two different objects, a business person and a residential person, as far as the Directory is concerned. The mapping is not defined in this Directory Specification, but is a matter for the users and providers of the Directory in the context of their applications.

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

7.3 Directory Entries

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

There are four 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 or compound entry (see 8.10). The object entry is said to represent the object. An object entry is either a single entry or a compound entry comprising an aggregate of entries together representing the object
- *Alias entries*: Used to provide alternative names for object entries (possibly the ancestor of a compound entry, but not child family members).
- *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.
- *Family members*: Special entries that are components of a compound entry. The ancestor of a compound entry is also a family member.

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 I.

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 13.7).

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 12.

8 Directory Entries

8.1 Definitions

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

- 8.1.1 attribute:** Information of a particular type. Entries are composed of attributes.
- 8.1.2 user attribute:** An attribute representing user information.
- 8.1.3 attribute hierarchy:** The aspect of an attribute that permits a user attribute type to be derived from a more generic user attribute type. The relationship of the two attribute type definitions (which mandates certain behaviour of attributes corresponding to these attribute types) is thus hierarchical.
- 8.1.4 attribute subtype (subtype):** An attribute type A is related to another attribute type B by the fact that either A has been derived from B, in which case A is a *direct* subtype of B, or A has been derived from an attribute type which is a subtype of B, in which case A is an *indirect* subtype of B.
- 8.1.5 attribute supertype (supertype):** An attribute type B is related to another attribute type A by the fact that either A has been derived from B, in which case B is a *direct* supertype of A, or A has been derived from an attribute type which is a subtype of B, in which case B is an *indirect* supertype of A.
- 8.1.6 attribute type:** That component of an attribute which indicates the class of information given by that attribute.
- 8.1.7 attribute value:** A particular instance of the class of information indicated by an attribute type.
- 8.1.8 attribute value assertion:** A proposition, which may be true, false, or undefined, according to the specified matching rules for the type, concerning the presence in an entry of an attribute value of a particular type.
- 8.1.9 auxiliary object class:** An object class which is descriptive of entries or classes of entries and is not used for the structural specification of the DIT.
- 8.1.10 collective attribute:** A user attribute whose values are the same for each member of an entry collection.
- 8.1.11 context:** A property that can be associated with a user attribute value to specify information that can be used to determine the applicability of the value.
- 8.1.12 context assertion:** A proposition, which may be true or false, regarding a context type and particular context values for that type, that determines the applicability of an attribute value.
- 8.1.13 context type:** That component of a context which indicates its type or purpose.
- 8.1.14 context list:** The set of contexts associated with an attribute value.
- 8.1.15 context value:** A particular instance of the property indicated by a context type.
- 8.1.16 derived attribute:** An attribute whose value or values is computed in whole or in part rather than directly stored.
- 8.1.17 derived object class value:** A value of an object class whose presence is not administered by a user but is computed. Derived object class values are categorized as abstract.

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

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

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

8.1.21 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.

8.1.22 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.

8.1.23 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.

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

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

8.1.26 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*.

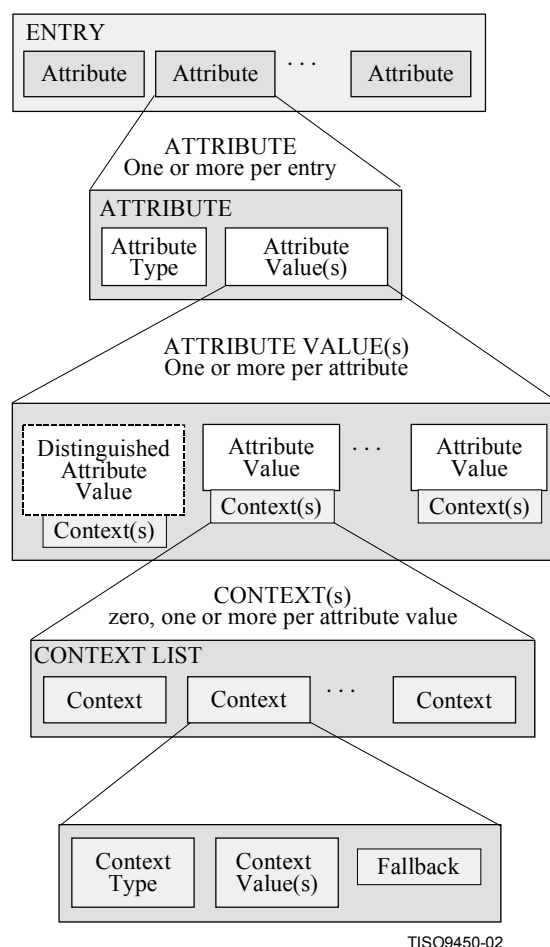


Figure 3 – Structure of an entry

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. A user attribute value may have zero, one, or more contexts associated with it in its context list. Operational attribute values shall not have contexts.

NOTE 2 – Attribute types, attribute values, and contexts are described in 8.4, 8.5 and 8.7 respectively. Operational attributes are described in clause 12.

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

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

8.3 Object Classes

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

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

Some object classes will be internationally standardized. Others will be defined by national administrative authorities and/or private organizations. This implies that a number of separate authorities will be responsible for defining object classes and unambiguously identifying them. This is accomplished by identifying each object class with an object identifier when the object class is defined. A notation for this purpose is provided in 13.3.3.

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 entry's or family member's 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 shall 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 mainly 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.

In addition to its use for deriving other object classes, an abstract object class value can be a derived value; that is, its presence is computed or inferred by the Directory. For example, the **parent** object class value for a particular entry is computed or inferred from the presence of a family member, of auxiliary object class **child**, immediately subordinate to the entry.

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

Structural object classes are related to associated entries:

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

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

8.3.3 Auxiliary Object Classes

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

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

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

This requirement may also be met using the facilities of DIT content rule definitions to dynamically (i.e. without registration) allow the addition or exclusion of attributes from entries at particular points in the DIT (see 13.3.3).

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 13.4.7, an attribute type is defined as:

AttributeType ::= ATTRIBUTE.&id

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

Certain attributes may not be stored and accessible in entries, but are intended to be carried in operations to convey information, e.g. diagnostics information, that conveniently can be expressed as attributes. Other attributes, called *control attributes*, may as part of their definition specify a special procedure to be executed based on the information in the attribute. A control attribute may be specified in an operation, placed in entries, etc. See 6.5.3 of ITU-T Rec. X.520 | ISO/IEC 9594-6 for an example.

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 13.4.7.

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 13.8). 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 13.4.7, an attribute value is defined as:

AttributeValue ::= ATTRIBUTE.&Type

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

8.6 Attribute Type Hierarchies

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

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

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

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

For an entry to contain a value of an attribute type belonging to an attribute hierarchy, that type shall 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 Contexts

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

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

Each context consists of a type field, a value field whose syntax is determined by the type, and a **fallback** flag. Using the notation of the **CONTEXT** information object class defined in 13.9, a Context is defined as:

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

contextType is an **OBJECT IDENTIFIER**, and is specified using the **CONTEXT** information object class defined in 13.9. It specifies the particular property represented by the Context.

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

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

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

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

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

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

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

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

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

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

8.8 Matching Rules

8.8.1 Overview

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

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

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

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

A matching rule is defined through the specification of:

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

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

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

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

8.8.2 Attribute Value Assertions

An attribute value assertion (AVA) is a proposition, which may be true, false, or undefined, according to the specified matching rules for the type, concerning the presence in an entry of an attribute value of a particular type. It involves an attribute type, an asserted attribute value, and optionally an assertion about contexts associated with the attribute value:

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

```

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

```

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

8.8.2.1 Evaluation of an AVA

An AVA is:

- a) undefined, if any of the following holds:
 - 1) the attribute type is unknown;
 - 2) the attribute type has no equality matching rule;
 - 3) the value does not conform to the data type indicated by the syntax of the assertion of the attribute's equality matching rule;

NOTE – 2) and 3) normally indicate a faulty AVA; 1) however, may occur as a local situation (e.g. a particular DSA has not been configured with support for that particular attribute type).
- b) true, if the entry contains an attribute of that type, and the attribute contains a value of that value, and the value contains a context that matches the **assertedContexts** as described in 8.8.2.2;
- c) false, otherwise.

8.8.2.2 Use of assertedContexts or context assertion defaults

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

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

- 1) If **assertedContexts** is not provided within an **AttributeValueAssertion**, then any context assertion for the given attribute type which has been supplied for the operation as a whole, as part of **operationContexts** as described in 7.3 of ITU-T Rec. X.511 | ISO/IEC 9594-3, shall be applied.
- 2) If the user has not provided **assertedContexts** for the AVA and there is no context assertion for the given attribute type which has been supplied for the operation as a whole, then the default context assertion for the given attribute type in the context assertion subentries (if any) controlling the entry shall be applied, as described in 14.7.
- 3) If there is no context assertion through steps 1) and 2) above, the DSA may apply a locally-defined default context assertion for the given attribute type. Such a default shall typically reflect local parameters, such as the language or location of the place of deployment of the DSA, or the current time of day, but may be tailored differently by the DSA for each DUA to which it responds.
- 4) If no context assertion is available from any of these sources, then the **assertion** shall be evaluated against all values of the attribute.

8.8.2.3 Evaluation of assertedContexts

assertedContexts is true if:

- a) **allContexts** is specified (this permits a context assertion to override any default context assertion that might otherwise be applied if **assertedContexts** were omitted from the **AttributeValueAssertion**); or
- b) each **ContextAssertion** in **selectedContexts** is true as described in 8.8.2.4.

assertedContexts is false otherwise.

8.8.2.4 Evaluation of a ContextAssertion

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

- a) the attribute value has a context of the same **contextType** of the **ContextAssertion** and any of the stored **contextValues** of that context matches with any of the asserted **contextValues** according to the definition of how a match is determined for that **contextType**; or
- b) the attribute value contains no contexts of the asserted **contextType**; or

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

A **ContextAssertion** is false otherwise.

8.8.3 Attribute Type Assertions

An attribute type assertion is a proposition, which may be true, false, or undefined, according to the associated contexts.

AttributeTypeAssertion ::= SEQUENCE {
 type **ATTRIBUTE.&id ({SupportedAttributes}),**
 assertedContexts **SEQUENCE SIZE (1..MAX) OF ContextAssertion OPTIONAL }**

8.8.3.1 Evaluation of an attribute type assertion

An attribute type assertion is:

- a) undefined, if the attribute type is unknown or if the attribute is not present in the entry;
- b) **TRUE**, if the entry contains an attribute of that type, and the attribute contains one or more values that contain a context that matches the **assertedContexts** as described in 8.8.3.2;
- c) **FALSE**, otherwise.

8.8.3.2 Use of assertedContexts or context assertion defaults

The inclusion of **assertedContexts** within an **AttributeTypeAssertion** is optional. If **assertedContexts** is specified, the **assertedContexts** shall be true for at least one attribute value according to the rules defined in 8.8.2.4.

If **assertedContexts** is not provided within an **AttributeTypeAssertion**, then a default context assertion may be applied in the same manner; that is, the default context assertion shall be true for at least one attribute value according to the rules defined in 8.8.2.4. The potential sources for a default context assertion are as specified in 8.8.2.2.

8.8.4 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.

8.8.5 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** matches 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** matches 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) shall 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 shall 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 precedes 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) shall 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 equals the result of evaluating the assertion against the value y. That is, two values that are indistinguishable using the equality relation are also indistinguishable using the substrings relation.

8.8.6 Object Identifier and Distinguished Name equality matching rules

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

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

8.9 Entry Collections

8.9.1 Overview

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

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

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

8.9.2 Collective Attributes

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

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 shall be administered via their associated subtrees.

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 12.7 and defined in 14.6.

8.10 Compound entries and families of entries

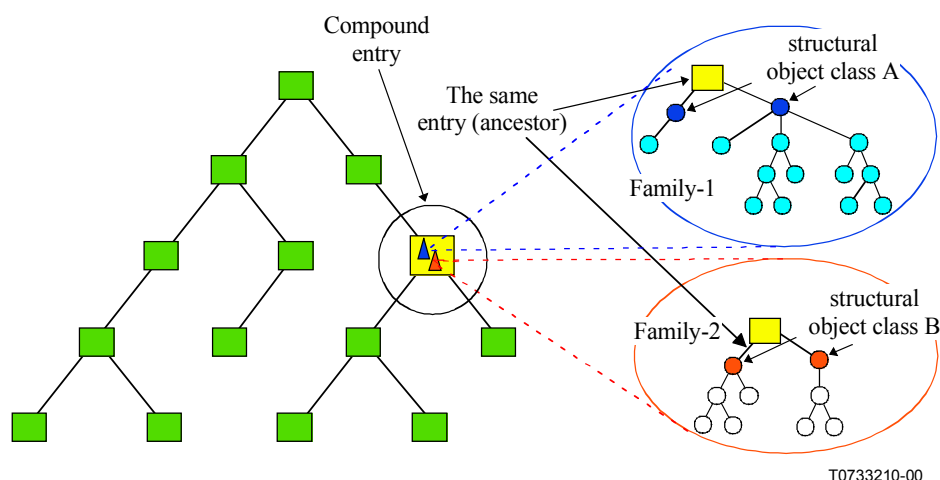


Figure 4 – Families of entries

A compound entry is a special entry that comprises family member entries. These family members form a hierarchy and thereby provide hierarchically organized information about the object represented by the compound entry. The compound entry is represented in the DIT by an ancestor family member, which is at the root of a tree containing the family members.

Family members can themselves be organized into one or more families for the purposes of filtering and information retrieval. Each family is a subtree; distinct families have no common family members apart from the shared root that is the ancestor. A family thus comprises an ancestor plus a set of subordinate family members.

A family is, beside the ancestor, composed of all of the immediately subordinate family members being of the same structural object class. Their subordinate members, if any, are also part of the same family independent of their structural object classes.

These concepts are illustrated in Figure 4.

A family member that is a child within a family tree is marked with the auxiliary object class **child**. The presence of the **child** object class value for an entry causes the immediately superior entry automatically to be marked with the abstract object class value **parent**. An entry that is both a **parent** and a **child** within a family tree is marked with both object class values. The ancestor is the only family member that is not of object class **child**. The construction of compound entries is carried out by marking entries with **child** object class values.

Each subordinate of a non-ancestor family member shall itself be a family member, and marked with a **child** object class value.

The ASN.1 definition of these object classes can be found in 13.3.3.

All family members of a compound entry shall be placed in the same naming context as the ancestor. Family members are not permitted to be alias entries. An alias shall not point to a child family member.

9 Names

9.1 Definitions

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

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

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

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

9.1.4 (directory) name: A construct that singles out a particular object from all other objects. A name shall be unambiguous (that is, denote just one object), however it need not be unique (that is, be the only name which unambiguously denotes the object).

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

9.1.6 local member name: A name for a family member constructed by the sequence of RDNs from the ancestor down to the member in question not including the RDN for the ancestor.

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

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

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

9.2 Names in General

A (*directory*) *name* is a construct that identifies a particular object from among the set of all objects. A name shall be unambiguous, that is, denotes just one object. However, a name need not be unique, that is, be the only name that unambiguously denotes the object. A (*directory*) name also identifies an entry. This entry is either an object entry that represents the object or an alias entry which contains information that helps the Directory to locate the entry that represents the object.

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

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

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

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

RDNSequence ::= SEQUENCE OF RelativeDistinguishedName

DistinguishedName ::= RDNSequence

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

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

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

9.3 Relative Distinguished Names

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

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

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

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

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

AttributeTypeAndDistinguishedValue ::= SEQUENCE {

type	ATTRIBUTE.&id ({SupportedAttributes}),
value	ATTRIBUTE.&Type({SupportedAttributes}{@type}),
primaryDistinguished	BOOLEAN DEFAULT TRUE,
valuesWithContext	SET SIZE (1..MAX) OF SEQUENCE {
distingAttrValue	[0] ATTRIBUTE.&Type ({SupportedAttributes}{@type}) OPTIONAL,
contextList	SET SIZE (1..MAX) OF Context } OPTIONAL }

ATTRIBUTE.&Type ({SupportedAttributes}{@type}) OPTIONAL,

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

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

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

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

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

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

Family members, like other entries, have RDNs. An RDN can consist of multiple attribute type and value pairs. Only primary RDNs can be used. The *local member name* of a family member is the sequence of RDNs from the ancestor down to that member. The local member name of the ancestor is an empty sequence.

NOTE 2 – RDNs are intended to be long-lived so that the users of the Directory can store the distinguished names of objects (e.g. in the Directory itself) without concerns for their obsolescence. Thus RDNs should be changed cautiously.

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

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

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

9.4 Name Matching

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

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

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

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

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

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

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

9.5 Names returned during operations

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

9.6 Names held as attribute values or used as parameters

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

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

9.7 Distinguished Names

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

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

NOTE 2 – ITU-T Rec. X.650 | ISO/IEC 7498-3 defines the concept of a primitive name. A distinguished name can be used as a primitive name for the object it identifies.

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

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

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

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

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

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

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

An example which illustrates the concepts of RDN and distinguished name appears in Figure 5.

9.8 Alias Names

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

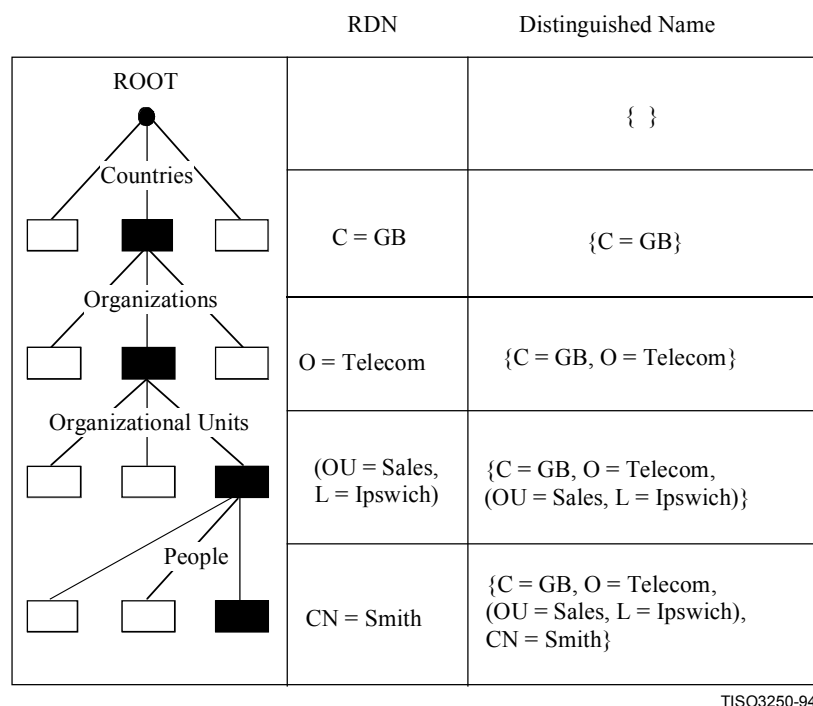


Figure 5 – Determination of distinguished names

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

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

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

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

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

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

Every alias entry shall belong to the **alias** object class which is defined in 13.3.3.

Family members are not permitted to be alias entries.

10 Hierarchical groups

10.1 Definitions

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

10.1.1 hierarchical child: For an entry, a hierarchical child is an entry for which it is a hierarchical parent.

10.1.2 hierarchical group: A hierarchical group is a collection of entries, including compound entries, that forms a logical tree that is not necessarily related to the DIT.

10.1.3 hierarchical leaf: This is an entry within a hierarchical group that has no hierarchical children.

10.1.4 hierarchical level: An integer that gives the distance from an entry within a hierarchical group to the hierarchical top in form of the number of hierarchical links between the entry and the hierarchical top.

10.1.5 hierarchical link: This is a general term for the logical relationship between two entries that have a hierarchical immediate parent/immediate child relationship.

10.1.6 hierarchical parent: For an entry, the hierarchical parents are the immediately hierarchical parent, its immediately hierarchical parent, recursively all the way up to and including the hierarchical top.

10.1.7 hierarchical sibling: For an entry, the hierarchical siblings are the entries having the same immediately hierarchical parent as itself.

10.1.8 hierarchical sibling-child: For an entry, its hierarchical sibling-children are the complete set of hierarchical children, at all lower levels, of its hierarchical siblings.

10.1.9 hierarchical top: This is the entry within a hierarchical group that is the root of the hierarchy. A hierarchical top has no immediately hierarchical parent.

10.1.10 immediately hierarchical child: For an entry, an immediately hierarchical child is an entry for which it is the immediately hierarchical parent. This immediately hierarchical child does not need to be an immediately subordinate entry within the DIT.

10.1.11 immediately hierarchical parent: For an entry, its immediately hierarchical parent is the entry, that within the hierarchical group, is its immediately superior entry. The immediately hierarchical parent does not need to be the immediately superior entry within the DIT.

10.2 Hierarchical relationship

Directory entries have a hierarchical relationship in the way they are placed in the DIT. However, entries may also have hierarchical relationships not reflected in DIT structure. As an example, a dynamic organization may not want to reflect its current organization directly in the DIT, as it may require frequent changes to the DIT structure. There is therefore a requirement in the Directory to be able to reflect hierarchical relationships independent of the DIT structure. *Hierarchical groups* form such relationships. A hierarchical group forms a logical tree with a root called the *hierarchical top*.

By referring to hierarchical relationships, it is possible in a Search operation to retrieve information not only from a given entry, but also from other entries within the same hierarchical group.

A compound entry is considered a single entry in the context of hierarchical groups. A child family member cannot be part of a hierarchical group in its own right.

NOTE – Hierarchical groups are intended to permit modelling of collections of distinct objects that have logically informal relationships, and particularly relationships that are, or could be, temporary. Compound entries, in contrast, model objects that comprise sub-objects that are conveniently considered as a hierarchy.

To describe navigation within a hierarchical group it is convenient to define terms for the relationships that a given entry has with other entries within the group. This is done in 10.1. Some of these defined terms for direct relationships are parallel to those defined for entry relationships within the DIT (*immediately hierarchical child*, *hierarchical child*, *immediately hierarchical parent* and *hierarchical parent*). However, it is also convenient to define terms for more distant relationships. In some situation, a user may want to retrieve information for *hierarchical siblings*, and even for their hierarchical children (*hierarchical sibling-children*).

An entry can only be a member of a single hierarchical group at one time.

An entry that is part of a hierarchical group holds operational attributes as defined in 14.9. These operational attributes reflect the relationship with other entries within the group, including the *hierarchical level* of the entry within the group. When a compound entry is part of a hierarchical group, the ancestor holds these operational attributes.

A hierarchical group has to be completely outside any service specific administrative area (see 16.3) or has to be completely contained within a service specific administrative area. A hierarchical group shall be confined to a single DSA. The Directory service shall detect and prevent attempts to break these rules.

SECTION 4 – DIRECTORY ADMINISTRATIVE MODEL

11 Directory Administrative Authority model**11.1 Definitions**

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

- 11.1.1 administrative area:** A subtree of the DIT considered from the perspective of administration.
- 11.1.2 administrative entry:** An entry located at an administrative point.
- 11.1.3 administrative point:** The root vertex of an administrative area.
- 11.1.4 administrative user:** A representative of an Administrative Authority. The full definition of the administrative user concept is outside the scope of this Directory Specification.
- 11.1.5 autonomous administrative area:** A subtree of the DIT whose entries are all administered by the same Administrative Authority. Autonomous administrative areas are non-overlapping.
- 11.1.6 DIT domain administrative authority:** An Administrative Authority in its role as the entity having responsibility for the administration of a part of the DIT.
- 11.1.7 DIT domain policy:** An expression of the general goals and acceptable procedures for a DIT Domain.
- 11.1.8 DMD administrative authority:** An Administrative Authority in its role as the entity responsible for the administration of a DMD.
- 11.1.9 DMD policy:** A policy governing the operation of the DSAs in a DMD.
- 11.1.10 DMO policy:** A policy defined by a DMO, expressed in terms of DMD and DIT Domain policies.
- 11.1.11 inner administrative area:** A specific administrative area whose scope is wholly contained within the scope of another specific administrative area of the same type.
- 11.1.12 policy:** An expression by an Administrative Authority of general goals and acceptable procedures.
- 11.1.13 policy attribute:** A generic term for any Directory operational attribute which expresses policy.
- 11.1.14 policy object:** An entity with which a policy is concerned.
- 11.1.15 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.
- 11.1.16 policy parameter:** A policy procedure is characterized by certain *policy parameters* which are subject to configuration (i.e. choice) by an Administrative Authority.
- 11.1.17 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.
- 11.1.18 specific administrative point:** The root vertex of a specific administrative area.

11.2 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 11.3, 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 12). This is to allow the modelling of information required for the regulation of Directory user information and for other administrative purposes.

Other aspects of the Directory Administrative Model require support for the distribution of administrative and operational information among the component parts of the Directory, i.e. DSAs. Clauses 22 through 24 describe a DSA Information Model to support these requirements.

11.3 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 11.6.

NOTE – Clause 14 defines the system schema necessary to support the administration of collective attributes. Clause 15 defines a framework for supporting subschema administration policies. Clause 17 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 11.7 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 15 defines subschema administration policy procedures).

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

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

11.4 Specific administrative authorities

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

- naming administration;
- subschema administration;
- security administration;
- collective attribute administration;
- service 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 15 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 12.7) in a DIT Domain.

The term *Service Authority* identifies the role of the Administrative Authority as it pertains to the establishment and administration of service constraints and adjustment.

11.5 Administrative areas and administrative points

11.5.1 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 6.

11.5.2 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 five kinds of specific administrative area:

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

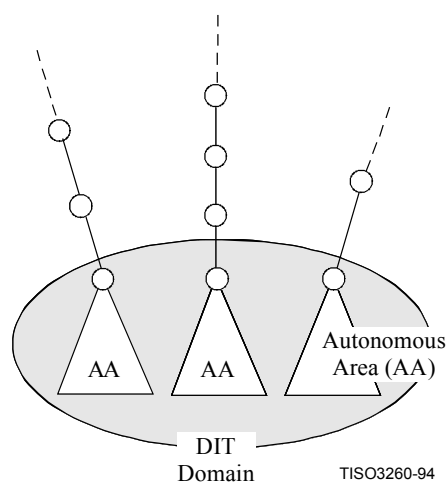


Figure 6 – A DIT Domain

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.

11.5.3 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, shall be defined as part of the definition of the specific administrative aspect within which they are contained.

11.5.4 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 14.3.

Clauses 22 through 24 describe how administrative areas are mapped onto DSAs and the DSA information model.

Figure 7 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 7 uses the abbreviations AAP (Autonomous Administrative Point), SAP (Specific Administrative Point) and IAP (Inner Administrative point).

11.5.5 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.

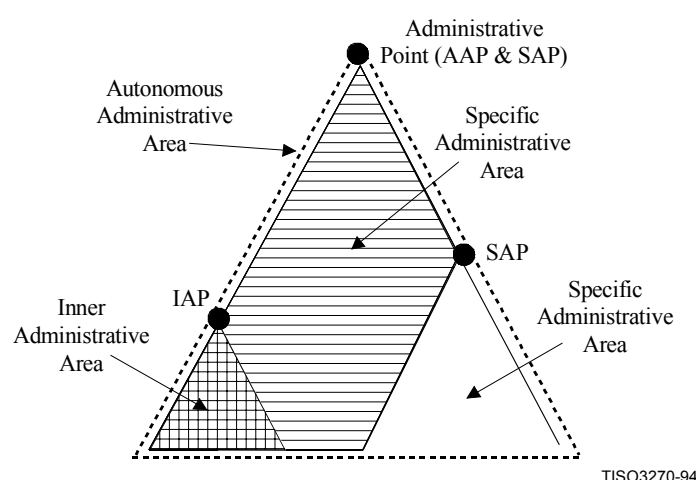


Figure 7 – Administrative points and areas

11.6 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 14 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 shall take such circumstances into account as part of its definition.

Specific DIT Domain policy objects and attributes are defined in clause 15 to support subschema administration.

11.7 DMD policies

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

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

Examples of such restrictions are:

- a) Limiting the basic service provided to Directory (i.e. non-administrative) users to interrogation operations only, in accordance with CCITT Rec. F.500.
- b) Limiting the service provided to users accessing the DSA indirectly, via chaining, including distinctions based on whether the user request traversed a trusted path.
- c) Limitations on requests accepted from users accessing the DSA directly when chaining is required to DSAs in the DMD known to be subject to limitations of the kind indicated in the previous point.
- d) Constraints on the kinds of searches certain users can perform, and on the characteristics of such searches (e.g. relaxation policies).

SECTION 5 – MODEL OF DIRECTORY ADMINISTRATIVE AND OPERATIONAL INFORMATION

12 Model of Directory Administrative and Operational Information**12.1 Definitions**

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

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

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

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

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

12.1.5 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.

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

12.1.7 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.

12.1.8 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.

12.1.9 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.

12.2 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 8, may be placed in the Directory by administrative authorities or by DSAs, and is used by the Directory in the course of its operation.

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

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

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

Entries are made visible to Directory users via the Directory abstract service, but their relationships to the DSAs that ultimately hold them are not. The DSA information model, described in clauses 22 through 24, expresses the mapping of these entries onto the information repositories of DSAs.

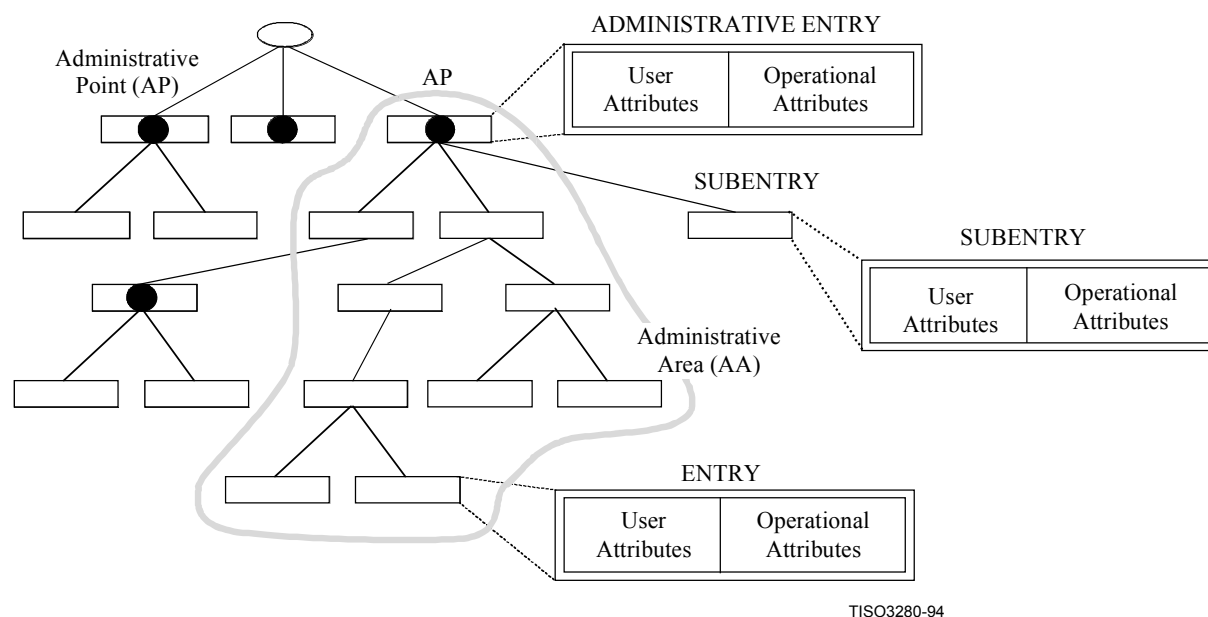


Figure 8 – Model of Directory Administrative and Operational Information

12.3 Subtrees

12.3.1 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.

12.3.2 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:

- Base* – The root vertex of the subtree or subtree refinement produced by the evaluation of a subtree specification;
- Chop* – A set of assertions concerning the names of the subordinate entries; and
- 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 9, for the case where subtrees are deployed within the context of administrative areas.

12.3.3 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).

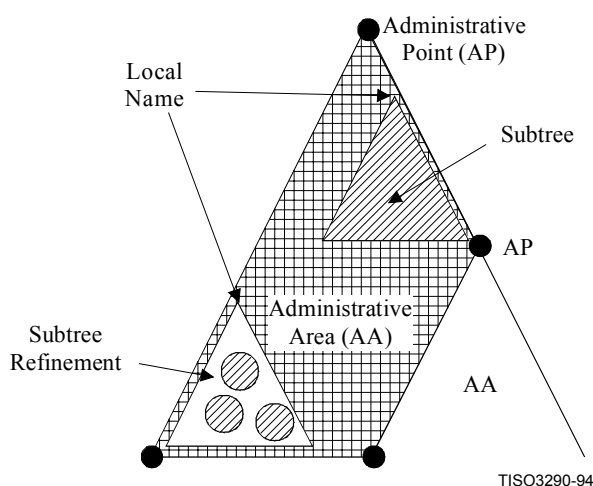


Figure 9 – Specification of Subtrees and Subtree Refinements within the context of Administrative Areas

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

LocalName ::= RDNSequence

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

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

12.3.4 Chop Specification

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

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

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

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

12.3.4.1 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.

12.3.4.2 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)

For the purpose of chop specifications, a compound entry is counted as a single entry. In a compound entry, all family members are counted as having the same base distance as the ancestor, since they are all part of the same logical entry.

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.

12.3.5 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.

If a family member is excluded from a subtree by this specification, all its subordinate family members are also excluded.

12.4 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 23 through 24.

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

12.5 Entries

12.5.1 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 14) applicable to the entry.

12.5.2 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.

12.6 Subentries

12.6.1 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 21).

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 10.

A subentry consists of:

- a **commonName** attribute, specified in ITU-T Rec. X.520 | ISO/IEC 9594-6 which contains the RDN of the subentry;
- a **subtreeSpecification** attribute, specified in clause 14;
- an **objectClass** attribute, specified in clause 13, 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 12.6.4).

12.6.2 Subentry RDN attribute

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

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

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

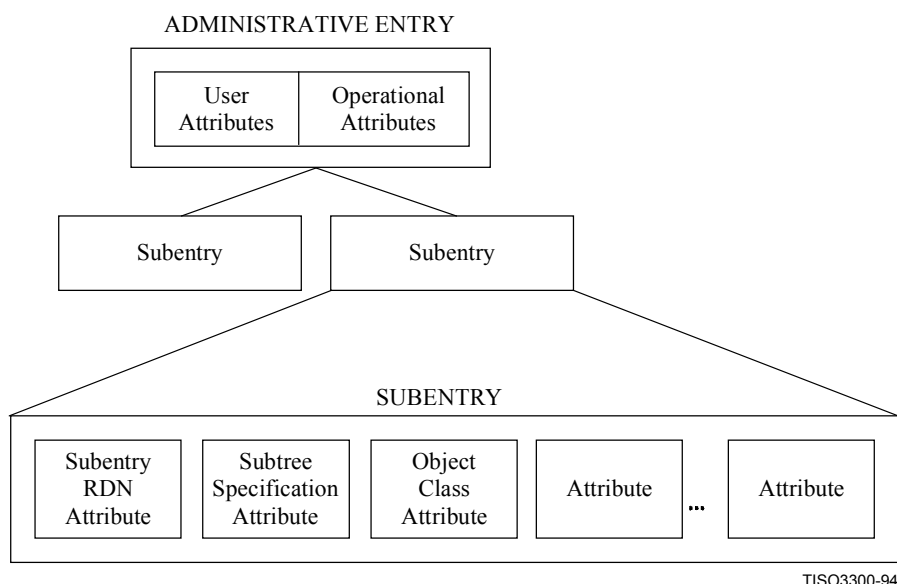


Figure 10 – Structure of a Subentry

12.6.3 Subtree Specification attribute

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

12.6.4 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 14, 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 shall 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**.

12.6.5 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.

12.7 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 14. 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 shall 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 shall 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 14.

12.8 Information model for context defaults

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

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

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

Context defaults are represented in the Directory Information model by a subentry, termed a *context default subentry*, whose **objectClass** attribute has the value **id-sc-contextAssertionSubentry** as defined in 14.7. A subentry of this class may be the immediate subordinate of an administrative entry whose **administrativeRole** attribute contains the value **id-ar-contextDefaultSpecificArea**.

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