FINAL
DRAFT

INTERNATIONAL STANDARD

ISO/IEC FDIS 9594-3

ISO/IEC JTC **1**

Secretariat: **ANSI**

Voting begins on:
2009-03-05

Voting terminates on:
2009-05-05

Information technology — Open Systems Interconnection — The Directory: Abstract service definition

Technologies de l'information — Interconnexion de systèmes ouverts (OSI) — L'annuaire: Définition du service abstrait

Please see the administrative notes on page ii-1

RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

Reference number
ISO/IEC FDIS 9594-3:2008(E)

© ISO/IEC 2008

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

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.

Copyright notice

This ISO document is a Draft International Standard and is copyright-protected by ISO. Except as permitted under the applicable laws of the user's country, neither this ISO draft nor any extract from it may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, photocopying, recording or otherwise, without prior written permission being secured.

Requests for permission to reproduce should be addressed to 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.org
Web www.iso.org

Reproduction may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

© ISO/IEC 2008 – All rights reserved

Not for Resale

In accordance with the provisions of Council Resolution 21/1986, this document is **circulated in the English language only**.

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Blank page

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

CONTENTS

Page

1 Scope 1
2 Normative references 1
2.1 Identical Recommendations | International Standards 1
2.2 Other references 2
3 Definitions 2
3.1 Basic Directory definitions 2
3.2 Directory model definitions 2
3.3 Directory information base definitions 2
3.4 Directory entry definitions 2
3.5 Name definitions 3
3.6 Distributed operations definitions 3
3.7 Abstract service definitions 3
4 Abbreviations 4
5 Conventions 4
6 Overview of the Directory service 4
7 Information types and common procedures 5
7.1 Introduction 5
7.2 Information types defined elsewhere 5
7.3 Common arguments 6
7.3.1 Critical extensions 7
7.3.2 Family grouping 9
7.4 Common results 9
7.5 Service controls 10
7.6 Entry information selection 12
7.6.1 Use of contextSelection or context selection defaults 14
7.6.2 Evaluation of contextSelection 15
7.6.3 Evaluation of a TypeAndContextAssertion 15
7.6.4 Family Return 15
7.7 Entry information 16
7.7.1 Entry information data type 16
7.7.2 Family information in entry information 17
7.8 Filter 18
7.8.1 Filter parameter 18
7.8.2 Filter item 19
7.8.3 Evaluating filters with family information 20
7.9 Paged results 21
7.10 Security parameters 23
7.11 Common elements of procedure for access control 24
7.11.1 Common elements of procedure for basic access control 25
7.11.2 Common elements of procedure for rule-based-access-control 26
7.11.3 Family information 26
7.12 Managing the DSA Information Tree 26
7.13 Procedures for families of entries 27
8 Bind and Unbind operations 28
8.1 Directory Bind 28
8.1.1 Directory Bind syntax 28
8.1.2 Directory Bind arguments 30
8.1.3 Directory Bind results 31
8.1.4 Directory Bind errors 31
8.2 Directory Unbind 31
9 Directory Read operations 31
9.1 Read 31
9.1.1 Read syntax 31

--`,,```,,,,````-`-`,,`,,`,`,,`---

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.1.2 Read arguments 32
9.1.3 Read results 32
9.1.4 Read errors 33
9.1.5 Read operation decision points for basic access control 33
9.1.6 Read operation decision points for rule-based access control 33
9.2 Compare 34
9.2.1 Compare syntax 34
9.2.2 Compare arguments 35
9.2.3 Compare results 35
9.2.4 Compare errors 35
9.2.5 Compare operation decision points for basic access control 35
9.2.6 Compare operation decision points for rule-based access control 36
9.3 Abandon 36
10 Directory Search operations 37
10.1 List 37
10.1.1 List syntax 37
10.1.2 List arguments 38
10.1.3 List results 38
10.1.4 List errors 40
10.1.5 List operation decision points for basic access control 40
10.1.6 List operation decision points for rule-based access control 40
10.2 Search 41
10.2.1 Search syntax 41
10.2.2 Search arguments 43
10.2.3 Search results 48
10.2.4 Service administration 49
10.2.5 Search errors 49
10.2.6 Search operation decision points for basic access control 49
10.2.7 Search operation decision points for rule-based access control 51
11 Directory Modify operations 52
11.1 Add Entry 52
11.1.1 Add Entry syntax 52
11.1.2 Add Entry arguments 52
11.1.3 Add Entry results 53
11.1.4 Add Entry errors 53
11.1.5 Add operation decision points for basic access control 53
11.1.6 Add Entry operation decision points for rule-based-access-control 54
11.2 Remove Entry 54
11.2.1 Remove Entry syntax 54
11.2.2 Remove Entry arguments 54
11.2.3 Remove Entry results 55
11.2.4 Remove Entry errors 55
11.2.5 Remove Entry operation decision points for basic access control 55
11.2.6 Remove Entry operation decision points for rule-based access control 55
11.3 Modify Entry 56
11.3.1 Modify Entry syntax 56
11.3.2 Modify Entry arguments 57
11.3.3 Modify Entry results 58
11.3.4 Modify Entry errors 58
11.3.5 Modify Entry operation decision points for basic access control 58
11.3.6 Modify Entry operation decision points for rule-based access control 59
11.4 Modify DN 60
11.4.1 Modify DN syntax 60
11.4.2 Modify DN arguments 60
11.4.3 Modify DN results 61
11.4.4 Modify DN errors 61
11.4.5 ModifyDN decision points for basic access control 61
11.4.6 Modify DN operation decision points for rule-based access control 62
12 Errors 62
12.1 Error precedence 62

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale
© ISO/IEC 2008 – All rights reserved

12.2 Abandoned 62
12.3 Abandon Failed 63
12.4 Attribute Error 63
12.5 Name Error 64
12.6 Referral 65
12.7 Security Error 66
12.8 Service Error 67
12.9 Update Error 68
13 Analysis of search arguments 70
13.1 General check of search filter 70
13.2 Check of request-attribute-profiles 71
13.3 Check of controls and hierarchy selections 73
13.4 Check of matching use 73
Annex A Abstract Service in ASN.1 75
Annex B Operational semantics for Basic Access Control 94
Annex C Examples of searching families of entries 108
C.1 Single family example 108
C.2 Multiple families example 109
C.2.1 Filter example 1 109
C.2.2 Filter example 2 110
C.2.3 Filter example 3 110
C.2.4 Filter example 4 111
Annex D External ASN.1 module 112
Annex E Amendments and corrigenda 119

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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 2.

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 document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 9594-3:2008 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.511 (11/2008).

This sixth edition cancels and replaces the fifth edition (ISO/IEC 9594-3:2005), which has been technically revised.

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*

--`,,```-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

© ISO/IEC 2008 – All rights reserved

Not for Resale

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 defines the capabilities provided by the Directory to its users.

This Recommendation | International Standard provides the foundation frameworks upon which industry profiles can be defined by other standards groups and industry forums. Many of the features defined as optional in these frameworks, may be mandated for use in certain environments through profiles. This sixth edition technically revises and enhances, but does not replace, the fifth edition of this Recommendation | International Standard. Implementations may still claim conformance to the fifth edition. However, at some point, the fifth edition will not be supported (i.e., reported defects will no longer be resolved). It is recommended that implementations conform to this sixth edition as soon as possible.

This sixth edition specifies versions 1 and 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 six editions, except for those specifically assigned to version 2, are accommodated using the rules of extensibility defined in ITU-T Rec. X.519 | ISO/IEC 9594-5.

Annex A, which is an integral part of this Recommendation | International Standard, provides the ASN.1 module for the Directory abstract service.

Annex B, which is not an integral part of this Recommendation | International Standard, provides charts that describe the semantics associated with Basic Access Control as it applies to the processing of a Directory operation.

Annex C, which is not an integral part of this Recommendation | International Standard, gives examples of the use of families of entries.

AnnexD, which is not an integral part of this Recommendation | International Standard, includes an updated copy of an external ASN.1 module referenced by this Directory Specification.

Annex E, 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.

--`,,```,,,,````-`-`,,`,,`,`,,`---

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION

Information technology – Open Systems Interconnection – The Directory: Abstract service definition

1 Scope

This Recommendation | International Standard defines in an abstract way the externally visible service provided by the Directory.

This Recommendation | International Standard does not specify individual implementations or products.

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 (2008) | ISO/IEC 9594-1:2008, *Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services*.
- ITU-T Recommendation X.501 (2008) | ISO/IEC 9594-2:2008, *Information technology – Open Systems Interconnection – The Directory: Models*.
- ITU-T Recommendation X.509 (2008) | ISO/IEC 9594-8:2008, *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks.*
- ITU-T Recommendation X.518 (2008) | ISO/IEC 9594-4:2008, *Information technology – Open Systems Interconnection – The Directory: Procedures for distributed operation*.
- ITU-T Recommendation X.519 (2008) | ISO/IEC 9594-5:2008, *Information technology – Open Systems Interconnection – The Directory: Protocol specifications.*
- ITU-T Recommendation X.520 (2008) | ISO/IEC 9594-6:2008, *Information technology – Open Systems Interconnection – The Directory: Selected attribute types.*
- ITU-T Recommendation X.521 (2008) | ISO/IEC 9594-7:2008, *Information technology – Open Systems Interconnection – The Directory: Selected object classes.*
- ITU-T Recommendation X.525 (2008) | ISO/IEC 9594-9:2008, *Information technology – Open Systems Interconnection – The Directory: Replication*.
- ITU-T Recommendation X.530 (2008) | ISO/IEC 9594-10:2008, *Information technology – Open Systems Interconnection – The Directory: Use of systems management for administration of the Directory.*
- ITU-T Recommendation X.680 (2008) | ISO/IEC 8824-1:2008, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*
- ITU-T Recommendation X.681 (2008) | ISO/IEC 8824-2:2008, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.*
- ITU-T Recommendation X.682 (2008) | ISO/IEC 8824-3:2008, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

– ITU-T Recommendation X.683 (2008) | ISO/IEC 8824-4:2008, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications*.

2.2 Other references

– RFC 2025 (1996), *The Simple Public-Key GSS-API Mechanism (SPKM)*.

– RFC 2222 (1997), *Simple Authentication and Security Layer (SASL)*.

3 Definitions

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

3.1 Basic Directory definitions

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

a) *Directory*;

b) *Directory Information Base*;

c) *(Directory) User*.

3.2 Directory model definitions

The following terms are defined in ITU-T Rec. X.501 | ISO/IEC 9594-2:

a) *Directory System Agent*;

b) *Directory User Agent.*

3.3 Directory information base definitions

The following terms are defined in ITU-T Rec. X.501 | ISO/IEC 9594-2:

a) *alias entry*;

b) *Directory Information Tree*;

c) *(Directory) entry*;

d) *immediate superior*;

e) *immediately superior entry/object*;

f) *object*;

g) *object class*;

h) *object entry*;

i) *subordinate*;

j) *superior*;

k) *ancestor*;

l) *family (of entries)*;

m) *compound entry*.

3.4 Directory entry definitions

The following terms are defined in ITU-T Rec. X.501 | ISO/IEC 9594-2:

a) *attribute*;

b) *attribute type*;

c) *attribute value*;

d) *attribute value assertion*;

e) *context*;

f) *context type*;

g) *context value*;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

h) *operational attribute*;

i) *user attribute*;

j) *matching rule*.

3.5 Name definitions

The following terms are defined in ITU-T Rec. X.501 | ISO/IEC 9594-2:

a) *alias, alias name*;

b) *distinguished name*;

c) *(directory) name*;

d) *purported name*;

e) *relative distinguished name*.

3.6 Distributed operations definitions

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

a) *bound DSA*;

b) *chaining*;

c) *initial performer*;

d) *referral*.

3.7 Abstract service definitions

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

3.7.1 additional search: A search that starts from **joinBaseObject** as specified by the originator in the **search** request.

3.7.2 contributing member: A family member within a compound entry, which made a contribution to either a Read, Search or Modify Entry operation.

3.7.3 explicitly unmarked entry: An entry or a family member that is excluded from the **SearchResult** according to a specification given in a control attribute referenced by the governing-search-rule.

3.7.4 family grouping: A set of members of a compound attribute that are grouped together for the purpose of operation evaluation.

3.7.5 filter: An assertion about the presence or value of certain attributes of an entry in order to limit the scope of a search.

3.7.6 originator: The user that originated an operation.

3.7.7 participation member: A family member that is either a contributing member or is a member of a family grouping that as a whole matched a **search** filter.

3.7.8 primary search: The search that starts from **baseObject** as specified by the originator in the search request.

3.7.9 relaxation: A progressive modification of the behaviour of a filter during a search operation so as to achieve more matched entries if too few are received, or fewer matched entries if too many are received.

3.7.10 service controls: Parameters conveyed as part of an operation, which constrain various aspects of its performance.

3.7.11 strand: A family grouping comprising all the members in a path from a leaf family member up to the ancestor inclusive. A family member will reside in as many strands as there are leaf family members below it (as immediate or non-immediate subordinates).

3.7.12 streamed result: A result of a single operation that is included in multiple responses.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4 Abbreviations

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

AVA Attribute Value Assertion
DIB Directory Information Base
DIT Directory Information Tree
DMD Directory Management Domain
DSA Directory System Agent
DUA Directory User Agent
RDN Relative Distinguished Name

5 Conventions

The term "Directory Specification" (as in "this Directory Specification") shall be taken to mean ITU-T Rec. X.511 | ISO/IEC 9594-3. The term "Directory Specifications" shall be taken to mean the X.500-series Recommendations and all parts of ISO/IEC 9594.

This Directory Specification uses the term *first 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 *second edition systems* to refer to systems conforming to the second 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 *third 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 *fourth edition systems* to refer to systems conforming to the fourth edition of the Directory Specifications, i.e., the 2001 editions of ITU-T Recs 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 Rec. X.509, and parts 1-10 of the ISO/IEC 9594:2001 edition.

This Directory Specification uses the term *fifth edition systems* to refer to systems conforming to the fifth edition of the Directory Specifications, i.e., the 2005 editions of the series of ITU-T X.500 Recommendations and the ISO/IEC 9594:2005 edition.

This Directory Specification uses the term *sixth edition systems* to refer to systems conforming to the sixth edition of the Directory Specifications, i.e., the 2008 editions of the series of ITU-T X.500 Recommendations and the ISO/IEC 9594:2008 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.

If the items in a list are numbered (as opposed to using "–" or letters), then the items shall be considered steps in a procedure.

6 Overview of the Directory service

As described in ITU-T Rec. X.501 | ISO/IEC 9594-2, the services of the Directory are provided through access points to DUAs, each acting on behalf of a user. These concepts are depicted in Figure 1. Through an access point, the Directory provides service to its users by means of a number of Directory operations.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

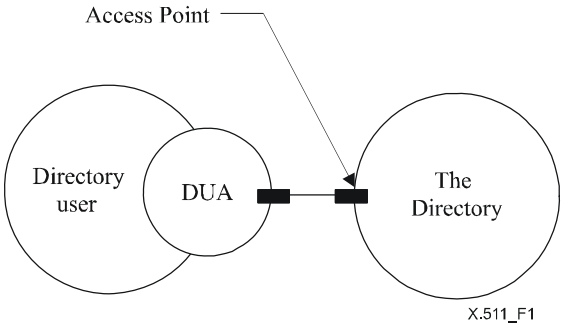

Figure 1 – Access to the Directory

The Directory operations are of three different kinds:

a) Directory Read operations, which interrogate a single Directory entry;

b) Directory Search operations, which interrogate potentially several Directory entries; and

c) Directory Modify operations.

The Directory Read operations, the Directory Search operations and the Directory Modify operations are specified in clauses 9, 10, and 11, respectively. Conformance to Directory operations is specified in ITU-T Rec. X.519 | ISO/IEC 9594-5.

7 Information types and common procedures

7.1 Introduction

This clause identifies, and in some cases defines, a number of information types which are subsequently used in the definition of Directory operations. The information types concerned are those which are common to more than one operation, are likely to be in the future, or which are sufficiently complex or self-contained as to merit being defined separately from the operation which uses them.

Several of the information types used in the definition of the Directory Service are actually defined elsewhere. Subclause 7.2 identifies these types and indicates the source of their definition. Each of the subclauses (7.3 through 7.10) identifies and defines an information type.

This clause also specifies some common elements of procedure that apply to most or all of the Directory operations.

7.2 Information types defined elsewhere

The following information types are defined in ITU-T Rec. X.501 | ISO/IEC 9594-2:

a) **Attribute**;

b) **AttributeType**;

c) **AttributeValue**;

d) **AttributeValueAssertion**;

e) **Context**;

f) **ContextAssertion**;

g) **DistinguishedName**;

h) **Name**;

i) **OPTIONALLY-PROTECTED**;

j) **OPTIONALLY-PROTECTED-SEQ**;

k) **RelativeDistinguishedName**.

The following information type is defined in ITU-T Rec. X.520 | ISO/IEC 9594-6:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

a) **PresentationAddress**.

The following information type is defined in ITU-T Rec. X.509 | ISO/IEC 9594-8:

a) **Certificate**;

b) **SIGNED**;

c) **CertificationPath**.

The following information type is defined in ITU-T Rec. X.880 | ISO/IEC 13712-1:

a) **InvokeId**.

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

a) **OperationProgress**;

b) **ContinuationReference**.

7.3 Common arguments

The **CommonArguments** information may be present to qualify the invocation of each operation that the Directory can perform.

```
CommonArguments ::= SET {
    serviceControls              [30]  ServiceControls              DEFAULT { },
    securityParameters      [29]  SecurityParameters   OPTIONAL,
    requestor                    [28]  DistinguishedName    OPTIONAL,
    operationProgress       [27]  OperationProgress
                                       DEFAULT { nameResolutionPhase notStarted },
    aliasedRDNs             [26]  INTEGER                  OPTIONAL,
    criticalExtensions      [25]  BIT STRING               OPTIONAL,
    referenceType           [24]  ReferenceType            OPTIONAL,
    entryOnly                    [23]  BOOLEAN                  DEFAULT TRUE,
    exclusions              [22]  Exclusions           OPTIONAL,
    nameResolveOnMaster     [21]  BOOLEAN                  DEFAULT FALSE,
    operationContexts       [20]  ContextSelection     OPTIONAL,
    familyGrouping               [19]  FamilyGrouping           DEFAULT entryOnly }
```

The **ServiceControls** component is specified in 7.5. Its absence is deemed equivalent to there being an empty set of controls.

The **SecurityParameters** component is specified in 7.10. If the argument of the operation is to be signed by the requestor, the **SecurityParameters** component shall be included in the argument. The absence of the **SecurityParameters** component is deemed equivalent to an empty set.

The **requestor** Distinguished Name identifies the originator of a particular operation. It holds the name of the user as identified at the time of binding to the Directory. It may be required when the request is to be signed (see 7.10), and shall hold the name of the user who initiated the request.

NOTE 1 – Where a user has alternative distinguished names differentiated by context, the name used as the value of **requestor** shall be the primary distinguished name where known. Otherwise, authentication and access control based on the value of **requestor** may not work as desired.

The **operationProgress**, **referenceType**, **entryOnly**, **exclusions** and **nameResolveOnMaster** components are defined in ITU-T Rec. X.518 | ISO/IEC 9594-4. They are supplied by a DUA either:

a) when acting on a continuation reference returned by a DSA in response to an earlier operation, and their values are copied by the DUA from the continuation reference; or

b) when the DUA represents an administrative user that is managing the DSA Information Tree and the **manageDSAIT** option is set in the service controls.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The **aliasedRDNs** component indicates to the DSA that the **object** component of the operation was created by the dereferencing of an alias on an earlier operation attempt. The integer value indicates the number of RDNs in the name that came from dereferencing the alias. (The value would have been set in the referral response of the previous operation.)

NOTE 2 – This component is provided for compatibility with first edition implementations of the Directory. DUAs (and DSAs) implemented according to later editions of the Directory Specifications shall always omit this parameter from the **CommonArguments** of a subsequent request. In this way, the Directory will not signal an error if aliases dereference to further aliases.

The **operationContexts** component supplies a set of context assertions which are applied to attribute value assertions and entry information selection made within this operation, which do not otherwise contain context assertions for the same attribute type and context type. If **operationContexts** is not present or does not address a particular attribute type or context type, then default context assertions shall be applied by the DSA as described in 7.6.1 and in 8.9.2.2 and 12.8 of ITU-T Rec. X.501 | ISO/IEC 9594-2. If **allContexts** is chosen, then all contexts for all attribute types are valid and context defaults that might have been supplied by the DSA are overridden. (**ContextSelection** is defined in 7.6).

familyGrouping is used to describe which family members should be selected for processing by a given operation. It is described more fully in 7.3.2.

7.3.1 Critical extensions

The **criticalExtensions** component provides a mechanism to list a set of extensions that are critical to the performance of a Directory operation. If the originator of the extended operation wishes to indicate that the operation shall be performed with one or more extensions (i.e., that performing the operation without these extensions is not acceptable), it does so by setting the **criticalExtensions** bit(s) which corresponds to the extension(s). If the Directory, or some part of it, is unable to perform a critical extension, it returns an indication of **unavailableCriticalExtension** (as a **serviceError** or **PartialOutcomeQualifier**). If the Directory is unable to perform an extension that is not critical, it ignores the presence of the extension.

This Directory Specification does not establish rules regarding the order in which a performing DSA is to decode and process PDUs that it receives. A DSA that receives an unknown critical extension shall return a **ServiceError** with problem **unavailableCriticalExtension** to signal that the operation failed.

These Directory Specifications define a number of extensions. The extensions take such forms as additional numbered bits in a BIT STRING, or additional components of a SET or SEQUENCE, and are ignored by first edition systems. Each such extension is assigned an integer identifier, which is the number of the bit that may be set in **criticalExtensions**. If the criticality of an extension is defined to be critical, the DUA shall set the corresponding bit in **criticalExtensions**. If the defined criticality is non-critical, the DUA may or may not set the corresponding bit in **criticalExtensions**.

The extensions, their identifiers, the operations in which they are permitted, the recommended criticality, the clauses in which they are defined, and the corresponding LDAP controls (if any) are shown in Table 1.

Table 1 – Extensions

Extension	Identifier	Operations	Criticality	Defined (subclauses)	LDAP Control
subentries	1	All	Non-critical	7.5	1.3.6.1.4.1.4203.1.10.1
copyShallDo	2	Read, Compare, List, Search	Non-critical	7.5	
attribute size limit	3	Read, Search	Non-critical	7.5	
extraAttributes	4	Read, Search	Non-critical	7.6	
modifyRightsRequest	5	Read	Non-critical	9.1	
pagedResultsRequest	6	List, Search	Non-critical	10.1	1.2.840.113556.1.4.319
matchedValuesOnly	7	Search	Non-critical	10.2	1.2.826.0.1.3344810.2.3
extendedFilter	8	Search	Non-critical	10.2	
targetSystem	9	Add Entry	Critical	11.1	
useAliasOnUpdate	10	Add Entry, Remove Entry, Modify Entry	Critical	11.1	
newSuperior	11	Modify DN	Critical	11.4	
manageDSAIT	12	All	Critical	7.5, 7.13	2.16.840.1.113730.3.4.2

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 1 – Extensions

Extension	Identifier	Operations	Criticality	Defined (subclauses)	LDAP Control
useContexts	13	Read, Compare, List, Search, Add Entry, Modify Entry, Modify DN	Non-critical	7.6, 7.8	
partialNameResolution	14	Read, Search	Non-critical	7.5	
overspecFilter	15	Search	Non-critical	10.1.3 f)	
selectionOnModify	16	Modify Entry	Non-critical	11.3.2	
Security parameters – Response	17	All	Non-critical	7.10	
Security parameters – Operation code	18	All	Non-critical	7.10	
Security parameters – Attribute certification path	19	All	Non-critical	7.10	
Security parameters – Error Protection	20	All	Non-critical	7.10	
SPKM Credentials	21	Directory Bind	(Note 3)	8.1.1	
Bind token – Response	22	Directory Bind	Non-critical	8.1.1	
Bind token – Bind Int. Alg, Bind Int Key, Conf Alg and Conf Key Info	23	Directory Bind	Non-critical	8.1.1	
Bind token – DIRQOP (obsolete)	24	Directory Bind	Non-critical	8.1.1	
Service administration	25	Read, Search, ModifyEntry	Critical	10.2.2, 13, clause 16 of ITU-T Rec. X.501 | ISO/IEC 9594-2	
entryCount	26	Search	Non-critical	10.1.3	
hierarchySelection	27	Search	Non-critical	7.5	
relaxation	28	Search	Non-critical	7.8	
familyGrouping	29	Compare, Search, RemoveEntry	Non-critical Non-critical Critical	7.3.2, 7.8.3 & 9.2.2 10.2 11.2.2	
familyReturn	30	Read, Search, ModifyEntry	Non-critical Non-critical Non-critical	7.6.4, 7.7.1 & 9.1.3 10.2.3 11.3.3	
dnAttributes	31	Search	Non-critical	10.2.2	
friend attributes	32	Read, Search	Non-critical	7.6, 7.8.2	
Abandon of paged results	33	List, Search	critical	7.9	
Paged results on the DSP	34	List, Search	Non-critical	7.9	
replaceValues	35	ModifyEntry	critical	11.3.1, 11.3.2	

NOTE 1 – The first extension is given the identifier 1 and corresponds to bit 1 of the BIT STRING. Bit 0 of the BIT STRING is not used.

NOTE 2 – Use of signing on errors Add Entry, Remove Entry, Modify Entry, Modify DN requires version 2 or higher of the protocol.

NOTE 3 – The SPKM credentials extension shall be critical unless used in associations established using version 2 or higher.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.2 Family grouping

Family grouping allows a single family member, several family members or all family members of a compound entry, to be grouped together for joint consideration prior to operation evaluation. These semantics can then be applied to the following operations (as indicated in the descriptions below): Compare (to define the scope within which the compared attribute might lie), Search (to define the groupings for which filtering might take place), Remove Entry (to define the groupings for removal). The following ASN.1 is used to select members of a family:

```
FamilyGrouping ::= ENUMERATED {
    entryOnly       (1),
    compoundEntry   (2),
    strands         (3),
    multiStrand     (4) }
```

entryOnly means that the specific family member selected by the operation is to be considered in the group. This is the default value, and ensures backward compatibility with previous editions of the Directory Specifications.

compoundEntry means that the complete compound entry selected by the operation is to be considered as a unit by combining all the attributes. For Remove Entry operations, it is only applicable when the object name specified is that of an ancestor of a compound entry, and it causes all family members to be removed by the same operation (subject to access control).

strands means that all the strands associated with the family member are to be selected by the operation. This option is not valid for the Remove Entry operation. For the Search operation, individual strands are considered for filter purposes. If the combined set of attributes of one or more strands matches the filter, the compound entry is said to match the filter. If the base object is a child member, only those strands that go through the base object are considered. For Compare operations, all the attributes from all the family members in all the strands to which the entry belongs are to be used in the comparison.

multiStrand is only applicable to the Search operation, and qualifies the matching rule for filtering on family information. It is ignored for other operations. It specifies that one strand from each family within a compound entry is to be considered at one time, but in all combinations. **multiStrand** is not applicable if the base object is a child family member, in which this case **multiStrand** shall be ignored and **entryOnly** shall be substituted.

7.4 Common results

The **CommonResults** or **CommonResultsSeq** information is present to qualify the result of each retrieval operation that the Directory can perform. In addition, it is present in any returned error.

```
CommonResults ::= SET {
    securityParameters  [30]  SecurityParameters                          OPTIONAL,
    performer           [29]  DistinguishedName                           OPTIONAL,
    aliasDereferenced   [28]  BOOLEAN                                     DEFAULT FALSE,
    notification        [27]  SEQUENCE SIZE (1..MAX) OF Attribute{{SupportedAttributes}}  OPTIONAL }

CommonResultsSeq ::= SEQUENCE {
    securityParameters  [30]  SecurityParameters                          OPTIONAL,
    performer           [29]  DistinguishedName                           OPTIONAL,
    aliasDereferenced   [28]  BOOLEAN                                     DEFAULT FALSE,
    notification        [27]  SEQUENCE SIZE (1..MAX) OF Attribute{{SupportedAttributes}}  OPTIONAL }
```

NOTE – **CommonResults** and **CommonResultsSeq** consist of the same components. The former is used when included in set types by the **COMPONENT OF** type, while the latter is used similarly in sequence types.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The **SecurityParameters** component is specified in 7.10. If the result is to be signed by the Directory, the **SecurityParameters** component shall be included in the result. The absence of the **SecurityParameters** component is deemed equivalent to an empty set.

The **performer** Distinguished Name identifies the performer of a particular operation. It may be required when the result is to be signed (see 7.10) and shall hold the name of the DSA that signed the result.

The **aliasDereferenced** component is set to **TRUE** when the purported name of an object or base object which is the target of the operation included any aliases which were dereferenced.

The **notification** component shall be used to qualify returned result and error APDUs, for example providing more precise error information. Standard notification attributes are defined in 6.13 of ITU-T Rec. X.520 | ISO/IEC 9594-6. Such notification attributes are not necessarily stored in directory entries.

7.5 Service controls

A **ServiceControls** parameter contains the controls, if any, that are to direct or constrain the provision of the service.

```
ServiceControls ::= SET {
        options                 [0]     ServiceControlOptions DEFAULT { },
        priority                        [1]     INTEGER { low (0), medium (1), high (2) } DEFAULT medium,
        timeLimit                       [2]     INTEGER OPTIONAL,
        sizeLimit                       [3]     INTEGER OPTIONAL,
        scopeOfReferral                 [4]     INTEGER { dmd(0), country(1) } OPTIONAL,
        attributeSizeLimit      [5]     INTEGER OPTIONAL,
        manageDSAITPlaneRef             [6]     SEQUENCE {
                dsaName                                 Name,
                agreementID                     AgreementID } OPTIONAL,
        serviceType             [7]     OBJECT IDENTIFIER OPTIONAL,
        userClass                       [8]     INTEGER OPTIONAL }

ServiceControlOptions ::= BIT STRING {
                preferChaining                  (0),
                chainingProhibited      (1),
                localScope                      (2),
                dontUseCopy             (3),
                dontDereferenceAliases  (4),
                subentries                      (5),
                copyShallDo             (6),
                partialNameResolution   (7),
                manageDSAIT                     (8),
                noSubtypeMatch                  (9),
                noSubtypeSelection      (10),
                countFamily             (11),
                dontSelectFriends       (12),
                dontMatchFriends        (13) }
```

The **options** component contains a number of indications, each of which, if set, asserts the condition suggested. Thus:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

a) **preferChaining** indicates that the preference is that chaining, rather than referrals, be used to provide the service. The Directory is not obliged to follow this preference.

b) **chainingProhibited** indicates that chaining, and other methods of distributing the request around the Directory, are prohibited.

c) **localScope** indicates that the operation is to be limited to a local scope. The definition of this option is itself a local matter, for example, within a single DSA or a single DMD.

d) **dontUseCopy** indicates that copied information as defined in ITU-T Rec. X.518 | ISO/IEC 9594-4 shall not be used to provide the service.

e) **dontDereferenceAliases** indicates that any alias used to identify the entry affected by an operation is not to be dereferenced.

NOTE 1 – This is necessary to allow reference to an alias entry itself rather than the aliased entry, e.g., in order to read the alias entry.

f) **subentries** indicates that a Search or List operation is to access subentries only; normal entries become inaccessible, i.e., the Directory behaves as though normal entries do not exist. If this service control is not set, then the operation accesses normal entries only and subentries become inaccessible. The service control is ignored for operations other than Search or List.

NOTE 2 – The effects of subentries on access control, schema, and collective attributes are still observed even if subentries are inaccessible.

NOTE 3 – If this service control is set, normal entries may still be specified as the base object of an operation.

g) **copyShallDo** indicates that if the Directory is able to partly but not fully satisfy a query at a copy of an entry, it shall not chain the query. It is meaningful only if **dontUseCopy** is not set. If **copyShallDo** is not set, the Directory will use shadow data only if it is sufficiently complete to allow the operation to be fully satisfied at the copy. A query may be only partly satisfied because some of the requested attributes are missing in the shadow copy, some of the attribute values for a given attribute are missing in the shadow copy, because the DSA does not hold all context information for the attribute values it does have, or because the DSA holding the shadowed data does not support the requested matching rules on that data. If **copyShallDo** is set and the Directory is not able to fully satisfy a query, it shall set **incompleteEntry** in the the returned entry information.

h) **partialNameResolution** indicates that if the Directory is able to resolve only part of the purported name in a Read or Search operation, i.e., it is about to return a **nameError**, the entry whose name consists of all resolved RDNs is to be considered the target of the operation and **partialName** is set to **TRUE** in the result. This service control is ignored for operations other than Read or Search.

NOTE 4 – If this service control is set, the purported name is a context prefix entry to which access is denied, and the requestor has access to the superior entry, then the existence of the context prefix entry will be indirectly disclosed to the requestor even if *DiscloseOnError* permission to the entry is denied.

i) **manageDSAIT** indicates that the operation has been requested by an administrative user so that the DSA Information Tree is managed. If multiple replications planes exist in the DSA to be managed, and the **manageDSAITPlaneRef** service control has not been included in the operation, then the DSA selects a suitable replication plane for the operation.

j) **noSubtypeMatch** indicates that attribute subtype matching shall not be attempted. This service control is ignored for operations other than Compare and Search operations.

k) **noSubtypeSelection** indicates that subtype selection shall not be made.

l) **countFamily** indicates that each member of a compound entry shall be counted as a separate entry, e.g., for the purposes of size and administrative limits, and relaxation controls. If this control is not set, then members of a compound attribute shall be counted as a single entry.

m) **dontSelectFriends** indicates that the specification of an anchor attribute in entry information selection does not automatically include friend attributes in the selection.

n) **dontMatchFriends** indicates that the specification of an anchor attribute in a filter item can only be satisfied by the values of the anchor attribute, and not by friend attributes.

o) **allowWriteableCopy** indicates that a DSE of type **writeableCopy** is acceptable in the provision of a query service request.

NOTE 5 – The **allowWriteableCopy** service control is distinct from **copyShallDo** in that this service control is used to indicate that a complete copy is requested, but that it does not need to be the primary master, whereas **copyShallDo** is used to indicate that any copy, whether complete or not, is acceptable.

If this component is omitted, the following are assumed: no preference for chaining but chaining not prohibited, no limit on the scope of the operation, use of copy permitted, aliases shall be dereferenced (except for modify operations for

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

which alias dereferencing is not supported), subentries are not accessible, and operations that cannot be fully satisfied by shadowed data are subject to further chaining. However, these defaults may be overwritten by search-rules within service specific administrative areas.

The **priority** (**low**, **medium**, or **high**) at which the service is to be provided. Note that this is not a guaranteed service in that the Directory, as a whole, does not implement queuing. There is no relationship implied with the use of priorities in underlying layers.

The **timeLimit** indicates the maximum elapsed time, in seconds, within which the service shall be provided. If the constraint cannot be met, an error is reported. If this component is omitted, no time limit is implied. In the case of time limit exceeded on a List or Search, the result is an arbitrary selection of the accumulated results.

NOTE 6 – This component does not imply the length of time spent processing the request during the elapsed time: any number of DSAs may be involved in processing the request during the elapsed time.

The **sizeLimit** is only applicable to List and Search operations. It indicates the maximum number of entries to be returned when paged results are not to be returned. In the case of size limit exceeded, the results of a List or Search operation may be an arbitrary selection of the accumulated results, equal in number to the size limit. Any further results shall be discarded. When paged results are being returned, the value of **sizeLimit** shall be ignored by the DSA performing the paging as detailed in 7.9.

The **scopeOfReferral** indicates the scope to which a referral returned by a DSA should be relevant. Depending on whether the values **dmd** or **country** are selected, only referrals to other DSAs within the selected scope shall be returned. This applies to the referrals in both a **referral** error and the **unexplored** parameter of **list** and **search** results.

The **attributeSizeLimit** indicates the largest size of any attribute (i.e., the type and all its values) that is included in returned entry information. If an attribute exceeds this limit, all of its values are omitted from the returned entry information and **incompleteEntry** is set in the returned entry information. The size of an attribute is taken to be its size in octets in the local concrete syntax of the DSA holding the data. Because of different ways applications store the data, the limit is imprecise. If this parameter is not specified, no limit is implied.

NOTE 7 – Attribute values returned as part of an entry's Distinguished Name are exempt from this limit.

Certain combinations of **priority**, **timeLimit**, and **sizeLimit** may result in conflicts. For example, a short time limit could conflict with low priority; a high size limit could conflict with a low time limit, etc.

The **manageDSAITPlaneRef** indicates that the operation has been requested by an administrative user so that a specific replication plane of the DSA Information Tree is managed. The **manageDSAITPlaneRef** service control is ignored if the **manageDSAIT** option is not set. The plane is identified by the **dsaName** component which is the name of the supplying DSA and the **agreementID** component which contains the shadowing agreement identifier.

The **serviceType** service control is only relevant for a **search** request that starts its initial evaluation phase within a service specific administrative area; it is otherwise ignored. If supplied, it increases the possibility of getting useful notification information returned in case of a faulty formulated **search** request.

The **userClass** service control is only relevant for a **search** request that starts its initial evaluation phase within a service specific administrative area and is otherwise ignored. It identifies a user-class. It allows a requestor to specify another user-class than the Directory would otherwise apply. If supplied, it also increases the possibility of getting useful notification information returned in case of a faulty formulated search request.

7.6 Entry information selection

An **EntryInformationSelection** parameter indicates what information is being requested from an entry in a retrieval service.

```
EntryInformationSelection ::= SET {
    attributes                  CHOICE {
        allUserAttributes       [0]   NULL,
        select                  [1]   SET OF AttributeType
        -- empty set implies no attributes are requested -- } DEFAULT allUserAttributes : NULL,
    infoTypes                   [2]   INTEGER {
        attributeTypesOnly              (0),
        attributeTypesAndValues         (1) } DEFAULT attributeTypesAndValues,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        extraAttributes                 CHOICE {
            allOperationalAttributes    [3]     NULL,
            select                      [4]     SET SIZE (1..MAX) OF AttributeType } OPTIONAL,
        contextSelection            ContextSelection OPTIONAL,
        returnContexts                  BOOLEAN DEFAULT FALSE,
        familyReturn                FamilyReturn  DEFAULT
                                            { memberSelect  contributingEntriesOnly } }

ContextSelection  ::=  CHOICE  {
        allContexts                 NULL,
        selectedContexts        SET SIZE (1..MAX) OF TypeAndContextAssertion }

TypeAndContextAssertion  ::=  SEQUENCE  {
        type                        AttributeType,
        contextAssertions       CHOICE {
            preference              SEQUENCE OF ContextAssertion,
            all                         SET OF ContextAssertion } }

FamilyReturn ::= SEQUENCE {
        memberSelect ENUMERATED {
            contributingEntriesOnly     (1),
            participatingEntriesOnly    (2),
            compoundEntry                   (3) },
        familySelect    SEQUENCE SIZE (1..MAX) OF OBJECT-CLASS.&id   OPTIONAL }
```

The **attributes** component specifies the user and operational attributes about which information is requested:

a) If the **select** option is chosen, then the attributes involved are listed. If the list is empty, then no attributes shall be returned. Information about a selected attribute shall be returned if the attribute is present. An **attributeError** with problem **noSuchAttributeOrValue** shall only be returned if none of the attributes selected is present.

b) If the **allUserAttributes** option is selected, then information is requested about all user attributes in the entry.

Attribute information is only returned if access rights are sufficient. A **securityError** (with problem **insufficientAccessRights**) shall only be returned in the case where access rights preclude the reading of all attribute values requested. Note that access control is also applied to the attributes and values eligible to be returned according to the components of **EntryInformationSelection**, and may further reduce the information that is returned.

NOTE 1 – Access control is also applied to the attributes and values eligible to be returned according to the components of **EntryInformationSelection**, and may further reduce the information that is returned.

The **infoTypes** component specifies whether both attribute type and attribute value information (the default) or attribute type information only is requested. If an attribute is of a type that is a carrier of other attributes, e.g., a **family-information** attribute, then the value(s) shall be returned independent of the setting of the **infoTypes** component, but the **infoTypes** specification shall be applied to the contained attributes. If the **attributes** component is such as to request no attributes, then this component is not meaningful.

The **extraAttributes** component specifies a set of additional user and operational attributes about which information is requested. If the **allOperationalAttributes** option is chosen, then information is requested about all directory operational attributes in the entry. If the **select** option is chosen, then information about the listed attributes is requested.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

NOTE 2 – This component may be used to request information about, for example, specific operational attributes when **attributes** is set to **allUserAttributes**, or about all operational attributes. If the same attribute is listed or implied in both **attributes** and **extraAttributes**, it is treated as though it has been requested only once.

A request for a particular attribute is always treated as a request for the attribute and all *subtypes* of that attribute (except for requests processed by first-edition systems) if the **noSubtypeSelection** service control option is not set. If the **noSubtypeSelection** service control option is set, only the requested attributes are returned, not their subtypes. Similarly, a request for a particular attribute that has friends is treated as a request for the attribute and all friend attributes, subject to the **dontSelectFriends** service control option not being set.

In responding to a request for attribute information, the Directory treats all *collective attributes* of an entry as if they were actual user attributes of the entry, i.e., they are selected like other user attributes and are merged into the returned entry information. A request for **allUserAttributes** requests all collective attributes of the entry as well as ordinary attributes of the entry. An attribute is a collective attribute of an entry if all of the following are true:

a) it is located in a subentry whose subtree specification includes the entry;

b) it is not excluded by the presence in the entry of a **collectiveExclusions** attribute value equal to the collective attribute type; and

c) it is permitted by the content rule for the structural object class for the entry.

The **contextSelection** component is used to specify which attribute values shall be returned of the attributes selected by **attributes** or **extraAttributes**. The **contextSelection** is evaluated only against the values of attributes that are candidates to be returned according to those other components of **EntryInformationSelection**. The evaluation of **contextSelection**, and the use of defaults if it is not supplied, is discussed in 7.6.1 to 7.6.3.

If the **infoTypes** component is such as to request no attribute values, or the **attributes** component is such as to request no attributes, then the **contextSelection** component is not meaningful. If, as a result of applying **contextSelection**, there are no values of an attribute eligible to be returned, the attribute may be returned without any values.

The **returnContexts** component is used to request the Directory to return attribute values with their associated context lists. If this component is absent or is specified with a value of **FALSE**, then no context information is returned in the result. If this component is specified with a value of **TRUE,** then all context information is returned for each attribute value returned. Note that the **contextSelection** component does not selectively affect which context information is returned when **returnContexts** is **TRUE**.

The **familyReturn** component (if present) is used to determine which entries within a compound entry shall be returned if one or more family members have been marked (see 7.6.4).

7.6.1 Use of contextSelection or context selection defaults

The **contextSelection** component is used to select certain attribute values of attributes selected by **attributes** or **extraAttributes**. The **contextSelection** is evaluated only against the values of attributes that are candidates to be returned according to those other components of **EntryInformationSelection**. For each attribute value, any context selection governing its attribute type shall evaluate to TRUE (as defined in 7.6.2), in order for that attribute value to be selected.

A **contextSelection** is said to govern an attribute type if any of the following conditions occur:

– the **ContextSelection** specifies **allContexts** (in which case all attribute values of all attribute types are selected);

– the **ContextSelection** has a **selectedContexts** which includes a **TypeAndContextAssertion** whose type is the same as or a supertype of the attribute type; or

– the **ContextSelection** has a **selectedContexts** which includes a **TypeAndContextAssertion** whose type is **id-oa-allAttributeTypes**.

If **contextSelection** is not provided or it does not govern the given attribute type, then a default **contextSelection** shall be applied. In addition to **contextSelection** in **EntryInformationSelection**, there are three potential sources for a **contextSelection**: 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 according to the following precedence:

1) If **contextSelection** is present in **EntryInformationSelection** and it governs the given attribute type as described above, then it shall be applied.

2) If **contextSelection** is not present within the **EntryInformationSelection**, or it is present but does not govern the given attribute type, then the **operationContexts** which has been supplied for the operation as described in 7.3 shall be applied if one is present and it governs the given attribute type as described above.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

3) If the request has neither a **contextSelection** in the **EntryInformationSelection** nor **operationContexts** for the operation, or neither governs the given attribute, then the values of the **contextAssertionDefaults** attribute in the context assertion subentries (if any) controlling the entry shall be applied as the **selectedContexts**. (Context assertion subentries are described in 14.7 of ITU-T Rec. X.501 | ISO/IEC 9594-2).

4) If there is no **contextSelection** from the sources described above that govern the given attribute type, then the DSA may apply a locally-defined default **contextSelection**. 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.

5) If no **contextSelection** is available from any of these sources that govern the given attribute type, then all values of the attribute are considered selected (i.e., **allContexts** is assumed as the base default).

NOTE – A default **contextSelection** that governs the given attribute type and makes an assertion about a certain context type shall be applied in addition to an earlier **contextSelection** governing the same attribute type but making an assertion about a different context type, in the same order of precedence as described above.

7.6.2 Evaluation of contextSelection

A **contextSelection** is TRUE (i.e., selects a given attribute value) if:

a) **allContexts** is specified (this permits a context selection to override any default that might otherwise be applied if this **contextSelection** were omitted); or

b) each **TypeAndContextAssertion** in **selectedContexts** is TRUE as described in 7.6.3.

A **contextSelection** is FALSE otherwise.

7.6.3 Evaluation of a TypeAndContextAssertion

A **TypeAndContextAssertion** is TRUE (i.e., selects a given attribute value) if:

a) the type of the attribute is not the same as (nor a subtype of) the **type** in the **TypeAndContextAssertion** and the **type** in the **TypeAndContextAssertion** is not **id-oa-allAttributeTypes**. In this case, the **TypeAndContextAssertion** is not applicable to the attribute type of the given attribute value and so does not eliminate the attribute value from selection; or

b) for the attribute value, the **contextAssertions** in **TypeAndContextAssertion** is TRUE as defined below.

NOTE 1 – The **OBJECT IDENTIFIER** value **id-oa-allAttributeTypes** may be used as the value of **type** in the **TypeAndContextAssertion** to force evaluation of the **contextAssertions** against an attribute value of any attribute type.

contextAssertions is expressed either as an ordered sequence of preferred contexts or as a compound set of context assertions:

a) If **all** is specified, then **contextAssertions** is TRUE for any attribute value only if each **ContextAssertion** in the SET is TRUE as defined in 8.9.2.4 of ITU-T Rec. X.501 | ISO/IEC 9594-2.

b) If **preference** is specified, then each **ContextAssertion** in the **SEQUENCE** is evaluated in turn against all candidate attribute values of the same attribute type, until a **ContextAssertion** evaluates TRUE as defined in 8.9.2.4 of ITU-T Rec. X.501 | ISO/IEC 9594-2. (The **fallback** flag, if present, is not taken into consideration until the entire SEQUENCE is exhausted.) Once a **ContextAssertion** evaluates TRUE for one of the candidate attribute values, it shall be evaluated for every candidate attribute value of the same attribute type, but subsequent **ContextAssertion** in the SEQUENCE are ignored.

NOTE 2 – **preference** provides a means for selection to be specified in terms of a first, second, etc., choice of context (e.g., Language = French but if no French then Language = English).

A **TypeAndContextAssertion** is FALSE otherwise.

7.6.4 Family Return

The **familyReturn** component is used to determine which entries within a compound entry shall be returned if one or more family members have been marked as contributing or participating members. The procedures for how family members are marked are further described in 7.13.

The **memberSelect** component specifies what entries are selected for return in the result:

– **contributingEntriesOnly** means that only family members marked as contributing members by the operation are to be returned. In the case of Read or Modify-Entry operations, this is the family member identified by the **object** operation argument; for the Search operation, it includes family members that contributed to the match.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

– **participatingEntriesOnly** means that only family members marked as participating members by the operation are to be returned. In the case of Read or Modify-Entry, this is the same as for **contributingEntriesOnly**.

– **compoundEntry** means that each family member within the compound entry is to be returned, except those that possible have been explicitly un-marked by a governing-search-rule for a Search operation.

The **familySelect** component supplements the **memberSelect** component by specifying that all child members of selected families shall be returned in addition to what is specified by **memberSelect**. The sequence of elements has no significance. A family is identified by the structural object class of the family members immediate subordinate to the ancestor. This component has no effect if the **memberSelect** specifies **compoundEntry**.

NOTE – A governing-search-rule may modify what information shall be returned (see 16.10 of ITU-T Rec. X.501 | ISO/IEC 9594-2).

7.7 Entry information

7.7.1 Entry information data type

The **EntryInformation** data type conveys selected information from an entry.

```
EntryInformation ::= SEQUENCE {
    name                    Name,
    fromEntry               BOOLEAN DEFAULT TRUE,
    information             SET SIZE (1..MAX) OF CHOICE {
        attributeType           AttributeType,
        attribute               Attribute{{SupportedAttributes}} } OPTIONAL,
    incompleteEntry     [3] BOOLEAN DEFAULT FALSE, -- not in first edition systems
    partialName         [4] BOOLEAN DEFAULT FALSE, -- not in first or second edition systems
    derivedEntry        [5] BOOLEAN DEFAULT FALSE -- not in pre-fourth edition systems -- }
```

The **Name** parameter indicates the Distinguished Name of the entry or the name of an alias to the entry. The Distinguished Name of the entry is returned whenever permitted by the access control policy. If access is allowed to the attributes of the entry but not to its Distinguished Name, the Directory may return either an error or the name of a valid alias to the entry.

The primary distinguished name is used for the **Name** parameter. This means that if an RDN forming the name includes an attribute which has multiple distinguished values differentiated by context, then the primary distinguished value is used as the **value** in the returned RDN's **AttributeTypeAndDistinguishedValue** for that attribute. Since for each RDN the returned **value** is thus always the primary distinguished value, **primaryDistinguished** shall be omitted for all **AttributeTypeAndDistinguishedValue**.

The RDNs in **Name** shall include alternative distinguished values only if a context selection has been applied to the entry information being returned. The alternative distinguished values are returned as part of **valuesWithContext** in the returned RDN's **AttributeTypeAndDistinguishedValue.** The context selections applied to the entry information being returned (see 7.6.1) are also applied to the alternative distinguished values to determine which distinguished values to use in **valuesWithContext**.

NOTE 1 – The context selection is not applied to the primary distinguished values returned in **Name**.

If a request has been made to return context information with the result, then context information shall also be included where available for the distinguished value within **Name** (using the **valuesWithContext** element of the RDNs). When alternative distinguished values are being returned, context information is always returned, for all distinguished values.

NOTE 2 – If the entry was located using an alias, then that alias is known to be a valid alias. Otherwise, how it is ensured that the alias is valid is outside the scope of these Directory Specifications.

NOTE 3 – Where a particular component of the Directory has a choice of alias names available to it for return, it is recommended that where possible it choose the same alias name for repeated requests by the same requestor, in order to provide a consistent service.

The **fromEntry** parameter indicates whether the information was obtained from the entry (**TRUE**) or a copy of the entry (**FALSE**).

The **information** parameter is included if any attribute information from the entry is being returned, and contains a set of **attributeTypes** and **attributes**, as appropriate.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The **incompleteEntry** parameter is included and set to **TRUE** whenever the returned entry information is incomplete in relation to the user's request, e.g., because attributes or attribute values are omitted for reasons of access control (and their existence is permitted to be disclosed), the presence of incomplete shadow information together with **copyShallDo**, or because the **attributeSizeLimit** has been exceeded. It is not set to **TRUE** because an alias name has been returned instead of the Distinguished Name.

The Directory shall complete the name resolution phase of operations in its entirety (including checking of all relevant knowledge references, following up on referrals, etc.) before the **partialNameResolution** service control is considered. If all name resolution options have been exhausted and at least one RDN has been resolved, the **partialName** parameter is included and set to **TRUE** if the request had the **partialNameResolution** service control set and the Directory was unable to complete name resolution on all RDNs of the relevant entry. When **partialName** is returned as **TRUE**, it indicates that the information being returned is from the entry at the point where the last RDN was successfully resolved.

The **derivedEntry** parameter is included and set to **TRUE** whenever the returned entry information contains joined results obtained by performing a join on data that originated from more than one directory entry. When this parameter is **TRUE**, the value in **name** may be the name of any of the related entries from which the entry information is derived, or it may be the name of an alias to any of those entries. The value in **name** should not be used in subsequent operations. If the **derivedEntry** parameter is set to **TRUE** and the response is signed, the signature is that of the DSA performing the join.

7.7.2 Family information in entry information

When information from a compound entry is to be returned, attributes from each member to be returned are selected according to the **EntryInformationSelection** (possibly modified by a governing-search-rule). When the **separateFamilyMembers** search control option is set in the **search** request each member is returned as a separate entry. Otherwise, if more than one member is to be returned, the entry information shall be packed in such way that the information appears to come from a single entry, which can be the ancestor or a subordinate member (the latter is appropriate when the base object of the **search** request is a family member subordinate to the ancestor and the ancestor has not been selected by **FamilyReturn**). The attributes from the other members shall be packed into a **family-information** derived attribute as described below.

NOTE 1 – According to above, multiple family members are always packed in a **read** or **modifyEntry** result.

The use of the **family-information** derived attribute is for packaging only; the attribute does not exist as a distinct entity; it cannot directly be selected by **entryInformationSelection** (any attempt to do so shall be ignored), nor can it be protected directly by access control.

```
family-information  ATTRIBUTE  ::= {
        WITH SYNTAX             FamilyEntries
        USAGE                   directoryOperation
        ID                      id-at-family-information }

FamilyEntries  ::=  SEQUENCE  {
        family-class            OBJECT-CLASS.&id,       -- structural object class value
        familyEntries           SEQUENCE OF FamilyEntry }

FamilyEntry  ::=  SEQUENCE  {
        rdn                     RelativeDistinguishedName,
        information             SEQUENCE OF CHOICE  {
                attributeType       AttributeType,
                attribute               Attribute{{SupportedAttributes}} },
        family-info             SEQUENCE SIZE (1..MAX) OF FamilyEntries OPTIONAL }
```

The **family-information** attribute is a multi-valued attribute. If the ancestor is designated as the source of information, each attribute value holds information from a single family. If a family member subordinate to the ancestor is

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

designated as the source of information, information is sorted into attribute values based on the structural object classes of the immediately subordinate members of the designated member.

Each family member that is selected is represented by a value of type **FamilyEntry**, which contains:

– Selected attribute information (where appropriate), either as an attribute type or as a complete attribute, depending on the **infoTypes** value in **EntryInformationSelection**;

NOTE 2 – As stated in 7.6, the **infoTypes** specification only applies for the contained attributes, not for the **family-information** attribute itself.

– Any nested **FamilyEntries** information in the form of a complete **family-information** attribute, collected in terms of the structural object classes of the subordinate entries;

– Unselected entries are not represented at all unless they are superior to one or more family members that have been selected.

7.8 Filter

7.8.1 Filter parameter

A **Filter** parameter applies a test that is either satisfied or not by a particular entry. The filter is expressed in terms of assertions about the presence or value of certain attributes of the entry, and is satisfied if and only if it evaluates to TRUE.

NOTE – A filter may be TRUE, FALSE, or UNDEFINED.

```
Filter ::= CHOICE {
        item    [0]     FilterItem,
        and     [1]     SET OF Filter,
        or      [2]     SET OF Filter,
        not     [3]     Filter }

FilterItem      ::=     CHOICE {
        equality                        [0]     AttributeValueAssertion,
        substrings                      [1]     SEQUENCE {
                type                                    ATTRIBUTE.&id ({ SupportedAttributes }),
                strings                         SEQUENCE OF CHOICE {
                        initial                 [0]     ATTRIBUTE.&Type
                                                        ({SupportedAttributes}{@substrings.type}),
                        any                             [1]     ATTRIBUTE.&Type
                                                        ({SupportedAttributes}{@substrings.type}),
                        final                           [2]     ATTRIBUTE.&Type
                                                        ({SupportedAttributes}{@substrings.type}),
                        control                         Attribute{SupportedAttributes}} } },     -- Used to specify
interpretation of following
                                                                                        -- items
        greaterOrEqual          [2]     AttributeValueAssertion,
        lessOrEqual     [3]     AttributeValueAssertion,
        present                 [4]     AttributeType,
        approximateMatch        [5]     AttributeValueAssertion,
        extensibleMatch         [6]     MatchingRuleAssertion,
        contextPresent          [7]     AttributeTypeAssertion }
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
MatchingRuleAssertion ::= SEQUENCE {
    matchingRule [1]    SET SIZE (1..MAX) OF MATCHING-RULE.&id,
    type         [2]    AttributeType OPTIONAL,
    matchValue   [3]    MATCHING-RULE.&AssertionType ( CONSTRAINED BY {
                        -- matchValue shall be a value of type specified by the &AssertionType field of
                        -- one of the MATCHING-RULE information objects identified by matchingRule -- } ),
    dnAttributes [4]    BOOLEAN DEFAULT FALSE }
```

A **Filter** is either a **FilterItem** (see 7.8.2), or an expression involving simpler filters composed together with the logical operators **and**, **or**, and **not**. The evaluation of a filter can be affected by the action of a relaxation policy, which can cause a substitution of one matching rule for another, or can supply values that are to be considered for matching.

A **Filter** which is a **FilterItem** has the value of the **FilterItem** (i.e., TRUE, FALSE, or UNDEFINED).

A **Filter** which is the **and** of a set of filters is TRUE if the set is empty or if each filter is TRUE; it is FALSE if at least one filter is FALSE; otherwise it is UNDEFINED (i.e., if at least one filter is UNDEFINED and no filters are FALSE).

A **Filter** which is the **or** of a set of filters is FALSE if the set is empty or if each filter is FALSE; it is TRUE if at least one filter is TRUE; otherwise it is UNDEFINED (i.e., if at least one filter is UNDEFINED and no filters are TRUE).

A **Filter** which is the **not** of a filter is TRUE if the filter is FALSE; FALSE if it is TRUE; and UNDEFINED if it is UNDEFINED.

A *non-negated* filter item is defined as one that is nested within an even number of **not** elements (possibly zero) within the outermost **Filter**. Thus, a filter comprising only filter items in an **and** or **or** combination would only contain non-negated items. A *negated* filter item is defined as one nested within an odd number of **not** elements within the outermost **Filter**.

7.8.2 Filter item

A **FilterItem** is an assertion about the presence or value(s) of attributes in the entry under test. An assertion about a particular attribute type is also satisfied if the entry contains a subtype of the attribute and the assertion is TRUE for the subtype and the **noSubtypeMatch** service control option is not set, or if there is a collective attribute of the entry (see 7.6) for which the assertion is TRUE, or if

- the **dontMatchFriends** service control option is not set; and
- the entry contains a friend attribute for the specified attribute which has a matching rule compatible with the assertion; and
- the assertion is TRUE for the friend attribute.

Each assertion is TRUE, FALSE, or UNDEFINED.

Every **FilterItem** includes or implies one or more **AttributeTypes** which identifies the particular attribute(s) concerned.

Any assertion about the values of such an attribute is only defined if the **AttributeType** is known by the evaluating mechanism, the purported **AttributeValue**(s) conforms to the attribute syntax defined for that attribute type, the implied or indicated matching rule is applicable to that attribute type, and (when used) a presented **matchValue** conforms to the syntax defined for the indicated matching rules. When these conditions are not met, the **FilterItem** shall evaluate to the logical value UNDEFINED.

NOTE 1 – Access control restrictions may affect the evaluation of the **FilterItem** and may cause the **FilterItem** to evaluate to UNDEFINED.

An assertion which is defined by these conditions additionally evaluates to UNDEFINED if it relates to an attribute value and the attribute type is not present in an attribute against which the assertion is being tested. An assertion which is defined by these conditions and relates to the presence of an attribute type evaluates to FALSE.

Attribute value assertions in filter items are evaluated using the matching rules defined for that attribute type, as substituted for, where applicable, in accordance with the action of a relaxation policy. Matching rule assertions are evaluated as specified in their definition. A matching rule defined for a particular syntax can only be used to make assertions about attributes of that syntax or subtypes of that syntax.

NOTE 2 – The action of a relaxation policy can cause a particular matching rule to revert to a **nullMatch** matching rule (which always evaluates as TRUE (if non-negated) or FALSE (if negated) – see 8.7.2 of ITU-T Rec. X.520 | ISO/IEC 9594-6.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

A **FilterItem** may be UNDEFINED (as described above). Otherwise, where the **FilterItem** asserts:

a) **equality** – It is TRUE if and only if there is a value of the attribute or one of its subtypes for which the **equality** matching rule applied to that value and the presented value returns TRUE.

b) **substrings** – It is TRUE if and only if there is a value of the attribute or one of its subtypes for which the **substring** matching rule applied to that value and the presented value in **strings** returns TRUE. See ITU-T Rec. X.520 | ISO/IEC 9594-6 for a description of the semantics of the presented value.

c) **greaterOrEqual** – It is TRUE if and only if there is a value of the attribute or one of its subtypes for which the **ordering** matching rule applied to that value and the presented value returns FALSE, i.e., there is a value of the attribute which is *greater than or equal to* the presented value.

d) **lessOrEqual** – It is TRUE if and only if there is a value of the attribute or one of its subtypes for which either the **equality** matching rule *or* the **ordering** matching rule applied to that value and the presented value returns TRUE, i.e., there is a value of the attribute which is *less than or equal to* the presented value.

e) **present** – It is TRUE if and only if the attribute or one of its subtypes is present in the entry.

f) **approximateMatch** – It is TRUE if and only if there is a value of the attribute or one of its subtypes for which some locally-defined approximate matching algorithm (e.g., spelling variations, phonetic match, etc.) returns TRUE. If an item matches for equality, it shall also satisfy an approximate match. Otherwise there are no specific guidelines for approximate matching in this edition of this Directory Specification. If approximate matching is not supported, this **FilterItem** should be treated as a match for **equality**.

g) **extensibleMatch** – It is TRUE if and only if there is a value of the attribute with the indicated **type** or one of its subtypes for which the matching rule specified in **matchingRule** applied to that value and the presented value **matchValue** returns TRUE.

If several matching rules are given, the way in which these rules are combined into a new rule is unspecified (it is a locally-defined algorithm, which reflects the semantics of the constituent matching rules, e.g., **phonetic** + **keyword** match).

If **type** is omitted, the match is made against all attribute types which are compatible with that matching rule. If **dnAttributes** is **TRUE**, the attributes of the Distinguished Name of the entry are used in addition to those of the entry in evaluating the match.

If an **extensibleMatch** is requested in a **filter** (rather than an **extendedFilter**), the **extendedFilter** bit in the **criticalExtensions** parameter in **CommonArguments** shall be set, indicating that the extension is critical.

If an implementation does not support any of the matching rules defined in the **matchingRule** subcomponent, or if none of the matching rules are compatible with the attribute type, an **extensibleMatch** filter item evaluates to UNDEFINED if the **performExactly** search control option is not set. If the **performExactly** search control option is set, the **search** request is rejected with:

– a **serviceError** with problem **unsupportedMatchingUse**;

– a **searchServiceProblem** notification attribute with the value **id-pr-unsupportedMatchingRule** if all the matching rules are unsupported, otherwise with the value **id-pr-unsupportedMatchingUse**;

– an **attributeTypeList** notification attribute having as value the attribute type for which the invalid matching rules were defined; and

– a **matchingRuleList** notification attribute having as values the object identifiers of the unsupported and/or incompatible matching rules.

NOTE 3 – An **extensibleMatch** is not permitted for first-edition systems.

h) **contextPresent** – It is TRUE if, and only if, the **AttributeTypeAssertion** for this attribute type or, if the **noSubtypeMatch** service control option is not set, one of its subtypes evaluates to TRUE.

If context assertions are included in an attribute value assertion in a filter item, then the filter item is evaluated against only those values which satisfy all the given context assertions, as described in 8.9.2 of ITU-T Rec. X.501 | ISO/IEC 9594-2. If no context assertions are included in an attribute value assertion, then default context assertions shall be applied as described in 8.9.2.2 of ITU-T Rec. X.501 | ISO/IEC 9594-2.

7.8.3 Evaluating filters with family information

Specific family groupings work as follows in fulfilling filter requirements:

entryOnly means that only family members that completely fulfil the filter requirements are marked as contributing and participating members (for the definition of contributing and participating members see 7.13).

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

compoundEntry means that the entire compound entry forms the group that shall satisfy the complete filter; within each compound entry that satisfies the filter, family members that contribute to the match are marked as contributing members, while all members of the compound entry are marked as participating entries.

strands means that the filter applies to each complete strand from a leaf to the ancestor. The compound entry matches the filter if at least one strand matches the filter. Family members on a matching strand that contribute to the match are marked as contributing members, while all the members on a matching strand are marked as participating members.

A strand is a set of members within a family that form a path from a leaf to the ancestor, so that there are as many strands as there are leaf entries.

multiStrand means that a combination of one strand from each family class is a family grouping for the purpose of matching. All combinations are to be considered one at the time. The compound entry matches the filter if at least one combination of strands matches the filter. Family members on a matching strand combination that contribute to the match are marked as contributing members, while all members of a matching strand combination are marked as participating members.

Two strands are of the same *family class* if and only if the family members that are immediately subordinate to the ancestor have the same structural object class.

A strand is *matched* for a filter if and only if it is present in at least one of all possible combinations of strands that causes the entry to match for the subfilter. The following are corollaries:

– If the ancestor matches the subfilter completely, *all* strands are matched.

– Similarly, if there are three family classes for a particular ancestor, and the subfilter is fulfilled by two of the classes without considering the third one, all strands for the third family class are matched.

multiStrand is only applicable if the base object is the ancestor (or higher) in the DIT. If the base object is a family member, but not the ancestor, then **multiStrand** shall be ignored and **entryOnly** shall be substituted.

7.9 Paged results

A **PagedResultsRequest** parameter is used by the DUA to request that the results of a List or Search operation be returned to it "page-by-page": it requests the DSA to return only a subset – a *page* – of the results of the operation, in particular the next **pageSize** subordinates or entries, and to return a **queryReference** which can be used to request the next set of results on a follow-up query.

Paged results may either be performed by the DSA to which the DUA has bound by a Bind operation (the bound DSA) or by the DSA that started the initial evaluation phase (the initial performer as detailed in 15.5.5 of ITU-T Rec. X.518 | ISO/IEC 9594-4).

It shall not be used if results are to be signed, unless there is an understanding among DSAs cooperating to provide the paged results that the DSA performing the paging may remove the signatures on results received from other DSAs and then itself sign the results to be returned toward the DUA. The way such an understanding is established is outside the scope of this Directory Specification. Although a DUA may request **pagedResults**, a DSA is permitted to ignore the request and return its results in the normal manner.

NOTE 1 – The result my be unpredictable in case of a configuration that is not "well-connected", e.g., where due to shadowing and use of NSSRs, the name resolution will locate more than one base object.

If paged results are requested and paging is performed, then the paging DSA shall ignore the **sizeLimit** service control, if any. If paging is not performed, the **sizeLimit** service control shall be honoured. A contributing DSA (see 15.5.5 of ITU-T Rec. | ISO/IEC 9594-4) shall honour the **sizeLimit** service control.

```
PagedResultsRequest ::= CHOICE {
    newRequest              SEQUENCE {
        pageSize                        INTEGER,
        sortKeys                        SEQUENCE SIZE (1..MAX) OF SortKey OPTIONAL,
        reverse             [1]     BOOLEAN DEFAULT FALSE,
        unmerged                [2]     BOOLEAN DEFAULT FALSE,
        pageNumber              [3]     INTEGER OPTIONAL },
    queryReference                  OCTET STRING,
```


--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

abandonQuery [0] OCTET STRING }

SortKey ::= SEQUENCE {

type AttributeType,

orderingRule MATCHING-RULE.&id OPTIONAL }

For a new list or search operation, the **PagedResultsRequest** is set to **newRequest**, which consists of the following parameters:

a) The **pageSize** parameter specifies the maximum number of subordinates or entries to return in the results. The DSA shall return up to but not more than the requested number of entries. The **sizeLimit**, if any, is ignored. The inclusion of family information does not count towards page size when packaged in **family-information** derived attributes.

b) The **sortKeys** parameter specifies a sequence of attribute types with optional ordering matching rules to use as sort keys for sorting the returned entries prior to return to the DUA. In the case of List operations, the sorting shall be by RDN, but the sorting requirements shall apply only to attributes within the RDN. In the case of Search operations, ordering shall only apply to attributes that are actually supplied (as a result of selection, and access control, with sorting by distinguished name as a fallback). The entries are sorted according to their values of the **type** attribute of the first **SortKey** in the sequence, and in the event of multiple entries having the same sort position, of the next **SortKey** in the sequence, and so on.

For a particular **SortKey**, the DSA uses the **orderingRule** matching rule if it is present, otherwise the **ordering** matching rule of the attribute if one is defined; it ignores the sort key if none is defined. If the attribute type is multi-valued, the "least" value is used; if the attribute type is missing from the returned results, it is regarded as "greater" than all other matched values. A DSA is permitted to support only certain sort key sequences (thus, a DSA that holds and returns its data in the internal order "alphabetic by surname" will be able to comply with only one sort key sequence). If it cannot support the requested sequence, it shall use a default sort sequence.

A hierarchical group shall not be separated, but returned in the sequence as specified by 10.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2. When sorting is performed, the first entry of a hierarchical group to be returned determines the position of the hierarchical group within the sorted result.

NOTE 2 – A hierarchical group may span pages.

c) If the **reverse** parameter is **TRUE**, then the DSA will return the sorted results in reverse order (i.e., from "greatest" to "least" – if the attribute type is multi-valued, the "greatest" is used; if the attribute type is missing from the returned results, it is regarded as "less" than all other matched values). If it is **FALSE**, the DSA returns them in forward order. If no **sortKeys** parameter is specified, this parameter is ignored.

d) If the **unmerged** parameter is **TRUE** and the DSA responsible for the paging is collecting results from a number of other DSAs, it shall return all the data from one DSA (in sort order) before returning data from the next DSA. If the parameter is **FALSE**, the DSA shall collect the results from all other DSAs and sort the merged data before returning any of it. If no **sortKeys** parameter is specified, this parameter is ignored. The semantic of the **unmerged** parameter is the same whether the DSA supports DSP paged results or not.

e) If the **pageNumber** parameter is present, it indicates the user wants to start with a particular page rather than necessarily the first one. This parameter shall be ignored if ordering is not requested.

For a follow-up request, i.e., to request the next set of paged results, the DUA makes the same list or search request as before, but sets **PagedResultsRequest** to **queryReference**, with the value of this parameter the same as that returned in the **PartialOutcomeQualifier** of the previous results. The DUA has no understanding of the **queryReference**, which is available to a DSA to use as it wishes to record context information for the query. The DSA uses this information to determine which results to return next.

The DUA may at any time indicate that no more pages are required by making the same **list** or **search** request as before, by setting the **PagedResultsRequest** set to **abandonQuery**, with the value identical to the **queryReference** value returned in the **PartialOutcomeQualifier** of the previous results. No further pages shall be requested or returned. It is implementation dependent as to when the pages will be purged.

In the case where the **queryReference** or the **abandonQuery** choice is made, the new request and the original information shall be identical in the following respects:

– **baseObject** within **SearchArgument** or **object** within **ListArgument** shall match for the present and the original request;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

- the **queryReference** subcomponent of **pagedResults** shall be identical to the **queryReference** value returned in the **PartialOutcomeQualifier** of the previous result;
- the options component of the **ServiceControls** data type shall specify identical options for the present and the original request;
- **operationProgress** (if present) shall be identical for the present and the original request.

Otherwise a **serviceError** with problem **invalidQueryReference** shall be returned.

NOTE 3 – If the DIB changes between **search** requests, the DUA may not see the effects of these changes. This is implementation dependent.

NOTE 4 – A query-reference may remain valid even if a DUA begins a new list or search operation. A DUA may request paged results with several queries and then return to an earlier query and request the next page of results using the query-reference supplied for it. The number of "active" query-references to which a DUA can return is a local DSA implementation option, as is the lifetime of those query-references.

NOTE 5 – Support of the **abandonQuery** choice is only available for post fourth edition systems.

NOTE 6 – When a DAP association terminates, access to all associated paged results is lost. Paged results can only be accessed within the DAP association within which they were originally invoked.

7.10 Security parameters

The **SecurityParameters** govern the operation of various security features associated with a Directory operation.

NOTE 1 – These parameters are conveyed from sender to recipient. Where the parameters appear in the argument of an operation the requestor is the sender, and the performer is the recipient. In a result, the roles are reversed.

```
SecurityParameters ::= SET {
        certification-path      [0]  CertificationPath                  OPTIONAL,
        name                    [1]  DistinguishedName          OPTIONAL,
        time                    [2]  Time                               OPTIONAL,
        random                  [3]  BIT STRING                         OPTIONAL,
        target                  [4]  ProtectionRequest          OPTIONAL,
        response                [5]  BIT STRING                         OPTIONAL,
        operationCode           [6]  Code                               OPTIONAL,
        attributeCertificationPath [7]  AttributeCertificationPath OPTIONAL,
        errorProtection         [8]  ErrorProtectionRequest     OPTIONAL,
        errorCode               [9]  Code                       OPTIONAL }

ProtectionRequest ::= INTEGER { none (0), signed (1) }

Time ::= CHOICE {
        utcTime                 UTCTime,
        generalizedTime         GeneralizedTime }

ErrorProtectionRequest ::= INTEGER { none (0), signed (1) }
```

The **CertificationPath** component is a sequence containing the signer's user certificate, and, optionally, a sequence of one or more certification authority (CA) certificates. (See clause 7 in ITU-T Rec. X.509 | ISO/IEC 9594-8.) The user certificate is used to bind the signer's public key and distinguished name, and may be used to verify the signature on a request argument, response, or error. This parameter shall be present and contain the signer's user certificate if the request argument, response, or error is signed. Additional certificates may be present and may be used to determine if the signer's user certificate is valid. Additional certificates are not required if the recipient shares the same certification authority as the signer. If the recipient requires a certification path for validation, and an acceptable parameter is not present, whether the recipient rejects the signature, or attempts to determine a certification path, is a local matter.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The **name** is the distinguished name of the first intended recipient of the argument or result. For example, if a DUA generates a signed argument, the name is the distinguished name of the DSA to which the operation is submitted.

NOTE 2 – Where the first intended recipient has alternative distinguished names differentiated by context, **name** may be an alternative name. However, authentication and access control which may be based on the value of **name** may not work as desired if the primary distinguished name is not used.

The **time** is the intended expiry time for the validity of the request, response, or error. It is used in conjunction with the random number to enable the detection of replay attacks.

The **random** value is a number that should be different for each request, response, or error. It is used in conjunction with the time parameter to enable the detection of replay attacks. If sequence integrity is required then the random argument may be used to carry a sequence integrity number as follows:

a) The random value used with operation arguments is derived using a pre-agreed sequence (e.g., the previous value plus 1) from:

   i) for the first operation sent from a system on a binding, the random value passed in the bind operation argument/result by the remote peer system; and

   ii) for subsequent operations, the random value passed in the previous operation in the same direction.

b) The random value used with operation results or errors is derived using some pre-agreed sequence from the random value in the request (e.g., random in request argument plus 1).

The **target ProtectionRequest** may appear only in the request for an operation to be carried out, and indicates the requestor's preference regarding the degree of protection to be provided to the result. Two levels are provided: **none** (no protection requested, the default), and **signed** (the Directory is requested to sign the result). The degree of protection actually provided to the result is indicated by the form of result and may be equal to or lower than that requested, based on the limitations of the Directory.

The **response** is used to convey any information back to the initiator of the request.

The **operationCode** is used to securely bind the operation code to the request arguments, results or errors.

The **attributeCertificationPath** is used to convey a security clearance for rule based access control, or other attribute, in an Attribute Certificate, optionally with the certificates needed to validate the Attribute Certificate.

The **errorProtection** request may appear only in the request for an operation to be carried out, and indicates the requestor's preference regarding the degree of protection to be provided to any error. Two levels are provided: **none** (no protection requested, the default), and **signed** (the Directory is requested to sign the error). The degree of protection actually provided to the error is indicated by the form of error and may be equal to or lower than that requested, based on the limitations of the Directory.

NOTE 3 – A DUA may request that any security label context be returned with an attribute value using the context selection.

The **errorCode** is used to secure the error code where an error is returned in response to an operation.

If the syntax of **Time** has been chosen as the **UTCTime** type, the value of the two-digit year field shall be rationalized into a four-digit year value as follows:

– If the 2-digit value is 00 through 49 inclusive, the value shall have 2000 added to it.

– If the 2-digit value is 50 through 99 inclusive, the value shall have 1900 added to it.

GeneralizedTime shall be used if the negotiated version is **v2** or greater. The use of **GeneralizedTime** when **v1** has been negotiated may prevent interworking with implementations unaware of the possibility of choosing either **UTCTime** or **GeneralizedTime**. It is the responsibility of those specifying the domains in which this Directory Specification will be used, e.g., profiling groups, as to when the **GeneralizedTime** may be used. In no case shall **UTCTime** be used for representing dates beyond 2049.

7.11 Common elements of procedure for access control

This subclause defines the elements of procedure that are common to all abstract service operations when **basic-access-control**, **rule-based-access-control**, or both are in effect. If both mechanisms are in effect, the order in which they are applied is a local matter, except that if access is denied to the entry, an attribute type or an attribute value by either mechanism, then a grant from the other mechanism shall not override it. In this respect, *DiscloseOnError* permission of **basic-access-control** is a grant that shall not override a deny of **rule-based-access-control**.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.11.1 Common elements of procedure for basic access control

7.11.1.1 Alias dereferencing

If, in the process of locating a target object entry (identified in the argument of an abstract service operation), alias dereferencing is required, no specific permissions are necessary for alias dereferencing to take place. However, if alias dereferencing would result in a **ContinuationReference** being returned (i.e., in a **Referral**), the following sequence of access controls applies. If the DSA chains the request to another DSA and receives back a referral from it, then the access controls shall be applied to the referral if the **targetObject** in the referral is the same as in the chained request. That is, the DSA shall police all referrals whether they were generated locally or remotely.

1) *Read* permission is required to the alias entry. If permission is not granted, the operation fails in accordance with the procedure described in 7.11.1.
2) *Read* permission is required to the **aliasedEntryName** attribute and to the single value that it contains. If permission is not granted, the operation fails and **nameError** with problem **aliasDereferencingProblem** shall be returned. The **matched** element shall contain the name of the alias entry.

NOTE – In addition to the access controls described above, security policy may prevent the disclosure of knowledge information which would otherwise be conveyed as a **ContinuationReference** in **Referral**. If such a policy is in effect and if a DUA constrains the service by specifying **chainingProhibited** the Directory may return a **serviceError** with problem **chainingRequired**. Otherwise, a **securityError** with problem **insufficientAccessRights** or **noInformation** shall be returned.

7.11.1.2 Return of Name Error

If, while performing an abstract service operation, the specified target object (alias or entry) – e.g., the Name of an entry to be read or the **baseObject** in a **search** request – could not be found, a **nameError** with problem **noSuchObject** shall be returned. The **matched** element shall either contain the name of the next superior entry to which *DiscloseOnError* permission is granted, or the name of the DIT root (i.e., an empty **RDNSequence**).

NOTE – The second alternative may be taken by a DSA which does not have access to all superior entries.

7.11.1.3 Non-disclosure of the existence of an entry

If access is denied under **rule-based-access-control**, then the *DiscloseOnError* permission is not applicable.

If, while performing an abstract service operation, the necessary entry level permission is not granted to the specified target object entry – e.g., the entry to be read – the operation fails and the error returned is one of: if *DiscloseOnError* permission is granted to the target entry, a **securityError** with problem **insufficientAccessRights** or **noInformation** shall be returned; otherwise, a **nameError** with problem **noSuchObject** shall be returned. The **matched** element shall either contain the name of the next superior entry to which *DiscloseOnError* permission is granted, or the name of the DIT root (i.e., an empty **RDNSequence**).

NOTE – The second alternative may be taken by a DSA which does not have access to all superior entries.

Additionally, whenever the Directory detects an operational error (including a referral), it shall ensure that in returning that error it does not compromise the existence of the named target entry and any of its superiors. For example, before returning a **serviceError** with problem **timeLimitExceeded** or an **updateError** with problem **notAllowedOnNonLeaf**, the Directory verifies that *discloseOnError* permission is granted to the target entry. If it is not, the procedure described in the paragraph above shall be followed.

7.11.1.4 Return of Distinguished Name

In a Compare, List, or Search operation, *ReturnDN* permission is required to the **object** (or **baseObject**) entry if as a result of dereferencing an alias, the object's distinguished name is to be returned in the **name** parameter of the operation result (see 9.2.3). If this permission is not granted, the Directory shall return an alias name for the entry instead, as described in 7.7, or shall omit the **name** parameter altogether.

In a Read or Search operation, *ReturnDN* permission is required to an entry in order to return its distinguished name in **EntryInformation**. If this permission is not granted, the Directory shall return the name of an alias instead, as described in 7.7, or if no alias name is available shall fail the operation with a **nameError** (in the case of Read) or omit the entry from the results (in the case of Search).

If the user supplied alias name is returned in the result, then the **aliasDeferenced** flag of **CommonResults** shall not be set to **TRUE**.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.11.2 Common elements of procedure for rule-based-access-control

7.11.2.1 Accessing an entry (entry level permission)

In order to access an entry, permission is required to access at least one attribute value in the entry. If entry level permission is not granted, then **nameError** with problem **noSuchObject** shall be returned.

7.11.2.2 Returning the name of an entry

In order to return the DN of an entry, permission is required to access all the attribute values of at least one context variant of the RDN of the entry (this is termed *RDN* permission). No permissions are required from any of the superiors of the entry. If RDN permission is not granted, then a DSA may choose to either return the DN of a valid alias of the entry for which RDN permission has been granted, or to omit the name component from the operation result.

NOTE – The selection of an appropriate alias name is further described in the Notes of 7.7.

7.11.2.3 Alias dereferencing

In order to dereference an alias, permission is required to access the **aliasedEntryName** attribute value.

7.11.2.4 Return of Name Error (noSuchObject)

The **matched** component of **nameError** with problem **noSuchObject** shall be set to the name of the next superior entry to which the requestor has RDN permission. If such an entry is not available to the DSA generating the error, then the name of the DIT root shall be returned.

7.11.2.5 Accessing an attribute

In order to access an attribute, permission is needed to access at least one of the values of the attribute.

7.11.2.6 Deleting information

In order to delete an attribute value, permission is needed to access that value. When deleting an entry or an attribute, the operation shall return a successful response if at least one attribute value is deleted, irrespective of how many values were requested to be deleted.

7.11.2.7 Invoking search-rules

In order to evaluate a search-rule against the arguments of a search operation, invoke permission to the search-rule is required for the requestor originating the search operation. The user needs no other permissions to access the search-rule attribute or the subentry that contains it.

7.11.3 Family information

Family information is treated as for any other information, except that ACI for which the **ProtectedItem** is marked as **includeFamily**; if the ACI is applicable to an ancestor or family member this causes subordinate family members to be subject to the same ACI. **IncludeFamily** is only meaningful when applied to an **entry** protected item.

7.12 Managing the DSA Information Tree

The DSA Information Tree held by a DSA can be managed using the Directory Abstract Service. When the DSA Information Tree is managed:

- all DSEs in a DSA are visible through DAP including the root DSE;
- attributes defined as no user modification may be modified (though the DSA can reply with an **serviceError** with problem **unwillingToPerform** if it cannot support the requested change);
- knowledge is merely another attribute which can be read and modified; and
- the DSA never chains requests or returns referrals or continuation references.

Visibility of DSEs and retrieval of or changes to operational attributes can be controlled via access control in the normal way.

The management of a DSA Information Tree is achieved by a DUA using the following procedures:

1) the DUA BINDs directly to the DSA which holds the DSA Information Tree that is to be managed;
2) for each operation that is used to manage the DSA Information Tree:
   - the **manageDSAIT** extension bit shall be set;
   - the **manageDSAIT** option shall be set;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

– the **manageDSAITPlaneRef** option shall be included if a specific replication plane is to be managed;

the following components are ignored by the Directory:

– **operationProgress** in **CommonArgument**;

– **referenceType** in **CommonArgument**;

– **entryOnly** in **CommonArgument**;

– **nameResolveOnMaster** in **CommonArgument**; and

– **chainingProhibited** in **ServiceControls**.

7.13 Procedures for families of entries

As specified in 7.3.2, family members within a compound entry may be grouped together for the purpose of operation evaluation. This grouping is only relevant for Compare, Search and Remove Entry operations. If family grouping is specified for any other operation, it shall be ignored.

For determining what family members that shall be returned according to the **familyReturn** component of the **entryInformationSelection**, the concepts of *contributing member* and *participating member* is introduced. These concepts are only relevant for operations that return entry information, i.e., Read, Search and Modify Entry operations.

If a family member makes an active contribution to the operation evaluation, it is marked as a contributing member. A family member makes a contribution to the match if it is part of a family grouping that match the filter and if it holds one or more attributes that are matched by non-negated filter items. It also contributes if it holds an attribute of a given type if a negated filter item for the same type does not match. In the case of a Read or Modify Entry operation, the family member that is selected by the operation (as specified by the **object** component of the operation) is the only member marked as a contributing member and as a participating member. In the case of a Search operation, family grouping is done for filter matching. If a family grouping matches a filter (see 7.8.3) all members that have contributed actively to the matching are marked as contributing members, while all entries of the grouping are marked as participating members. If the filter used is the default filter (**and : { }**), then all members of a family grouping shall be marked as participating members, but not as contributing members.

When a family grouping of compound entry matches the filter and the **SearchArgument** specifies hierarchy selection (except for **self**), the selected entries shall also be marked if applicable. If the ancestor of the compound entry is marked as participating (and possibly also as contributing), all referenced entries of the hierarchical group that are not compound entries shall be selected, otherwise they shall be excluded. If a referenced entry is a compound entry, the marking of its members shall be done as follows. Each member of the referenced compound entry that have the same local member name as a member of the matched compound entry is marked the same way. All other members of the referenced compound entry are left unmarked.

As a Search filter possibly can match several compound entries, the resulting selection and marking shall be the union of those for the individual matched compound entries.

If a matched entry not being a compound entry references a compound entry in its hierarchy selection, all the members of that compound entry are marked as participating.

How this marking of entries affects the return of entry information is detailed in 7.6.4.

Family members may be packed into a **family-information** derived attribute. If only a single member of a compound entry is returned in the result, packaging shall not be performed. However, if several members are returned from a Read or Modify Entry operation, these members shall be packed. In the case of a Search operation where several members of a compound attribute are returned they shall be packed unless the **separateFamilyMembers** search control option is set, in which case the members shall be returned as separate entries.

When performing search operations involving compound entries, there are four relevant phases for a Search operation:

a) The groupings of family members within each entry of interest, as defined by **familyGrouping**, are logically considered within each candidate entry (i.e., as selected by subset). By pooling together all the attributes of the group, all attribute values for a given attribute type are considered to belong to that single attribute type, even if they originated from different family members.

b) The filter is applied to each family grouping; if the filter is satisfied for the grouping, the compound entry then satisfies the filter, and is considered to be selected by the filter. Family members are marked as described above.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

c) The marked entries are augmented, as specified by **familyReturn** in **EntryInformationSelection**, to mark all entries that would be returned.

d) If the **additionalControl** component is present in a governing-search-rule (see 16.10.8 of ITU-T Rec. X.501 | ISO/IEC 9594-2), the markings, and thereby what is returned, may be changed as the result of processing the referenced control attributes.

8 Bind and Unbind operations

The Directory Bind and Directory Unbind operations, defined in 8.1 and 8.2 respectively, are used by the DUA at the beginning and end of a particular period of accessing the Directory.

8.1 Directory Bind

8.1.1 Directory Bind syntax

A Directory Bind operation is used at the beginning of a period of accessing the Directory. The arguments of the operation may be signed (see 17.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2) by the requestor. If so requested, the Directory may sign the results.

```
directoryBind OPERATION ::= {
    ARGUMENT DirectoryBindArgument
    RESULT DirectoryBindResult
    ERRORS { directoryBindError } }

DirectoryBindArgument ::= SET {
    credentials [0] Credentials OPTIONAL,
    versions [1] Versions DEFAULT {v1} }

Credentials ::= CHOICE {
    simple [0] SimpleCredentials,
    strong [1] StrongCredentials,
    externalProcedure [2] EXTERNAL,
    spkm [3] SpkmCredentials,
    sasl [4] SaslCredentials }

SimpleCredentials ::= SEQUENCE {
    name [0] DistinguishedName,
    validity [1] SET {
        time1 [0] CHOICE {
            utc UTCTime,
            gt GeneralizedTime } OPTIONAL,
        time2 [1] CHOICE {
            utc UTCTime,
            gt GeneralizedTime } OPTIONAL,
        random1 [2] BIT STRING OPTIONAL,
        random2 [3] BIT STRING OPTIONAL } OPTIONAL,
    password [2] CHOICE {
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        unprotected         OCTET STRING,
        protected               HASH {OCTET STRING} } OPTIONAL}

StrongCredentials ::= SET {
    certification-path          [0]  CertificationPath OPTIONAL,
    bind-token                  [1]  Token,
    name                        [2]  DistinguishedName OPTIONAL,
    attributeCertificationPath  [3]  AttributeCertificationPath OPTIONAL }

SpkmCredentials ::= CHOICE {
    req  [0]  SPKM-REQ,
    rep  [1]  SPKM-REP-TI }

SaslCredentials ::= SEQUENCE {
    mechanism           [0]  DirectoryString { ub-saslMechanism },
    credentials         [1]  OCTET STRING OPTIONAL,
    saslAbort           [2]  BOOLEAN DEFAULT FALSE }

ub-saslMechanism INTEGER ::= 20     -- According to RFC 2222

Token ::= SIGNED { TokenContent }

TokenContent ::= SEQUENCE {
    algorithm               [0]  AlgorithmIdentifier{{SupportedAlgorithms}},
    name                [1]  DistinguishedName,
    time                    [2]  Time,
    random                  [3]  BIT STRING,
    response                [4]  BIT STRING OPTIONAL }

Versions ::= BIT STRING {v1(0), v2(1) }

DirectoryBindResult ::= DirectoryBindArgument

directoryBindError ERROR ::= {
    PARAMETER           OPTIONALLY-PROTECTED {
        SET {
            versions                [0]  Versions DEFAULT {v1},
            error                   CHOICE {
                serviceError        [1]  ServiceProblem,
                securityError       [2]  SecurityProblem },
            securityParameters  [30]  SecurityParameters OPTIONAL } } }
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

BindKeyInfo ::= ENCRYPTED { BIT STRING }

8.1.2 Directory Bind arguments

The **credentials** argument of the **DirectoryBindArgument** allows the Directory to establish the identity of the user. The credentials may be **simple**, or **strong** or externally defined (**externalProcedure**) (as described in ITU-T Rec. X.509 | ISO/IEC 9594-8).

If **simple** is used, it consists of a **name** (always the distinguished name of an object), an optional **validity**, and an optional **password**. This provides a limited degree of security. The **password** may be **unprotected**, or it may be **protected** (either Protected1 or Protected2) as described in 18.1 of ITU-T Rec. X.509 | ISO/IEC 9594-8. The **validity** supplies **time1**, **time2**, **random1** and **random2** arguments, which derive their meaning by bilateral agreement, and which may be used to detect replay. In some instances a protected password may be checked by an object which knows the password only after locally regenerating the protection to its own copy of the password and comparing the result with the value in the bind argument (**password**). In other instances, a direct comparison may be possible. A possible approach for protected password may be found in an informative annex of ITU-T Rec. X.509 | ISO/IEC 9594-8.

GeneralizedTime shall be used for **time1** and **time2** if the negotiated version is **v2** or greater. The use of **GeneralizedTime** when **v1** has been negotiated may prevent interworking with implementations unaware of the possibility of choosing either **UTCTime** or **GeneralizedTime**. It is the responsibility of those specifying the domains in which this Directory Specification will be used, e.g., profiling groups, as to when the **GeneralizedTime** may be used. **UTCTime** shall not be used for representing dates beyond 2049.

If **strong** is used, it consists of a **bind-token**, and, optionally, a **certification-path** (certificate and sequence of certification-authority cross-certificates, as defined in ITU-T Rec. X.509 | ISO/IEC 9594-8) and the **name** of the requestor. This enables the Directory to authenticate the identity of the requestor establishing the association, and vice versa. If **StrongCredentials** or **SpkmCredentials** are used in a bind operation, information relating to identity and authorization is conveyed. This enables the identity of either entity to be authenticated, and also enables use of established encryption and integrity cryptographic keying material.

If the operation is to be signed and encrypted, an attribute certificate containing the attribute (see clause 12 of ITU-T Rec. X.509 | ISO/IEC 9594-8) may be used to convey the clearances required to access the attribute. The **attributeCertificationPath** is used to convey a security clearance for rule based access control, or other attribute, conveyed in an Attribute Certificate, optionally with the certificates needed to validate the Attribute Certificate.

The arguments of the bind token are used as follows. **algorithm** is the identifier of the algorithm employed to sign this information. **name** is the name of the intended recipient. The **time** parameter contains the expiry time of the token. The **random** number is a number which should be different for each unexpired token, and may be used by the recipient to detect replay attacks.

> NOTE – Where names are used in either simple or strong credentials, it is possible to use alternative distinguished names if they exist. However, authentication and access control based on the name may not work as desired if the primary distinguished name is not used. Following successful processing of an authenticated BIND operation, whatever the name used in the BIND argument, the bound entities shall thereafter know each other by their primary distinguished names, to facilitate operation of access controls while the BIND is in effect.

If **externalProcedure** is used, then the semantics of the authentication scheme being used is outside the scope of the Directory Specifications.

sasl is used when using the Simple Authentication and Security Layer (SASL) specified in RFC 2222. If a **directoryBind** operation is invoked with a **SaslCredentials** mechanism value set to the empty string, a **SecurityError** of **inappropriateAuthentication** shall be returned.

The **versions** argument of the **DirectoryBindArgument** identifies the versions of the service which the DUA is prepared to participate in. The value **v1** denotes the protocol version 1 and the value **v2** denotes the protocol version 2. The value **v2** shall be used if in a subsequent **ModifyEntry** operation the **alterValues** or **resetValue** modification types are to be sent in a request or a result other than **NULL** is required (see 11.3). The value shall be set to **v2** if signing of errors or result to Add Entry, Remove Entry, Modify Entry, Modify DN is used.

Migration to future versions of the Directory should be facilitated by:

a) any elements of **DirectoryBindArgument** other than those defined in this Directory Specification shall be accepted and ignored;

b) additional options for named bits of **DirectoryBindArgument** (e.g., versions) not defined shall be accepted and ignored.

The **response** component is used to carry a number derived from random if challenge response of authentication is required.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

8.1.3 Directory Bind results

Should the bind request succeed, a result shall be returned.

The **credentials** argument of the **DirectoryBindResult** allows the user to establish the identity of the Directory. It allows information identifying the DSA (that is directly providing the Directory service) to be conveyed to the DUA. It shall be of the same form (i.e., **CHOICE**) as that supplied by the user.

The **versions** parameter of the **DirectoryBindResult** indicates which of the versions of the service requested by the DUA is actually going to be provided by the DSA.

8.1.4 Directory Bind errors

Should the bind request fail, a bind error shall be returned. If the Bind request was either using strong authentication or SPKM credentials are supplied, then the Bind responder may sign the error parameters.

The **versions** parameter of the **directoryBindError** indicates which versions are supported by the DSA.

The **SecurityParameters** components (see 7.10) shall be included if the error is to be signed.

A **securityError** or **serviceError** shall be supplied as follows:

– **securityError inappropriateAuthentication**
invalidCredentials
blockedCredentials

– **serviceError unavailable**
saslBindInProgress

8.2 Directory Unbind

The unbinding at the end of a period of accessing the Directory is for the OSI environment specified in 7.6.4 and 7.6.5 of ITU-T Rec. X.519 | ISO/IEC 9594-5 and for the TCP/IP environment in 9.3.2 of ITU-T Rec. X.519 | ISO/IEC 9594-5.

NOTE – On unbinding, all paged results not so far accessed become inaccessible, and should be disposed of.

9 Directory Read operations

There are two 'read-like' operations: **read** and **compare**, defined in 9.1 and 9.2, respectively. The **abandon** operation, defined in 9.3, is grouped with these operations for convenience.

9.1 Read

9.1.1 Read syntax

A Read operation is used to extract information from an explicitly identified entry. It may also be used to verify a distinguished name. The arguments of the operation may be signed (see 17.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2) by the requestor. If so requested, the Directory may sign the result.

```
read OPERATION ::= {
    ARGUMENT ReadArgument
    RESULT   ReadResult
    ERRORS   { attributeError | nameError | serviceError | referral | abandoned |
               securityError }
    CODE     id-opcode-read }
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
ReadArgument ::= OPTIONALLY-PROTECTED {
    SET {
        object                  [0]  Name,
        selection               [1]  EntryInformationSelection  DEFAULT { },
        modifyRightsRequest     [2]  BOOLEAN DEFAULT FALSE,
        COMPONENTS OF                CommonArguments } }

ReadResult ::= OPTIONALLY-PROTECTED {
    SET {
        entry           [0]  EntryInformation,
        modifyRights    [1]  ModifyRights OPTIONAL,
        COMPONENTS OF        CommonResults } }

ModifyRights ::= SET OF SEQUENCE {
    item                     CHOICE {
        entry           [0]  NULL,
        attribute       [1]  AttributeType,
        value           [2]  AttributeValueAssertion },
    permission          [3]  BIT STRING { add (0), remove (1), rename (2), move (3) } }
```

9.1.2 Read arguments

The **object** argument identifies the object entry from which information is requested. Should the **Name** involve one or more aliases, they are dereferenced (unless this is prohibited by the relevant service controls). The **Name** may be an alternative name and may include context information, as described in 9.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2.

The **selection** argument indicates what information from the entry is requested (see 7.6). However, it should not be assumed that the attributes returned are the same as or limited to those requested.

The **CommonArguments** (see 7.3) include a specification of the service controls and security parameters applying to the request. For the purposes of this operation the **sizeLimit** component is not relevant and is ignored if provided. If the argument of this operation is to be signed by the requestor, the **SecurityParameters** (see 7.10) component shall be included in the arguments.

The **modifyRightsRequest** argument is used to request return of the requestor's modification rights to the entry and its attributes.

9.1.3 Read results

Should the request succeed, the result shall be returned.

The entry result parameter holds the requested information (see 7.7). This may include family information, if required by the presence of a **familyReturn** element in **EntryInformationSelection**.

The **modifyRights** parameter is present if it was requested via the **modifyRightsRequest** argument, and the user has modification privileges to some or all of the requested entry information, and the return of this information is permitted by the local security policy. If returned, the modification rights of the requestor are returned for the entry and for the attributes specified in the **selection** argument. The parameter contains the following:

- An element of the **SET** is returned for the **entry**; for each user **attribute** requested which the user has the right to add or remove; and for each returned attribute **value** for which the user's rights to add or remove it differ from those of the corresponding attribute.
- The returned **permission** indicates what operations or actions on the entry by the user would succeed. In the case of an entry, **remove** indicates that a **RemoveEntry** operation would succeed; **rename** indicates that a **ModifyDN** operation with the **newSuperior** parameter absent would succeed; and **move** that a **ModifyDN** operation with the **newSuperior** parameter present and an unchanged RDN would succeed.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

In the case of attributes and values, **add** indicates that a **ModifyEntry** operation that adds the attribute or value would succeed; and **remove** indicates that a **ModifyEntry** operation that removes the attribute or value would succeed.

NOTE – An operation to move an entry to a new superior may also depend on permissions associated with the new superior (as for example with **basic-access-control**). These are ignored when determining **permission**.

The **CommonResults** (see 7.4) include the security parameters applying to the response. If this result is to be signed, encrypted by the Directory, the **SecurityParameters** (see 7.10) component shall be included in the results.

9.1.4 Read errors

Should the request fail, one of the listed errors shall be reported. If none of the attributes explicitly listed in **selection** can be returned, then an **attributeError** with problem **noSuchAttributeOrValue** shall be reported. The circumstances under which other errors shall be reported are defined in clause 12.

9.1.5 Read operation decision points for basic access control

If **rule-based-access-control** is also to be applied, the order in which it is applied with respect to **basic-access-control** is a local matter, except that if access is denied to the entry, an attribute type or an attribute value by either mechanism, it shall not be overridden by the other mechanism. In this respect, *DiscloseOnError* permission of **basic-access-control** is a permission that shall not override a deny of **rule-based-access-control**.

If **basic-access-control** is in effect for the entry being read, the following sequence of access controls applies:

1) *Read* permission is required to the entry being read. If permission is not granted, the operation fails in accordance with 7.11.1.3.

2) If the **infoTypes** element of **selection** specifies that attribute types only are to be returned, then for each attribute type that is to be returned, *Read* permission is required. If permission is not granted, the attribute type is omitted from the **ReadResult**. If as a consequence of applying these controls no attribute information is returned, the entire operation fails in accordance with 9.1.5.1.

3) If the **infoTypes** element of **selection** specifies that attribute types and values are to be returned, then for each attribute type and for each value that is to be returned, *Read* permission is required. If permission to an attribute type is not granted, the attribute is omitted from **ReadResult**. If permission to an attribute value is not granted, the value is omitted from its corresponding attribute. In the event that permission is not granted to any of the values within the attribute, an **Attribute** element containing an empty **SET OF AttributeValue** is returned. If as a consequence of applying these controls, no attribute information is returned, the entire operation fails in accordance with 9.1.5.1.

NOTE – Privileges that permit a DAP read operation may not work in an LDAP environment where browse permission is required to support an equivalent read service.

9.1.5.1 Error returns

If the operation fails as defined in 9.1.5 items 2) or 3), the valid error returns are one of:

a) If an open-ended option was specified (i.e., **allUserAttributes** or **allOperationalAttributes**), a **securityError** with problem **insufficientAccessRights** or **noInformation** shall be returned.

b) Otherwise, if a **select** option was specified (in **attributes** and/or in **extraAttributes**), then if the *DiscloseOnError* permission is granted to any of the selected attributes, a **securityError** with problem **insufficientAccessRights** or **noInformation** shall be returned. Otherwise, an **attributeError** with problem **noSuchAttributeOrValue** shall be returned.

9.1.5.2 Non-disclosure of incomplete results

If an incomplete result is being returned in **EntryInformation**, i.e., some of the attributes or attribute values have been omitted because of the applicable access controls, the **incompleteEntry** element shall be set to **TRUE** if *DiscloseOnError* permission is granted to at least one attribute type withheld from the result, or at least one attribute value withheld from the result (for which attribute type *Read* permission was granted).

9.1.6 Read operation decision points for rule-based access control

If **basic-access-control** is also applied, the order in which it is applied with respect to **rule-based-access-control** is a local matter, except that if access is denied to the entry, an attribute type or an attribute value by either mechanism, it shall not be overridden by the other mechanism. In this respect, *DiscloseOnError* permission of **basic-access-control** is a permission that shall not override a deny of **rule-based-access-control**.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

If **rule-based-access-control**, **rule-and-basic-access-control**, or **rule-and-simple-access-control** is in effect for the entry being read, the following access controls apply:

1) If entry level access is denied under **rule-based-access-control,** then the operation fails with **nameError** with problem **noSuchObject** in accordance with 7.11.2.4.
2) If access to the entry is not permitted under the **basic-access-control** scheme as described in 9.1.5 item 1) then the operation fails in accordance with 7.11.1.3.
3) If the **infoTypes** element of **selection** specifies that attribute types only are to be returned, then if under **rule-based-access-control,** access is not granted for all attribute values of that type, the attribute type is omitted from the **ReadResult**. If as a consequence of applying these controls no attribute information is returned, the entire operation fails returning an **attributeError** with problem **noSuchAttributeOrValue** in accordance with 9.1.5.1 b).
4) If the **infoTypes** element of **selection** specifies that attribute types only are to be returned, **basic-access-control** is applied as described in 9.1.5 item 2).
5) Under **rule-based access controls**, if the **infoTypes** element of selection specifies that attribute types and values are to be returned, then for each attribute value that is to be returned, access shall be granted. If access to an attribute value is not granted, the attribute value is omitted from its corresponding attribute. In the event that access is not granted to any of the attribute values within an attribute, the whole attribute is omitted from **ReadResult**. If as a consequence of applying these controls, no attribute information is returned, the entire operation fails returning an **attributeError** with problem **noSuchAttributeOrValue**.
6) **basic-access-control** is applied as described in 9.1.5 item 3).
7) The name of the entry returned in the operation result is determined as defined in 7.11.2.2.

9.2 Compare

9.2.1 Compare syntax

A Compare operation is used to compare a value (which is supplied as an argument of the request) with the value(s) of a particular attribute type in a particular object entry. The arguments of the operation may be signed (see 17.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2) by the requestor. If so requested, the Directory may sign the result.

Any value of **familyGrouping** except **multiStrand** may be used, and the attributes in all the grouped family members are to be used in the comparison against the purported attribute value assertion. If **familyGrouping** specifies **multiStrand**, **compoundEntry** is assumed.

```
compare OPERATION ::= {
        ARGUMENT CompareArgument
        RESULT          CompareResult
        ERRORS          { attributeError | nameError | serviceError | referral | abandoned |
                        securityError }
        CODE        id-opcode-compare }

CompareArgument ::= OPTIONALLY-PROTECTED {
        SET {
                object                  [0]     Name,
                purported               [1]     AttributeValueAssertion,
                COMPONENTS OF           CommonArguments } }

CompareResult ::= OPTIONALLY-PROTECTED {
        SET {
                name                            Name OPTIONAL,
                matched                 [0]     BOOLEAN,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
fromEntry          [1]  BOOLEAN DEFAULT TRUE,
matchedSubtype     [2]  AttributeType OPTIONAL,
COMPONENTS OF      CommonResults } }
```

9.2.2 Compare arguments

The **object** argument is the name of the particular object entry concerned. Should the **Name** involve one or more aliases, they are dereferenced (unless prohibited by the relevant service control). The **Name** may be an alternative name and may include context information, as described in 9.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2.

The **purported** argument identifies the attribute type and value to be compared with that in the entry. The comparison is TRUE if the entry holds the purported attribute type or one of its subtypes, or there is a collective attribute of the entry which is the purported attribute type or one of its subtypes (see 7.6), and if there is a value of that attribute which matches the purported value using the attribute's **equality** matching rule.

NOTE – A **compare** request cannot be satisfied by a friend attribute type of the attribute type specified in the argument.

If context assertions are included in the attribute value assertion, then the matching shall be attempted only against those values which satisfy all the given context assertions, as described in 8.9.2 of ITU-T Rec. X.501 | ISO/IEC 9594-2. If no context assertions are included in the attribute value assertion, then default context assertions shall be applied as described in 8.9.2.2 of ITU-T Rec. X.501 | ISO/IEC 9594-2.

The **CommonArguments** (see 7.3) include a specification of the service controls and security parameters applying to the request. For the purposes of this operation the **sizeLimit** component is not relevant and is ignored if provided. If the argument of this operation is to be signed by the requestor, the **SecurityParameters** (see 7.10) component shall be included in the arguments.

9.2.3 Compare results

Should the request succeed (i.e., the comparison is actually carried out), the result shall be returned.

The **name** is the distinguished name of the entry or an alias name of the entry, as described in 7.7. It is present only if an alias has been dereferenced, RDNs have been resolved to primary RDNs, or context selection has been applied and the name to be returned differs from the **object** name supplied in the operation argument.

The **matched** result parameter, holds the result of the comparison. The parameter takes the value **TRUE** if the values were compared and matched, and **FALSE** if they did not.

If **fromEntry** is **TRUE** the information was compared against the entry; if **FALSE** the information was compared against a copy.

The **matchedSubtype** parameter is present only if the result of the match was TRUE and if the match succeeded because a subtype of the purported attribute was matched. It contains the matched subtype. If more than one such subtype is available, the one highest in the hierarchy is returned.

The **CommonResults** (see 7.4) include the security parameters applying to the response. If this result is to be signed by the Directory, the **SecurityParameters** (see 7.10) component shall be included in the results.

9.2.4 Compare errors

Should the request fail, one of the listed errors shall be reported. The circumstances under which the particular errors shall be reported are defined in clause 12.

9.2.5 Compare operation decision points for basic access control

If **rule-based-access-control** is also applied, the order in which it is applied with respect to **basic-access-control** is a local matter, except that if access is denied to the entry, an attribute type or an attribute value by either mechanism, it shall not be overridden by the other mechanism. In this respect, *DiscloseOnError* permission of **basic-access-control** is a permission that shall not override a deny of **rule-based-access-control**.

If **basic-access-control** is in effect for the entry being compared, the following sequence of access controls applies:

1) *Read* permission is required to the entry to be compared. If permission is not granted, the operation fails in accordance with 7.11.1.3.
2) *Compare* permission is required to the attribute being compared. If permission is not granted, the operation fails in accordance to 9.2.5.1.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

3) If there exists a value within the attribute being compared that matches the **purported** argument and for which *Compare* permission is granted, the operation returns the value **TRUE** in the **matched** result parameter of the **CompareResult**. Otherwise, the operation returns the value **FALSE**.

9.2.5.1 Error returns

If the operation fails as defined in 9.2.5 item 2), the valid error returns are one of: if the *DiscloseOnError* permission is granted to the attribute being compared, a **securityError** with problem **insufficientAccessRights** or **noInformation** shall be returned; otherwise, an **attributeError** with problem **noSuchAttributeOrValue** shall be returned.

9.2.6 Compare operation decision points for rule-based access control

If **basic-access-control** is also applied, the order in which it is applied with respect to **rule-based-access-control** is a local matter, except that if access is denied to the entry, an attribute type or an attribute value by either mechanism, it shall not be overridden by the other mechanism. In this respect, *DiscloseOnError* permission of **basic-access-control** is a permission that shall not override a deny of **rule-based-access-control**.

If **rule-based-access-control**, **rule-and-basic-access-control**, or **rule-and-simple-access-control** is in effect for the entry being compared, the following access controls apply:

1) if entry level access is denied under **rule-based-access-control**, then the operation fails with **nameError** with problem **noSuchObject** in accordance with 7.11.2.4;
2) if access to the entry is not permitted under the **basic-access-control** scheme as described in 9.2.5 item 1), then the operation fails in accordance with 7.11.1.3;
3) if access is not granted to the attribute value being compared, the Directory shall act as though the attribute value was not present;
4) **basic-access-control** is applied as described in 9.2.5 items 2) and 3);
5) the name returned in the operation result is determined as defined in 7.11.2.2.

9.3 Abandon

Operations that interrogate the Directory may be abandoned using the **abandon** operation if the user is no longer interested in the result. The arguments of the operation may be signed (see 17.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2) by the requestor. If so requested, the Directory may sign the result.

```
abandon OPERATION ::= {
        ARGUMENT AbandonArgument
        RESULT          AbandonResult
        ERRORS          { abandonFailed }
        CODE        id-opcode-abandon }

AbandonArgument ::= OPTIONALLY-PROTECTED-SEQ {
        SEQUENCE {
            invokeID            [0]     InvokeId } }

AbandonResult ::= CHOICE {
        null            NULL,
        information     OPTIONALLY-PROTECTED-SEQ {
            SEQUENCE {
                invokeID                    InvokeId,
                COMPONENTS OF   CommonResultsSeq } } }
```

There is a single argument, the **invokeID** which identifies the operation that is to be abandoned. The value of the **invokeID** is the same **invokeID** that was used to invoke the operation that is to be abandoned.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Should the request succeed, a result shall be returned. If this result is to be signed by the Directory, the **SecurityParameters** (see 7.10) component of **CommonResultsSeq** (see 7.4) shall be included in the results. If the result of the operation is not to be signed by the Directory, no information shall be conveyed with the result. The original operation shall fail with an **abandoned** error.

Should the request fail, the **abandonFailed** error shall be reported. As a local matter, a DSA may choose not to abandon the operation and shall then return the **abandonFailed** error. This error is described in 12.3.

Abandon is only applicable to interrogation operations, i.e., Read, Compare, List, and Search operations.

A DSA may abandon an operation locally. If the DSA has chained or multicasted the operation to other DSAs, it may in turn request them to abandon the operation.

10 Directory Search operations

There are two 'search-like' operations: List and Search, defined in 10.1 and 10.2 respectively.

10.1 List

10.1.1 List syntax

A List operation is used to obtain a list of the immediate subordinates of an explicitly identified entry. Under some circumstances, the list returned may be incomplete. The arguments of the operation may be signed (see 17.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2) by the requestor. If so requested, the Directory may sign the result.

```
list    OPERATION ::= {
        ARGUMENT ListArgument
        RESULT          ListResult
        ERRORS          { nameError | serviceError | referral | abandoned | securityError }
        CODE        id-opcode-list }

ListArgument ::= OPTIONALLY-PROTECTED {
        SET {
            object              [0]     Name,
            pagedResults        [1]     PagedResultsRequest OPTIONAL,
            listFamily                  [2]     BOOLEAN DEFAULT FALSE,
            COMPONENTS OF               CommonArguments } }

ListResult ::= OPTIONALLY-PROTECTED {
        CHOICE {
            listInfo                    SET {
                name                        Name OPTIONAL,
                subordinates                [1]     SET OF SEQUENCE {
                    rdn                             RelativeDistinguishedName,
                    aliasEntry                  [0]     BOOLEAN DEFAULT FALSE,
                    fromEntry                   [1]     BOOLEAN DEFAULT TRUE },
                partialOutcomeQualifier     [2]     PartialOutcomeQualifier OPTIONAL,
                COMPONENTS OF                       CommonResults },
            uncorrelatedListInfo  [0]       SET OF ListResult } }
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
PartialOutcomeQualifier ::= SET {
        limitProblem                    [0]     LimitProblem  OPTIONAL,
        unexplored                      [1]     SET SIZE (1..MAX) OF ContinuationReference  OPTIONAL,
        unavailableCriticalExtensions   [2]     BOOLEAN DEFAULT  FALSE,
        unknownErrors                   [3]     SET SIZE (1..MAX) OF ABSTRACT-SYNTAX.&Type
OPTIONAL,
        queryReference                  [4]     OCTET STRING  OPTIONAL,
        overspecFilter                  [5]     Filter OPTIONAL,
        notification                    [6]     SEQUENCE SIZE (1 .. MAX) OF
                                                        Attribute{{SupportedAttributes}}  OPTIONAL,
        entryCount                              CHOICE {
            bestEstimate                [7]     INTEGER,
            lowEstimate                 [8]     INTEGER,
            exact                       [9]     INTEGER } OPTIONAL,
        streamedResult                  [10]    BOOLEAN DEFAULT  FALSE }

LimitProblem ::= INTEGER {
        timeLimitExceeded (0), sizeLimitExceeded (1), administrativeLimitExceeded (2) }
```

10.1.2 List arguments

The **object** argument identifies the object entry (or possibly the root) whose immediate subordinates are to be listed. Should the **Name** involve one or more aliases, they are dereferenced (unless prohibited by the relevant service control). The **Name** may be an alternative name and may include context information, as described in 9.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2.

The **pagedResults** argument is used to request that results of the operation be returned page-by-page, as described in 7.9.

If **listFamily** is **TRUE** and the **object** is an ancestor, the listed subordinates are taken from immediately subordinate family members; no other subordinates are included. Otherwise, the listed subordinates are taken only from immediately subordinate entries that are not family members.

The **CommonArguments** (see 7.3) include a specification of the service controls applying to the request. If the argument of this operation is to be signed by the requestor, the **SecurityParameters** (see 7.10) component shall be included in the arguments.

10.1.3 List results

The request succeeds, subject to access controls, if the **object** is located, regardless of whether there is any subordinate information to return.

The **name** is the distinguished name of the entry or an alias name of the entry, as described in 7.7. It is present only if an alias has been dereferenced, RDNs have been resolved to primary RDNs, or context selection has been applied and the name to be returned differs from the **object** name supplied in the operation argument.

The **subordinates** parameter conveys the information on the immediate subordinates, if any, of the named entry. Should any of the subordinate entries be aliases, they shall not be dereferenced.

The **rdn** parameter is the relative distinguished name of the subordinate. This may be affected by contexts as described for **Name** in 7.7.

The **fromEntry** parameter indicates whether the information was obtained from the entry (**TRUE**) or a copy of the entry (**FALSE**).

The **aliasEntry** parameter indicates whether the subordinate entry is an alias entry (**TRUE**) or not (**FALSE**).

The **partialOutcomeQualifier** consists of nine subcomponents as described below. This parameter shall be present whenever the result is incomplete because of a time limit, size limit, or administrative limit problem, because regions of

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

the DIT were not explored, because some critical extensions were unavailable, because an unknown error was received, because paged results are being returned, an overspecified filter is to be indicated, one or more notification attributes are to be returned, or the result of an operation is a streamed result and this response is not the last response of the result:

a) The **LimitProblem** parameter indicates whether the time limit, the size limit, or an administrative limit has been exceeded. The results being returned are those which were available when the limit was reached.

b) The **unexplored** parameter shall be present if regions of the DIT were not explored. Its information allows the DUA to continue the processing of the List operation by contacting other access points if it so chooses. The parameter consists of a set (possibly empty) of **ContinuationReferences**, each consisting of the name of a base object from which the operation may be progressed, an appropriate value of **OperationProgress**, and a set of access points from which the request may be further progressed. The **ContinuationReferences** that are returned shall be within the scope of referral requested in the operation service control. See 12.6.

c) The **unavailableCriticalExtensions** parameter indicates, if present, that one or more critical extensions were unavailable in some part of the Directory.

d) The **unknownErrors** parameter is used to return unknown error types or parameters received from other DSAs in the processing of the operation. Each member of the **SET** contains one such unknown error. See 12.2.4 of ITU-T Rec. X.519 | ISO/IEC 9594-5.

e) The **queryReference** parameter shall be present when the DUA has requested paged results and the DSA has not returned all the available results. See 7.9. It shall be absent when the DSA can determine that all results valid for the user have been returned (i.e., other than as a result of applying access control).

f) The **overspecFilter** component is only used in conjunction with the Search operation when, as a consequence of over-specified filtering, the returned Search result is empty, although there are candidate entries either matching only portions of the filter or matching only approximately the filter. It is returned only if the search request included the **checkOverspecified** item and the Directory can determine that the filter was over-specified. It consists of the filter supplied in the **search** argument with those elements of the filter that succeeded in matching some entries omitted. The actual procedure for generating the **overspecFilter** is a local matter.

NOTE 1 – The return of a suitable **overspecFilter** in a distributed Directory is for further study.

g) The **notification** parameter may be used to send qualifications of error outcomes, and may also for the Search operation be used to return a **proposedRelaxation** attribute (see 6.13.15 of ITU-T Rec. X.520 | ISO/IEC 9594-6) which provides a relaxation policy that could be applied by the user. In this case, the sequence of **MRMapping** elements that would have been used to affect the relaxation (or tightening) policy specified by the relevant search-rule may be supplied.

NOTE 2 – The ordering of **sequence-of Attribute** in **notification** is not significant.

h) The **entryCount** parameter is only relevant in **search** results and if then present, it gives a best estimate of the number of entries that fulfil the search criteria. This subcomponent shall be present if, and only if,

- **entryCount** search control option is set in the search argument or by a governing-search-rule;
- if paged results have been requested or a size limit has been exceeded; and
- if the feature is supported by at least one of the participating DSAs.

When the **entryCount** subcomponent is present, the **bestEstimate** or exact choice shall be taken if all performing DSAs support the feature and if all the eligible DSAs have participated in the operation. The exact choice shall be taken if all participating DSAs can supply an exact count, otherwise, the **bestEstimate** choice shall be taken. If not all the eligible DSAs participated in the operation or if some of the participating DSA do not support the **entryCount** parameter, the **lowEstimate** choice shall be taken. Family members of a compound entry only count as a single entry.

i) The **streamedResult** parameter indicates, when present and **TRUE**, that the DSA is sending a streamed result and that this response is not the last response of the result. If absent or present with the value FALSE, this parameter indicates that this response is the final response of a streamed result or that it is a non-streamed response. Each of the responses in a streamed result shall be identified with the same **invokeId**.

If a limit problem is encountered which results in a **limitProblem** element being used in **PartialOutcomeQualifier**, this component shall be repeated in all subsequent results supplied as part of the paged result set.

NOTE 3 – Each of the responses in a streamed result shall be identified with the same **invokeID**. As such, this option is only available with the IDM Directory protocols as specified in ITU-T Rec. X.519 | ISO/IEC 9594-5.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

When the DUA has requested a protection request of signed, or if the Directory for other reasons is not able to correlate information, the **uncorrelatedListInfo** parameter may comprise a number of sets of result parameters originating from and signed by different components of the Directory. If no DSA in the chain can correlate all the results, the DUA shall assemble the actual result from the various pieces.

The **CommonResults** (see 7.4) include the security parameters applying to the response. If this result is to be signed by the Directory, the **SecurityParameters** (see 7.10) component shall be included in the results.

10.1.4 List errors

Should the request fail, one of the listed errors shall be reported. The circumstances under which the particular errors shall be reported are defined in clause 12.

10.1.5 List operation decision points for basic access control

If **rule-based-access-control** is also applied, the order in which it is applied with respect to **basic-access-control** is a local matter, except that if access is denied to the entry, an attribute type or an attribute value by either mechanism, it shall not be overridden by the other mechanism. In this respect, *DiscloseOnError* permission of **basic-access-control** is a permission that shall not override a deny of **rule-based-access-control**.

If **basic-access-control** is in effect for the portion of the DIB where the **list** operation is being performed, the following sequence of access controls applies:

1) No specific permission is required to the entry identified by the **object** argument.
2) For each immediate subordinate for which a **RelativeDistinguishedName** is to be returned in **subordinates**, *Browse* and *ReturnDN* permissions are required to that entry. Entries for which these permissions are not granted are ignored. If as a consequence of applying these controls, no subordinate information (excluding any **ContinuationReferences** in **PartialOutcomeQualifier**) is returned and if *DiscloseOnError* permission is not granted to the entry identified by the **object** argument, the operation fails and a **nameError** with problem **noSuchObject** shall be returned. The **matched** element shall either contain the name of the next superior entry to which *DiscloseOnError* permission is granted, or the name of the DIT root (i.e., an empty **RDNSequence**). Otherwise, the operation succeeds but no subordinate information (excluding any **ContinuationReferences** in **PartialOutcomeQualifier**) is conveyed with it.

NOTE 1 – In the case of a **nameError** being returned, the empty **RDNSequence** may be used by a DSA which does not have access to all superior entries.

NOTE 2 – Security policy may prevent the disclosure of subordinate information which would otherwise be conveyed as **ContinuationReferences** in **PartialOutcomeQualifier**. If such a policy is in effect and if a DUA constrains the service by specifying **chainingProhibited**, the Directory may return a **serviceError** with problem **chainingRequired**. Otherwise, the procedure described in item 2) above is followed.

NOTE 3 – Security policy may prevent the Directory from indicating that a listed subordinate entry is an alias entry. For example, if the DUA is not granted *Read* access to the alias entry, its **objectClass** attribute and the value **alias** that it contains, the Directory may omit the **aliasEntry** component of **subordinates** from the **ListResult** or set it to **FALSE**.

NOTE 4 – If *DiscloseOnError* permission is not granted to the entry identified by the **object** argument, a **partialOutcomeQualifier** indicating a **limitProblem** or **unavailableCriticalExtensions** should not be returned as it may compromise the security of this entry.

10.1.6 List operation decision points for rule-based access control

If **basic-access-control** is also applied, the order in which it is applied with respect to **rule-based-access-control** is a local matter, except that if access is denied to the entry, an attribute type or an attribute value by either mechanism, it shall not be overridden by the other mechanism. In this respect, *DiscloseOnError* permission of **basic-access-control** is a permission that shall not override a deny of **rule-based-access-control**.

If **rule-based-access-control**, **rule-and-basic-access-control**, or **rule-and-simple-access-control** is in effect for the portion of the DIB where the **List** operation is being performed, the following access controls apply:

1) If rule-based entry level permission is denied to the entry identified by the **object** argument, then **nameError** with problem **noSuchObject** is returned in accordance with 7.11.2.4.
2) For each immediate subordinate for which a **RelativeDistinguishedName** is to be returned in **subordinates**, rule-based RDN permission must be granted to that entry. Entries for which access is not granted are ignored.
3) **basic-access-control** is applied as described in 10.1.5.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

10.2 Search

10.2.1 Search syntax

A Search operation is used to search one or more portions of the Directory for entries of interest, and to return selected information from those entries. The arguments of the operation may be signed (see 17.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2) by the requestor. If so requested, the Directory may sign the result.

```
search OPERATION ::= {
        ARGUMENT SearchArgument
        RESULT          SearchResult
        ERRORS          { attributeError | nameError | serviceError | referral | abandoned |
                        securityError }
        CODE        id-opcode-search }

SearchArgument ::= OPTIONALLY-PROTECTED {
        SET {
                baseObject          [0]     Name,
                subset              [1]     INTEGER {
                        baseObject(0), oneLevel(1), wholeSubtree(2) } DEFAULT baseObject,
                filter                      [2]     Filter DEFAULT and : { },
                searchAliases       [3]     BOOLEAN DEFAULT TRUE,
                selection                   [4]     EntryInformationSelection DEFAULT { },
                pagedResults        [5]     PagedResultsRequest OPTIONAL,
                matchedValuesOnly   [6]     BOOLEAN DEFAULT FALSE,
                extendedFilter      [7]     Filter OPTIONAL,
                checkOverspecified  [8]     BOOLEAN DEFAULT FALSE,
                relaxation                  [9]     RelaxationPolicy            OPTIONAL,
                extendedArea        [10]    INTEGER                     OPTIONAL,
                hierarchySelections [11]    HierarchySelections DEFAULT { self },
                searchControlOptions        [12]    SearchControlOptions DEFAULT { searchAliases },
                joinArguments               [13]    SEQUENCE SIZE (1..MAX) OF JoinArgument OPTIONAL,
                joinType                    [14]    ENUMERATED {
                        innerJoin(0), leftOuterJoin(1), fullOuterJoin(2) } DEFAULT leftOuterJoin,
                COMPONENTS OF       CommonArguments } }

HierarchySelections ::= BIT STRING {
        self            (0),
        children        (1),
        parent      (2),
        hierarchy       (3),
        top             (4),
        subtree         (5),
        siblings        (6),
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        siblingChildren     (7),
        siblingSubtree (8),
        all                 (9) }

SearchControlOptions ::= BIT STRING {
        searchAliases               (0),
        matchedValuesOnly           (1),
        checkOverspecified          (2),
        performExactly                  (3),
        includeAllAreas                 (4),
        noSystemRelaxation          (5),
        dnAttribute                     (6),
        matchOnResidualName             (7),
        entryCount                      (8),
        useSubset                       (9),
        separateFamilyMembers       (10),
        searchFamily                (11) }

JoinArgument ::= SEQUENCE {
        joinBaseObject          [0]     Name,
        domainLocalID           [1]     DomainLocalID OPTIONAL,
        joinSubset              [2]     ENUMERATED {
                baseObject(0), oneLevel(1), wholeSubtree(2) } DEFAULT baseObject,
        joinFilter              [3]     Filter  OPTIONAL,
        joinAttributes      [4]     SEQUENCE SIZE (1..MAX) OF JoinAttPair OPTIONAL,
        joinSelection       [5]     EntryInformationSelection }

DomainLocalID ::= UnboundedDirectoryString
```

DomainLocalID **is a string which locally uniquely identifies a remote domain holding a portion of another DIT.**

NOTE – This string is defined locally and does not need to be registered by any registration authority.

```
JoinAttPair ::= SEQUENCE {
        baseAtt             AttributeType,
        joinAtt         AttributeType,
        joinContext         SEQUENCE SIZE (1..MAX) OF JoinContextType OPTIONAL }

JoinContextType ::= CONTEXT.&id({SupportedContexts})

SearchResult ::= OPTIONALLY-PROTECTED {
        CHOICE {
            searchInfo                  SET {
                name                        Name  OPTIONAL,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        entries                    [0]    SET OF EntryInformation,
        partialOutcomeQualifier    [2]    PartialOutcomeQualifier OPTIONAL,
        altMatching                [3]    BOOLEAN  DEFAULT FALSE,
        COMPONENTS OF                     CommonResults },
    uncorrelatedSearchInfo         [0]    SET OF SearchResult } }
```

10.2.2 Search arguments

The **baseObject** argument identifies the object entry (or possibly the root) relative to which the primary search is to take place. The **baseObject** may be an alternative name and may include context information, as described in 9.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2.

The **subset** argument indicates whether the primary search is to be applied to:

a) the **baseObject** only;

b) the immediate subordinates of the base object only (**oneLevel**);

c) the base object and all its subordinates (**wholeSubtree**).

If the base object is an ordinary entry, compound entries shall be counted as a single entry with respect to the **subset** specification. If the base object is the ancestor of a compound entry, the **searchFamily** search control option controls the exact behaviour. If the base object is a child family member, family members shall count as individual entries.

The **filter** argument is used to eliminate entries from the primary search space which are not of interest. Information shall only be returned on entries which satisfy the filter (see 7.8). In the presence of a basic user-supplied or search-rule-supplied relaxation policy, the filter shall be evaluated for the first time with the required substitutions of matching rules.

In the presence of a user-supplied or a search-rule-supplied relaxation policy, or both, the return of fewer results than minimum shall cause a re-evaluation of the filter, using the appropriate relaxations (see 7.8 and also below, for the relaxation element of **SearchArgument**), progressively until there are enough entries or no more relaxations are defined. Similarly, the return of more results than the maximum shall cause a re-evaluation of the filter, using the appropriate tightenings, progressively until there are few enough entries or no more tightenings are defined.

NOTE 1 – If no search-rule relaxations are provided, the user may need to simplify the filter and try again, or alternatively to define a user defined relaxation.

The **familyGrouping** component of **CommonArguments** is used to logically merge together entries in a family prior to applying the filter, as described in 7.3.2 and 7.8.3.

Aliases shall be dereferenced while locating the base object, subject to the setting of the **dontDereferenceAliases** service control. Aliases among the subordinates of the base object shall be dereferenced during the search, subject to the setting of the **searchAliases** parameter. If the **searchAliases** parameter is **TRUE**, aliases shall be dereferenced, if the parameter is **FALSE**, aliases shall not be dereferenced. If the **searchAliases** parameter is **TRUE**, the search shall continue in the subtree of the aliased entry.

The **selection** argument indicates what information from the entries is requested (see 7.6). However, it should not be assumed that the attributes returned are the same as or limited to those requested.

NOTE 2 – A DSA that is coordinating distributed operations for related entries (i.e., has finished name resolution for a Search argument containing **joinArguments** and needs to acquire a collection of potentially related entries from non-internal sources) needs to override the DAP-supplied **infoTypes** value with **attributeTypesAndValues** for the purposes of distributed operations, and needs also to include join attributes (i.e., attributes in the set specified by **JoinAttPair.joinAtt** within **JoinArgument.joinAttributes**) in the selection of attributes to be returned using distributed operations. However, entries and derived entries that are returned to the user by the coordinating DSA shall omit attribute values in the DAP-returned information if the **infoTypes** value was **attributeTypesOnly**, and shall thus return **EntryInformation** in accordance with the original user request.

The **pagedResults** argument is used to request that results of the operation be returned page-by-page, as described in 7.9.

The **matchedValuesOnly** argument indicates that certain attribute values are to be omitted from the returned entry information. Specifically, where an attribute to be returned is multi-valued, and some but not all of the values of that attribute contributed to the search filter, in its last effective form (i.e., taking relaxed matching rules into account) returning TRUE via filter items other than **present**, then the values that did not so contribute are omitted from the returned entry information.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

If the **matchedValuesOnly** argument is specified in the **search** argument, the following logic processing applies to the attributes to be returned:

a) If the filter consists of one filter item, the following rules apply:
   - if the type of the filter item is **present**, then the **matchedValuesOnly** argument has no effect on the attribute in this filter item.
   - If the type of the filter item is **equality**, **substrings**, **greaterOrEqual**, **lessOrEqual**, **approximateMatch**, **contextPresent** or **extensibleMatch** and the assertion is not TRUE for the attribute, then the **matchedValuesOnly** argument has no effect on this attribute. If the assertion is TRUE, then the values of this attribute that did not match the filter item are omitted from the returned entry information.
   - If the filter item is negated, then the **matchedValuesOnly** argument has no effect on this attribute.

b) If the filter is complex (consists of more than one filter item), then the following rules apply:
   - If the filter contains a negated (i.e., **not**) filter, then the **matchedValuesOnly** argument has no effect on any attribute within the negated filter.

   NOTE 3 – This applies to nested negated filters as well.
   - The **matchedValuesOnly** argument has no effect on the attributes of any elements of **or** filters that evaluate to FALSE or UNDEFINED.
   - An attribute that occurs multiple times in the filter only needs one of its occurrences to evaluate to TRUE as described in a), second bullet above, for the **matchedValuesOnly** argument to be effective, i.e., one instance of effectiveness overrides one or more instances of ignore.
   - Each filter in an **or** filter should be evaluated for **matchedValuesOnly**, even if the truth of the filter can be determined before full evaluation is complete.

The **extendedFilter** argument is used in mixed version environments to specify an alternative filter to that described above. When this argument is present, the **filter** argument (if any) shall be ignored by second and subsequent edition systems. The **extendedFilter** is always ignored by first edition systems. Search relaxation is applied just as for **filter**.

NOTE 4 – By including both filters, a DUA can specify one filter to be used by first edition systems and a different filter to be used by second and subsequent edition systems in the distributed processing of the Search request. first edition systems do not support attribute polymorphism or matching rule assertions.

The **checkOverspecified** argument is used to request the Directory to return an **overspecFilter** item in **partialOutcomeQualifier** if the result of the search operation is empty and the Directory is able to determine that this is due to the filter being overspecified.

The **relaxation** component may be used to specify a user-supplied **RelaxationPolicy** using the construct defined in 16.10 of ITU-T Rec. X.501 | ISO/IEC 9594-2.

Substitutions specified by a **search** request shall not be performed within a service specific administrative area if the substitution causes the search to be invalid with respect the governing-search-rule. The governing-search-rule can be violated when the substituting matching rule:

a) effectively removes one or more filter items from the **search** filter; or

b) violates the **matchingUse** specification for the attribute type (see 16.10.2 of ITU-T Rec. X.501 | ISO/IEC 9594-2).

NOTE 5 – The **nullMatch** matching rule has the effect of removing one or more filter items from the filter. When using this matching rule, the governing-search-rule might be violated.

If the Search operation is performed outside a service specific administrative area or if the governing-search-rule does not provide a **RelaxationPolicy** component, the user-supplied **RelaxationPolicy** is applied as described in 16.10.7 of ITU-T Rec. X.501 | ISO/IEC 9594-2. When a search-rule-provided **RelaxationPolicy** is also present, the combination is applied according to the following procedure:

1) The search-rule specified basic substitution policy, if any, is applied already during the search-validation process. The possible basic substitutions specified by the governing-search-rule are thereby applied *a priori*.

2) The basic substitutions and the mapping-based mapping specified in the **search** request, if present, shall then be applied. However, basic substitutions that cause the governing-search-rule to be violated shall not be applied, but shall be ignored. The **oldMatchingRule** value (if supplied) in this case applies to the basic matching rule, i.e., the one that would have been applicable in the absence of a search-rule-applied basic substitution policy.

3) The relaxation/tightening substitutions, if any, specified in the **search** request are then applied together with any specified mapping-based matching following the rules defined in 16.10.7 of ITU-T Rec. X.501 |

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

ISO/IEC 9594-2. If substituting matching rule is encountered at any point that caused non-conformance with the governing-search-rule, this particular substitution is abandoned completely, together with any further substitutions that may have been specified by the **search** request for that attribute type. If during the process, the **minimum** or **maximum** specification specified in the **search** request is satisfied, the process stops.

4) The governing search-rule-supplied relaxation or tightening substitutions are applied, with the exception there shall be no substitution for attribute types for which relaxation or tightening substitutions has been performed. That is, further relaxation or tightening substitutions apply only to matching rules for attribute types that so far have not been exposed to relaxation or tightening substitution. In this part of the process, the **maximum** or **minimum** specifications in the **search** request still apply, rather than those in specified in the governing-search-rule.

If a substitution specified in the **search** request proposes an unsupported matching rule, the existing matching rule stays in place. If this strategy fails to produce a supported matching rule, the filter-item is evaluated as UNDEFINED.

The user can propose that the system provides a relaxation or tightening by specifying the dummy matching rule **systemProposedMatch**.

The **extendedArea** component indicates the level of relaxation (if greater than zero) or the level of tightening (if less than zero). If this component is present, it affects relaxation or tightening, as it is described in 16.10.7 of ITU-T Rec. X.501 | ISO/IEC 9594-2.

The **hierarchySelection** search control specifies by means of a bit string the hierarchical selection to be performed within a hierarchical group with respect to each matched entry. It is ignored for matched entries that are not part of a hierarchical group. If several entries within a hierarchy are matched, hierarchical selection will not result in the same entry being returned more than once. If this search control is not present, no hierarchical selection is performed. When present, the following choices are possible either alone or in combinations:

a) **self** indicates that entry information shall be returned from the matched entries. If this is the only choice, it corresponds to performing no hierarchical selection.

b) **children** indicates that for each matched entry, the entry information is returned from all immediately hierarchical children, if any, of each matched entry. No information is returned from the matched entry if this is the only setting.

c) **parent** indicates that for each matched entry, the entry information is returned from the immediately hierarchical parent, if any, of each matched entry. No information is returned from the matched entry if this is the only setting.

d) **hierarchy** indicates that for each matched entry, the entry information is returned from all the hierarchical parents. No information is returned from the matched entry if this is the only setting.

e) **top** indicates that for each matched entry, the entry information from the hierarchical top is returned. No information is returned from the matched entry if this is the only setting, unless the matched entry is the top entry.

f) **subtree** indicates that for each matched entry, the entry information is returned from all its hierarchical children, if any. No information is returned from the matched entry if this is the only setting.

g) **siblings** indicates that for each matched entry, the entry information from all hierarchical siblings is returned. No information is returned from the matched entry if this is the only setting.

h) **siblingChildren** indicates that for each matched entry, the entry information from the immediately hierarchical children of all hierarchical siblings is returned. No information is returned from the matched entry and its siblings if this is the only setting.

i) **siblingSubtree** indicates that for each matched entry, the entry information from all the children of all hierarchical siblings is returned. No information is returned from the matched entry and its siblings if this is the only setting.

j) **all** indicates that for each matched entry, the entry information from all entries of the hierarchical group is returned.

The **searchControlOptions** component contains only control options applicable for the Search operation. This component has indicators with the same semantics as the boolean type components of the search argument. An implementation supporting the service administration extension shall support this component. A sending supporting implementation (e.g., a DUA) shall in addition to setting the boolean type components also set the corresponding bits in this component (unless defaults apply). If a supporting DSA implementation receives a **search** request with this

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

component, it shall ignore the boolean type components in the request. If this component is absent in a request, the default setting shall be understood to be all bits reset, except as indicated below:

a) The **searchAliases** search control option is a replacement for the **searchAliases** search argument component. If this bit is set, it corresponds to the **searchAliases** component being **TRUE**. If the **searchControlOptions** component is absent, the default value is according to the **searchAliases** component, i.e., if the **searchAliases** component is absent or set to **TRUE**, this bit defaults to being set.

b) The **matchedValuesOnly** search control option is a replacement for the **matchedValuesOnly** search argument component. If this bit is set, it corresponds to the **matchedValuesOnly** component being **TRUE**. If the **searchControlOptions** component is absent, the default value is according to the **matchedValuesOnly** component, i.e., if the **matchedValuesOnly** component is set to **TRUE**, this bit defaults to being set; otherwise it defaults to being reset.

c) The **checkOverspecified** search control option is a replacement for the **checkOverspecified** search argument component. If this bit is set, it corresponds to the **checkOverspecified** component being **TRUE**. If the **searchControlOptions** component is absent, the default value is according to the **checkOverspecified** component, i.e., if the **checkOverspecified** component is set to **TRUE**, this bit defaults to being set; otherwise it defaults to being reset.

d) The **performExactly** search control option indicates that an operation shall be performed exactly according to the relevant matching rules as specified or implied by the filter after a basic matching rule substitution, if applicable. When an **extensibleMatch** filter item specifies an unsupported matching rule, the **search** request shall be rejected when this search control option is set. Otherwise, the filter item evaluates to UNDEFINED. If the Search operation starts its initial evaluation phase within a service specific administrative area and a matching restriction in a search-rule is violated, that search-rule will fail the search-validation if, and only if, this search control option is set.

e) The **includeAllAreas** search control option is only relevant if the **extendedArea** component is included with a value of zero or greater. In all other cases, it is ignored. If the value is **TRUE**, inclusive relaxation is performed; otherwise exclusive relaxation is performed if possible (see 13.6 of ITU-T Rec. X.501 | ISO/IEC 9594-2).

f) The **noSystemRelaxation** search control option is used when the user requires that DSA-supplied relaxation policies shall not be applied. The DSA still applies a basic policy, unless there is a user-supplied basic policy that overrides it, but no subsequent relaxations or tightenings shall be applied. That is, the filter is never evaluated more than once over the set of candidate entries, except because of user-supplied relaxations.

g) The **dnAttribute** search control option is used to indicate that the attributes of the Distinguished Name of an entry are used in addition to those of the entry when evaluating filter against the entry. If set, it overrides any possible **dnAttribute** specification in **extensibleMatch** filter items. It also applies to all filter item types.

h) The **matchOnResidualName** search control option is only relevant if the **partialNameResolution** service control option is set. It is used to indicate that if the Directory is able to resolve only part of the purported name in a **search** operation, the AVAs of the unresolved RDNs shall be treated as AND'ed **equality** filter items. These filter items are AND'ed with the search filter both for search evaluation against search-rules and for entry matching.

i) The **entryCount** search control option indicates that an entry count shall be supplied in the **search** result in case either a service control size limit or an administrative size limit has been exceeded. The **entryCount** gives an indication of how many entries would have been returned had a size limit not been encountered. This search control is ignored if the **subentries** service control option is set.

j) The **useSubset** search control option indicates that the **imposedSubset** search-rule component shall be ignored (see 16.10.9 of ITU-T Rec. X.501 | ISO/IEC 9594-2).

k) The **separateFamilyMembers** search control option indicates that family members are returned as separate entries rather than being embedded in the **family-information** derived attribute.

l) The **searchFamily** search control option specifies how the search is performed if the base object is the ancestor of a compound attribute. This option is ignored if the base object is not an ancestor or if the **entryOnly** is set in either the **CommonArguments** or in the **ChainingArguments**. If this option is set, the operation is only performed on the compound entry and each family member count as a separate entry with respect to the **subset** and **sizeLimit** specifications. If **searchFamily** option is not set, the compound entry is considered a single entry with respect to the **subset** specification.

NOTE 6 – The latter implies that if, as an example, **subset** is set to **baseObject** and **familyGrouping** is **entryOnly**, then each individual family member is within the scope of the search.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The **joinArguments** argument is used to specify additional portions of the Directory to be searched for the purpose of identifying and accessing entries related to those in the primary search and to specify the attributes to be used in joining the related entries. Although specified as a SEQUENCE, the order in which **joinArgument** arguments appear is not significant.

NOTE 7 – When **joinArguments** is specified, the primary search and each additional search is considered to produce a set of intermediate results. Each set of intermediate results resulting from a specification of **joinArgument** will be joined with the results of the primary search, and all joins will be performed prior to returning any results in **SearchResult**. Intermediate results are not visible to users of the directory.

The **joinBaseObject** argument identifies the object entry (or possibly the root) relative to which each additional search is to take place. The **joinBaseObject** may be an alternative name and may include context information, as described in 9.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2.

The **domainLocalID** argument optionally specifies a separate DIT in which the search for **joinBaseObject** is to be initiated. If absent, the search for **joinBaseObject** is to be initiated in all DITs known to the DSA.

The **joinSubset** argument indicates whether the additional search is to be applied to:

a) the **joinBaseObject** only;

b) the immediate subordinates of the join base object only (**oneLevel**);

c) the join base object and all its subordinates (**wholeSubtree**).

The **joinFilter** argument is used to eliminate entries from the additional search space which are not of interest. Only information which satisfies the **joinFilter** will be considered for joining with related entries. If **joinFilter** is not specified, the value in the **filter** component of the **SearchArgument** will be used. If the **filter** component of the **SearchArgument** is not supplied, the default value for that component will be used. When present, **joinFilter** will be treated according to the rules for **extendedFilter**.

The **joinAttributes** argument is used to specify the pairs of attributes to be used in joining entries from the primary search with entries from an additional search. An entry from the primary search (the "primary entry") is considered related to an entry from an additional search (the "additional entry") if there exists a **joinAttrPair** such that the following conditions are true:

a) the primary entry has a value for the attribute type specified by **baseAtt**;

b) the additional entry has a value for the attribute type specified by **joinAtt**;

c) one of the attribute values in the primary entry and one of the attribute values in the additional entry are equal according to the following rules:

   i) if the attribute types are the same, the equality matching rule for that attribute type is applied;

   ii) if the attribute types are not the same, but have the same syntax, the equality matching rule for the attribute type specified for the primary entry is applied;

   iii) if **joinContexts** is present, only attribute values of the specified contexts may be used in evaluating in accordance with rule i) or ii) above. If **joinContexts** is absent, attribute values of all contexts may be used in evaluating in accordance with rule i) or ii) above.

In evaluating **joinAttributes** for potential joins, subtypes of the join attributes shall be ignored. Only the explicitly-identified **baseAtt** and **joinAtt** shall be used to evaluate a potential join.

If an equality rule is applied and evaluates to either FALSE or UNDEFINED, the entries are not considered to be related.

If no suitable matching rule can be applied under condition c) above, the entries are not considered to be related.

NOTE 8 – Care should be taken to prevent the unintentional retrieval of meaningless data when specifying joins involving multi-valued attributes. For example, if an entry uses a multi-valued attribute such as an employee identifier to denote membership in a committee, specification of that multi-valued attribute in performing the join could result in the return of an uncorrelated set containing group member names, telephone numbers, electronic mail addresses, and so on. Nevertheless, when outer joins are specified, all entries that are retrieved will be returned, even if not related.

The **joinSelection** argument is used to eliminate attributes from the additional search intermediate result which are not of interest.

The **joinType** argument is used to specify the type of join to be performed on related entries, as follows:

a) If **innerJoin** is specified, the resulting entry set will include only those entries for which a join has been performed based on the attribute pairs specified in **joinAttributes**. Each resulting entry will include all the corresponding related entries as **relatedEntry** attribute values.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

b) If **leftOuterJoin** is specified, the resulting entry set will include all entries selected by the primary search; all entries for which a join has been performed based on the attribute pairs specified in **joinAttributes** will include all the corresponding related entries as **relatedEntry** attribute values.

c) If **fullOuterJoin** is specified, the resulting entry set will include all entries from the primary and additional searches; all entries for which a join has been performed based on the attribute pairs specified in **joinAttributes** will include all the corresponding related entries as **relatedEntry** attribute values rather than as explicit entries.

No attempt at joining shall take place unless the **joinAttributes** value contains at least one **JoinAttPair** and each **JoinAttPair** is valid in terms of matching rules. If this is not the case, no attempt at joining shall take place, and the following shall be the outcome of merging for each **JoinAttPair**, depending on the join-type:

Join-type	Merged output
inner-join	empty
left-outer-join	primary results only
full-outer-join	results from primary and joined search

Otherwise, entries shall only be eligible for joining when they can supply all of the relevant join-attribute values.

The results of joining shall include all combinations of matched join attributes.

NOTE 9 – For example, consider A B and C as entries from the primary search and P, Q, R as entries from an additional search using J, a corresponding **JoinAttPair** value, and suppose that the following matches take place as a result of J:

– A with P, A with Q, A with R

– B with Q

– C with P and C with Q

Then, the joined results will include:

– A with {P,Q,R}

– B with {Q}

– C with {P,Q}

even though Q's results occur three times.

The **CommonArguments** (see 7.3) include a specification of the service controls and security parameters applying to the request. If the argument of this operation is to be signed by the requestor, the **SecurityParameters** (see 7.10) component shall be included in the arguments.

10.2.3 Search results

The request succeeds, subject to access controls, if the **baseObject** is located, regardless of whether there are any subordinates to return, and if there are no service restrictions as specified within a service specific administrative area that prevent the Search operation from proceeding.

NOTE 1 – As a corollary to this, the outcome of an unfiltered search applied to a single entry may not be identical to a read which seeks to interrogate the same set of attributes of the entry. This is because the latter shall return an **AttributeError** if none of the selected attributes exist in the entry.

The **name** is the distinguished name of the entry or an alias name of the entry, as described in 7.7. It is present only if an alias has been dereferenced, RDNs have been resolved to primary RDNs, or context selection has been applied and the name to be returned differs from the **baseObject** name supplied in the operation argument.

The **entries** parameter conveys the requested information from each entry (zero or more) which satisfied the filter (see 7.5). The names supplied as part of **entries** may be affected by contexts as described for **Name** in 7.7. The entry information may include family information, as required by the **familyReturn** element of **EntryInformationSelection**. The interaction between **familyGrouping** and **familyReturn** is defined in a four-phase evaluation of a filter and subsequent evaluation of what to return, as described in 7.8.3.

The **partialOutcomeQualifier** is as described in 10.1.3.

NOTE 2 – Where returned entry information is incomplete for a particular entry, it is indicated via the **incompleteEntry** parameter in the returned entry information.

The **altMatching** parameter indicates that a matching rule has not been applied exactly as specified in the **search** request.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The **appliedRelaxation** attribute in the **notifications** element of **CommonResults** shall be used to list the attributes of the filter which have been subject to relaxation or tightening, other than those made by the **basic** element of a relaxation policy (see 6.13.16 of ITU-T Rec. X.520 | ISO/IEC 9594-6).

The **uncorrelatedSearchInfo** parameter is as described for **uncorrelatedListInfo** in 10.1.3.

The **CommonResults** (see 7.4) include the security parameters applying to the response. If this result is to be signed by the Directory, the **SecurityParameters** (see 7.10) component shall be included in the results.

10.2.4 Service administration

An administrative authority may establish service specific administrative areas as specified in clause 7 of ITU-T Rec. X.501 | ISO/IEC 9594-2. This allows the administrative authority to administer the service by restricting the Search operation with respect to what areas of the DIT can be searched and what type of searches can be formed, what information can be returned, etc., by definition of search-rules.

10.2.5 Search errors

Should the request fail, one of the listed errors shall be reported. The circumstances under which the particular errors shall be reported are defined in clause 12.

When searches are performed within service specific administrative areas, a number of additional, quite detailed elements of error information may be returned as detailed in clause 13.

10.2.6 Search operation decision points for basic access control

If **rule-based-access-control** is also applied, the order in which it is applied with respect to **basic-access-control** is a local matter, except that if access is denied to the entry, an attribute type or an attribute value by either mechanism, it shall not be overridden by the other mechanism. In this respect, *DiscloseOnError* permission of **basic-access-control** is a permission that shall not override a deny of **rule-based-access-control**.

If **basic-access-control** is in effect for the portion of the DIT to be searched, the following sequence of access controls applies:

1) No specific permission is required to the entry identified by the **baseObject** argument.

NOTE 1 – If the **baseObject** is within the scope of the **SearchArgument** (i.e., when the **subset** argument specifies **baseObject** or **wholeSubtree**) the access controls specified in items 2) through 5) apply.

2) For each entry within the scope of the **SearchArgument** which is to be a candidate for consideration, *Browse* permission is required. Entries for which this permission is not granted are ignored.

3) The **filter** argument is applied to each entry left to be considered after taking item 2) into account, in accordance with the following:

   a) For each **FilterItem** that specifies an attribute, *FilterMatch* permission for the attribute type is required before the **FilterItem** can be evaluated as either TRUE or FALSE. A **FilterItem** for which this permission is not granted evaluates as UNDEFINED.

   b) For each **FilterItem** that additionally specifies an attribute value, *FilterMatch* permission is required for each stored attribute value which is to be considered for the purposes of matching. If there is a value that both matches the **FilterItem** and for which permission is granted, the **FilterItem** evaluates to TRUE, otherwise it evaluates to FALSE.

4) If present, the **joinCriteria** argument is applied to each entry left to be considered after taking item 3) into account, in accordance with the following:

   a) For each **JoinCriterialtem** which specifies an attribute, *FilterMatch* permission for the attribute type is required before the **JoinCriterialtem** can be evaluated as either TRUE or FALSE. A **JoinCriterialtem** for which this permission is not granted evaluates as UNDEFINED.

   b) For each **JoinCriterialtem** that additionally specifies an attribute value, *FilterMatch* permission is required for each stored attribute value which is to be considered for the purposes of matching. If there is a value that both matches the **JoinCriterialtem** and for which permission is granted, the **JoinCriterialtem** evaluates to TRUE, otherwise it evaluates to FALSE.

5) Once the procedures defined in 2) through 4) have been applied, the entry is either selected or discarded. If as a consequence of applying these controls to the entire scoped subtree no entries have been selected (excluding any **ContinuationReferences** in **partialOutcomeQualifier**) and if *DiscloseOnError* permission is not granted to the entry identified by the **baseObject** argument, the operation fails and a **nameError** with problem **noSuchObject** shall be returned. The **matched** element shall either contain the name of the next superior entry to which *DiscloseOnError* permission is granted, or the name of the DIT

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

root (i.e., an empty **RDNSequence**). Otherwise, the operation succeeds but no subordinate information is conveyed with it.

NOTE 2 – In the case of a **nameError** being returned, the empty **RDNSequence** may be used by a DSA which does not have access to all superior entries.

NOTE 3 – Security policy may prevent the disclosure of knowledge information which would otherwise be conveyed as **ContinuationReferences** in **partialOutcomeQualifier**. If such a policy is in effect and if a DUA constrains the service by specifying **chainingProhibited**, the Directory may return a **serviceError** with problem **chainingRequired**. Otherwise, the **ContinuationReference** is omitted from **partialOutcomeQualifier**.

6) Otherwise, for each selected entry the information returned is as follows:

   a) If the **infoTypes** element of **selection** specifies that attribute types only are to be returned, then for each attribute type that is to be returned, *Read* permission is required. If permission is not granted, the attribute type is omitted from **EntryInformation**. If as a consequence of applying these controls no attribute type information is selected, the **EntryInformation** element is returned but no attribute type information is conveyed with it (i.e., the **SET OF CHOICE** element is omitted or empty).

   b) If the **infoTypes** element of **selection** specifies that attribute types and values are to be returned, then for each attribute type and for each value that is to be returned, *Read* permission is required. If permission to an attribute type is not granted, the attribute is omitted from **EntryInformation**. If permission to an attribute value is not granted, the value is omitted from its corresponding attribute. In the event that permission is not granted to any of the values within the attribute, an **Attribute** element containing an empty **SET OF AttributeValue** is returned. If as a consequence of applying these controls no attribute information is selected, the **EntryInformation** element is returned but no attribute information is conveyed with it (i.e., the **SET OF CHOICE** element is omitted or empty).

NOTE 4 – If *DiscloseOnError* permission is not granted to the entry identified by the **baseObject** argument, a **partialOutcomeQualifier** indicating a **limitProblem** or **unavailableCriticalExtensions** should not be returned as it may compromise the security of this entry.

10.2.6.1 Search operation decision points for basic-access-control in the presence of additional searches

If the **joinArguments** argument is present, and if **basic-access-control** is in effect for the portion of the DIT to be searched, the following sequence of access controls applies to each additional search:

1) No specific permission is required to the entry identified by the **joinBaseObject** argument.

NOTE 1 – If the **joinBaseObject** is within the scope of the **joinArgument** (i.e., when the **joinSubset** argument specifies **baseObject** or **wholeSubtree**) the access controls specified in items 2) through 6) apply.

2) For each entry within the scope of the **joinArgument** which is to be a candidate for consideration, *Browse* permission is required. Entries for which this permission is not granted are ignored.

3) If present, the **joinFilter** argument is applied to each entry left to be considered after taking item 2) into account, in accordance with the following:

   a) For each **FilterItem** which specifies an attribute, *FilterMatch* permission for the attribute type is required before the **FilterItem** can be evaluated as either TRUE or FALSE. A **FilterItem** for which this permission is not granted evaluates as UNDEFINED.

   b) For each **FilterItem** which additionally specifies an attribute value, *FilterMatch* permission is required for each stored attribute value which is to be considered for the purposes of matching. If there is a value which both matches the **FilterItem** and for which permission is granted, the **FilterItem** evaluates to TRUE, otherwise it evaluates to FALSE.

4) If the **joinFilter** argument is not present, the **filter** argument is applied to each entry left to be considered after taking item 2) into account, in accordance with the following:

   a) For each **FilterItem** which specifies an attribute, *FilterMatch* permission for the attribute type is required before the **FilterItem** can be evaluated as either TRUE or FALSE. A **FilterItem** for which this permission is not granted evaluates as UNDEFINED.

   b) For each **FilterItem** which additionally specifies an attribute value, *FilterMatch* permission is required for each stored attribute value which is to be considered for the purposes of matching. If there is a value which both matches the **FilterItem** and for which permission is granted, the **FilterItem** evaluates to TRUE, otherwise it evaluates to FALSE.

5) Once the procedures defined in 2) through 4) have been applied, the entry is either selected or discarded. If as a consequence of applying these controls to the entire scoped subtree no entries have been selected (excluding any **ContinuationReferences** in **partialOutcomeQualifier**) and if *DiscloseOnError* permission is not granted to the entry identified by the **baseObject** argument, the operation fails and a **nameError** with problem **noSuchObject** shall be returned. The **matched** element shall either contain the name of the next superior entry to which *DiscloseOnError* permission is granted, or the name of the DIT

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

root (i.e., an empty **RDNSequence**). Otherwise, the operation succeeds but no subordinate information is conveyed with it.

NOTE 2 – In the case of a **nameError** being returned, the empty **RDNSequence** may be used by a DSA which does not have access to all superior entries.

NOTE 3 – Security policy may prevent the disclosure of knowledge information which would otherwise be conveyed as **ContinuationReferences** in **partialOutcomeQualifier**. If such a policy is in effect and if a DUA constrains the service by specifying **chainingProhibited** the Directory may return a **serviceError** with problem **chainingRequired**. Otherwise, the **ContinuationReference** is omitted from **partialOutcomeQualifier**.

6) Otherwise, for each selected entry the information returned is as follows:

   a) If the **infoTypes** element of **selection** specifies that attribute types only are to be returned, then for each attribute type that is to be returned, *Read* permission is required. If permission is not granted, the attribute type is omitted from **EntryInformation**. If as a consequence of applying these controls no attribute type information is selected, the **EntryInformation** element is returned but no attribute type information is conveyed with it (i.e., the **SET OF CHOICE** element is omitted or empty).

   b) If the **infoTypes** element of **selection** specifies that attribute types and values are to be returned, then for each attribute type and for each value that is to be returned, *Read* permission is required. If permission to an attribute type is not granted, the attribute is omitted from **EntryInformation**. If permission to an attribute value is not granted, the value is omitted from its corresponding attribute. In the event that permission is not granted to any of the values within the attribute, an **Attribute** element containing an empty **SET OF AttributeValue** is returned. If as a consequence of applying these controls no attribute information is selected, the **EntryInformation** element is returned but no attribute information is conveyed with it (i.e., the **SET OF CHOICE** element is omitted or empty).

NOTE 4 – If *DiscloseOnError* permission is not granted to the entry identified by the **baseObject** argument, a **partialOutcomeQualifier** indicating a **limitProblem** or **unavailableCriticalExtensions** should not be returned as it may compromise the security of this entry.

10.2.6.2 Alias dereferencing during Search

No specific permissions are necessary for alias dereferencing to take place in the course of a **search** operation (subject to the **searchAliases** parameter being set to **TRUE**). However, for each alias entry encountered, if alias dereferencing would result in a **ContinuationReference** being returned in **partialOutcomeQualifier**, the following access controls apply: *Read* permission is required to the alias entry, the **aliasedEntryName** attribute and to the single value that it contains. If any of these permissions is not granted, the **ContinuationReference** shall be omitted from **partialOutcomeQualifier**. These access controls shall also be applied to a **continuationReference** that is received in a response from another DSA. That is, the DSA shall police all **continuationReferences** whether they were generated locally or not.

NOTE – In addition to the access controls described above, security policy may prevent the disclosure of knowledge information that would otherwise be conveyed as **ContinuationReferences** in **partialOutcomeQualifier**. If such a policy is in effect and if a DUA constrains the service by specifying **chainingProhibited**, the Directory may return a **serviceError** with problem **chainingRequired**. Otherwise, the **ContinuationReference** is omitted from **partialOutcomeQualifier**.

10.2.6.3 Non-disclosure of incomplete results

If an incomplete result is being returned in **EntryInformation**, i.e., some of the attributes or attribute values have been omitted because of the applicable access controls, the **incompleteEntry** element shall be set to **TRUE** if *DiscloseOnError* permission is granted to at least one attribute type withheld from the result, or at least one attribute value withheld from the result (for which attribute type *Read* permission was granted).

10.2.7 Search operation decision points for rule-based access control

If **basic-access-control** is also applied, the order in which it is applied with respect to **rule-based-access-control** is a local matter, except that if access is denied to the entry, an attribute type or an attribute value by either mechanism, it shall not be overridden by the other mechanism. In this respect, *DiscloseOnError* permission of **basic-access-control** is a permission that shall not override a deny of **rule-based-access-control**.

If **rule-based-access-control**, **rule-and-basic-access-control**, or **rule-and-simple-access-control** is in effect for the portion of the DIB where the **search** operation is being performed, the following access controls apply:

1) If rule-based entry level permission is denied to the entry identified by the **baseObject** argument, then **nameError** with problem **noSuchObject** is returned as defined in 7.11.2.4.

2) Under **rule-based-access-control**, each entry within the scope of the **SearchArgument** for which entry level access is denied is ignored.

3) **basic-access-control** on entries is applied as defined in 10.2.6 item 2).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4) The **filter** is applied ignoring attribute values to which access is denied under **rule-based-access-control**.

5) **basic-access-control** on the **filter** is applied as defined in 10.2.6 items 3) and 4).

6) For any selected entry:

   a) for each attribute type that may be returned under **rule-based-access-control**, access must be granted to at least one attribute value of that type;

   b) attribute values to which access is denied under **rule-based-access-control** shall not be returned.

7) **basic-access-control** is applied to the information returned as defined in 10.2.6 item 5).

11 Directory Modify operations

There are four operations to modify the Directory: **addEntry**, **removeEntry**, **modifyEntry**, and **modifyDN** defined in 11.1 through 11.4, respectively.

NOTE 1 – Each of these operations identifies the target entry by means of its distinguished name.

NOTE 2 – The success of **addEntry**, **removeEntry**, and **modifyDN** operations may depend on the physical distribution of the DIB across the Directory. Failure shall be reported with an **updateError** with problem **affectsMultipleDSAs**. See ITU-T Rec. X.518 | ISO/IEC 9594-4.

NOTE 3 – In the event of failure of the underlying communications mechanism, the outcome of the operations is undetermined. The user should use Directory interrogation operations to check whether the attempted modification operation succeeded or not.

11.1 Add Entry

11.1.1 Add Entry syntax

An **addEntry** operation is used to add a leaf entry (either an object entry or an alias entry) to the DIT. The arguments of the operation may be signed (see 17.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2) by the requestor. If so requested, the Directory may sign the result.

```
addEntry  OPERATION  ::=  {
        ARGUMENT  AddEntryArgument
        RESULT            AddEntryResult
        ERRORS            { attributeError | nameError | serviceError | referral | securityError |
                          updateError }
        CODE        id-opcode-addEntry }

AddEntryArgument  ::=  OPTIONALLY-PROTECTED {
        SET {
            object              [0]    Name,
            entry               [1]    SET OF Attribute{{SupportedAttributes}},
            targetSystem        [2]    AccessPoint  OPTIONAL,
            COMPONENTS OF              CommonArguments } }

AddEntryResult  ::=  CHOICE {
        null              NULL,
        information       OPTIONALLY-PROTECTED-SEQ {
                          SEQUENCE { COMPONENTS OF CommonResultsSeq } } }
```

11.1.2 Add Entry arguments

The **object** argument identifies the entry to be added. Its immediate superior, which must already exist for the operation to succeed, is determined by removing the last RDN component (which belongs to the entry to be created). **object** may be an alternative name and may include context information, as described in 9.3 of ITU-T Rec. X.501 |

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

ISO/IEC 9594-2. The last RDN component shall be the primary RDN and shall include all distinguished values with their context lists for all attributes contributing to the RDN. Where any **AttributeTypeAndDistinguishedValue** in the last RDN component is provided without alternative distinguished values, the single value provided shall be used as the single distinguished value for that attribute.

The **entry** argument contains the attribute information which, together with that from the RDN, constitutes the entry to be created. The Directory shall ensure that the entry conforms to the Directory schema. Where the entry being created is an alias, no check is made to ensure that the **aliasedEntryName** attribute points to a valid entry.

The **targetSystem** argument indicates the DSA to hold the new entry. If this argument is absent, it shall be taken to mean the same DSA as holds the superior of the new object. If the argument is present, it shall be the DSA with the specified **AccessPoint**. The parameter shall be absent when subentries are to be added.

If the argument is present, the **targetSystem** bit in the **criticalExtensions** parameter in **CommonArguments** shall be set, indicating that this extension is critical.

NOTE 1 – If the choice of indicated or implied DSA conflicts with local administrative policy, the operation is not performed and an error is returned.

The **CommonArguments** (see 7.3) includes a specification of the service controls and security parameters applying to the request. The **dontDereferenceAlias** option is ignored (and treated as set) unless the **useAliasOnUpdate** critical extension bit is set in **criticalExtensions**. Thus aliases are dereferenced by this operation only if **dontDereferenceAlias** is not set and **useAliasOnUpdate** is set. The **sizeLimit** component is ignored if provided. If the argument of this operation is to be signed by the requestor, the **SecurityParameters** (see 7.10) component shall be included in the arguments.

NOTE 2 – Update operations that involve dereferencing of an alias name will always fail if they encounter first edition DSAs.

11.1.3 Add Entry results

Should the request succeed, a result shall be returned. If this result is to be signed by the Directory, the **SecurityParameters** (see 7.10) component of **CommonResultsSeq** (see 7.4) shall be included in the results. If the result of this operation is not to be signed by the Directory, no information shall be conveyed with the result.

11.1.4 Add Entry errors

Should the request fail, one of the listed errors shall be reported. The circumstances under which the particular errors shall be reported are defined in clause 12.

11.1.5 Add operation decision points for basic access control

If **rule-based-access-control** is also applied the order in which it is applied with respect to **basic-access-control** is a local matter, except that if access is denied to the entry, an attribute type or an attribute value by either mechanism, it shall not be overridden by the other mechanism. In this respect, *DiscloseOnError* permission of **basic-access-control** is a permission that shall not override a deny of **rule-based-access-control**.

If **basic-access-control** is in effect for the entry being added, the following sequence of access controls applies:

1) No specific permission is required to the immediate superior of the entry identified by the **object** argument.

NOTE 1 – Security policy may prevent Directory users from adding entries across DSA boundaries (e.g., using the **targetSystem** argument). In this event, an appropriate **nameError**, **serviceError**, **securityError** or **updateError** may be returned provided that it does not compromise the existence of the immediate superior entry. If it does (i.e., *DiscloseOnError* is not granted to the superior entry), the procedure defined in 7.11.3 shall be followed with respect to the superior entry.

2) If an entry already exists with a distinguished name equal to the **object** argument, the operation fails in accordance with 11.1.5.1, item a).

3) *Add* permission is required for the new entry being added. If this permission is not granted, the operation fails in accordance with 11.1.5.1, item b).

NOTE 2 – The *Add* permission shall be provided as **prescriptiveACI** when attempting to add an entry and as **prescriptiveACI** or **subentryACI** when attempting to add a subentry.

4) For each attribute type and for each value that is to be added, *Add* permission is required. If any permission is absent, the operation fails in accordance with 11.1.5.1, item c).

11.1.5.1 Error returns

If the operation fails as defined in 11.1.5, the following procedure applies:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

a) If the operation fails as defined in 11.1.5 item 2), the valid error returns are one of: if *DiscloseOnError* or *Add* permission is granted to the existing entry, an **updateError** with problem **entryAlreadyExists** shall be returned. Otherwise, the procedure described in 7.11.3 is followed with respect to the entry being added.

b) If the operation fails as defined in 11.1.5 item 3), the procedure described in 7.11.3 is followed with respect to the entry being added.

c) If the operation fails as defined in 11.1.5 item 4), the valid error return is **securityError** with problem **insufficientAccessRights** or **noInformation**.

11.1.6 Add Entry operation decision points for rule-based-access-control

If **basic-access-control** is also applied, the order in which it is applied with respect to **rule-based-access-control** is a local matter, except that if access is denied to the entry, an attribute type or an attribute value by either mechanism, it shall not be overridden by the other mechanism. In this respect, *DiscloseOnError* permission of **basic-access-control** is a permission that shall not override a deny of **rule-based-access-control**.

If **rule-based-access-control**, **rule-and-basic-access-control**, or **rule-and-simple-access-control** is in effect for the portion of the DIB where the **addEntry** operation is being performed, the following sequence of access control applies:

1) If rule-based entry level permission to the immediate superior is denied then **nameError** with problem **noSuchObject** is returned as defined in 7.11.2.4.

2) **basic-access-control** is applied as defined in 11.1.5.

11.2 Remove Entry

11.2.1 Remove Entry syntax

A Remove Entry operation is used to remove a leaf entry (either an object entry, family member or an alias entry) or a non-leaf ancestor and its children, from the DIT. The arguments of the operation may be signed (see 17.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2) by the requestor. If so requested, the Directory may sign the result.

```
removeEntry OPERATION ::= {
    ARGUMENT RemoveEntryArgument
    RESULT RemoveEntryResult
    ERRORS { nameError | serviceError | referral | securityError | updateError }
    CODE id-opcode-removeEntry }

RemoveEntryArgument ::= OPTIONALLY-PROTECTED {
    SET {
        object [0] Name,
        COMPONENTS OF CommonArguments } }

RemoveEntryResult ::= CHOICE {
    null NULL,
    information OPTIONALLY-PROTECTED-SEQ {
        SEQUENCE { COMPONENTS OF CommonResultsSeq } } }
```

11.2.2 Remove Entry arguments

The **object** argument identifies the entry to be deleted. The **object** may be an alternative name and may include context information, as described in 9.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2.

The **CommonArguments** (see 7.3) includes a specification of the service controls and security parameters applying to the request. The **dontDereferenceAlias** option is ignored (and treated as set) unless the **useAliasOnUpdate** critical extension bit is set in **criticalExtensions**. Thus aliases are dereferenced by this operation only if **dontDereferenceAlias** is not set and **useAliasOnUpdate** is set. The **sizeLimit** component is ignored if provided. If the argument of this

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

operation is to be signed by the requestor, the **SecurityParameters** (see 7.10) component shall be included in the arguments.

NOTE – Update operations that involve dereferencing of an alias name will always fail if they encounter first edition DSAs.

FamilyGrouping may be set as follows:

- **entryOnly** is the default for this operation. The entry to be removed shall be a leaf entry.
- **compoundEntry** may be specified for an ancestor. All the members of the compound entry will be removed. The operation will fail with an **updateError** with problem **notAncestor** if the target object is not an ancestor. The operation will also fail with an appropriate error if it is not possible to remove all members, e.g., for security reason.

If **FamilyGrouping** is absent or set to any other value than above, **entryOnly** is assumed.

11.2.3 Remove Entry results

Should the request succeed, a result shall be returned. If this result is to be signed by the Directory, the **SecurityParameters** (see 7.10) component of **CommonResultsSeq** (see 7.4) shall be included in the results. If the result of the operation is not to be signed by the Directory, no information shall be conveyed with the result.

When family information is selected by **familyReturn** in **EntryInformationSelection** the information returned is defined in 7.6.4.

The information returned in **information** component corresponds to the state of the DIB after the (successful) Modify Entry operation.

11.2.4 Remove Entry errors

Should the request fail, one of the listed errors shall be reported. The circumstances under which the particular errors shall be reported are defined in clause 12.

11.2.5 Remove Entry operation decision points for basic access control

If **rule-based-access-control** is also applied, the order in which it is applied with respect to **basic-access-control** is a local matter, except that if access is denied to the entry, an attribute type or an attribute value by either mechanism, it shall not be overridden by the other mechanism. In this respect, *DiscloseOnError* permission of **basic-access-control** is a permission that shall not override a deny of **rule-based-access-control**.

If **basic-access-control** is in effect for the entry being removed, the following access controls apply:

- *Remove* permission is required for the entry being removed. If this permission is not granted, the operation fails in accordance with 7.11.1.

NOTE – No specific permissions are required for any of the attributes and attribute values present within the entry being removed.

11.2.6 Remove Entry operation decision points for rule-based access control

If **basic-access-control** is also applied, the order in which it is applied with respect to **rule-based-access-control** is a local matter, except that if access is denied to the entry, an attribute type or an attribute value by either mechanism, it shall not be overridden by the other mechanism. In this respect, *DiscloseOnError* permission of **basic-access-control** is a permission that shall not override a deny of **rule-based-access-control**.

If **rule-based-access-control**, **rule-and-basic-access-control**, or **rule-and-simple-access-control** is in effect for the entry being removed, the following sequence of access control applies:

1) If rule-based entry level permission is not granted to the target entry, the operation fails with **nameError** with problem **noSuchObject** as defined in 7.11.2.4.
2) Entry level **basic-access-control** is applied as specified in 11.2.5.
3) If rule-based access is not granted to an attribute value then it shall not be removed.
4) If rule-based RDN permission is not granted, then none of the attribute values of the RDN shall be removed. If all the values of an attribute are removed, then the attribute is removed from the entry. If all the attributes are removed, then the entry is removed from the DIT. If at least one attribute value is removed, and the requestor does not have RDN permission, the operation succeeds but the entry remains in the DIT with one or more attributes.

NOTE 1 – Unless all the values of the label context for distinguished values of the entry have all the same values, this may not support a rule-based access-control policy.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

5) Under **rule-based-access-control**, if RDN permission is granted, but permission to access at least one other attribute value is not granted, then the RDN is not removed, and the operation fails with **securityError** with problem **insufficientAccessRights**. It is a local matter whether other attribute values to which the requestor has access permission are removed or not.

NOTE 2 – This reveals to the requestor that at least one attribute value exists that is inaccessible.

6) If all the attributes of the entry are removed, then the entry is removed from the DIT, and the operation is successful.

11.3 Modify Entry

11.3.1 Modify Entry syntax

The Modify Entry operation is used to perform a series of one or more of the following modifications to a single entry:

a) add a new attribute;

b) remove an attribute;

c) add attribute values;

d) remove attribute values;

e) replace attribute values;

f) modify an alias;

g) add a constant to all values of an attribute;

h) delete all attribute values for which fallback is **FALSE** in every context.

The arguments of the operation may be signed (see 17.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2) by the requestor. If so requested, the Directory may sign the result.

```
modifyEntry  OPERATION  ::=  {
        ARGUMENT  ModifyEntryArgument
        RESULT              ModifyEntryResult
        ERRORS              { attributeError | nameError | serviceError | referral | securityError |
                            updateError }
        CODE          id-opcode-modifyEntry }

ModifyEntryArgument  ::=  OPTIONALLY-PROTECTED {
        SET {
            object                  [0]     Name,
            changes                 [1]     SEQUENCE OF EntryModification,
            selection               [2]     EntryInformationSelection OPTIONAL,
            COMPONENTS OF           CommonArguments } }

ModifyEntryResult  ::=  CHOICE {
        null                NULL,
        information     OPTIONALLY-PROTECTED-SEQ  {
            SEQUENCE {
                entry           [0]     EntryInformation OPTIONAL,
                COMPONENTS OF   CommonResultsSeq } } }

EntryModification     ::=     CHOICE {
        addAttribute            [0]     Attribute{{SupportedAttributes}},
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

removeAttribute [1] AttributeType,

addValues [2] Attribute{{SupportedAttributes}},

removeValues [3] Attribute{{SupportedAttributes}},

alterValues [4] AttributeTypeAndValue,

resetValue [5] AttributeType{{SupportedAttributes}},

replaceValues [6] Attribute }

11.3.2 Modify Entry arguments

The **object** argument identifies the entry to which the modifications should be applied. The **object** may be an alternative name and may include context information, as described in 9.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2.

The **changes** argument defines a sequence of modifications that are applied in the order specified. If any of the individual modifications fails, then an **attributeError** is generated and the entry left in the state it was prior to the operation. That is, the operation is atomic. The end result of the sequence of modifications shall not violate the Directory schema. However, it is possible, and sometimes necessary, for the individual **EntryModification** changes to appear to do so. The following types of modification may occur:

a) **addAttribute** – This identifies a new attribute to be added to the entry, which is fully specified by the argument. Any attempt to add an already existing attribute results in an **attributeError**.

b) **removeAttribute** – The argument identifies (by its type) an attribute to be removed from the entry. Any attempt to remove a non-existing attribute results in an **attributeError**.

NOTE 1 – This operation is not allowed if the attribute type is present in the RDN.

c) **addValues** – This identifies an attribute by the attribute type in the argument, and specifies one or more attribute values to be added to the attribute. An attempt to add an already existing value results in an error. An attempt to add a value to a non-existent type results in the type and value being added.

d) **removeValues** – This identifies an attribute by the attribute type in the argument, and specifies one or more attribute values to be removed from the attribute. If the values are not present in the attribute, this results in an **attributeError**. An attempt to remove the last value from an attribute results in the attribute type being removed.

NOTE 2 – This operation is not allowed if one of the values is present in the RDN.

Attributes or attribute values to be added may be specified with or without a context list. Contexts cannot be added to existing attribute values, removed from existing attribute values, nor modified. To alter a context list of an existing attribute value, first remove the attribute value, and then insert the same attribute value with the new context list. When an attribute value is removed, no context list shall be supplied, and any existing context list associated with the attribute value being removed is removed with the attribute value.

e) **alterValues** – This identifies an attribute type, and specifies a quantity to be added to all values of the attribute. An attempt to apply this modification to an attribute whose syntax is other than **INTEGER** or **REAL** results in an **attributeError**.

f) **resetValue** – This identifies an attribute by its type, and removes all values of the attribute (if any) which have an associated attribute value context for which fallback is **FALSE**. **resetValue** does not remove any attribute values that have no context.

g) **replaceValues** – This replaces all existing values of the given attribute type with the values supplied, creating the attribute type if it did not exist. A replace with no value removes the attribute type if it exists, and is ignored if the type does not exist.

NOTE 3 – This Directory Specification does not establish rules regarding the order in which a performing DSA is to decode and process PDUs that it receives. If a DSA decodes the entire PDU before processing each element, and if a new and unexpected value, such as **replaceValues**, is in place for a non-optional CHOICE, it is possible that the DSA will signal an encoding error. If, however, the DSA decodes the elements as they are needed, it will most likely detect an unknown critical extension and return an unsupported critical extension reason code to signal that the operation failed. In either case, it is correct for the DSA to not process the operation; however, implementers should be aware that either signal may be used to indicate the failure of the operation.

Values may be replaced by a combination of **addValues** and **removeValues** in a single **ModifyEntry** operation.

The **CommonArguments** (see 7.3) includes a specification of the service controls and security parameters applying to the request. The **dontDereferenceAlias** option is ignored (and treated as set) unless the **useAliasOnUpdate** critical extension bit is set in **criticalExtensions**. Thus, aliases are dereferenced by this operation only if

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

dontDereferenceAlias is not set and **useAliasOnUpdate** is set. The **sizeLimit** component is ignored if provided. If the argument of this operation is to be signed by the requestor, the **SecurityParameters** (see 7.10) component shall be included in the arguments.

NOTE 4 – Update operations that involve dereferencing of an alias name will always fail if they encounter first edition DSAs.

The **selection** argument specifies an optional entry information selection that controls whether information is returned in the operation result and specifies the specific attributes and values to be returned. It shall only be specified if the version negotiated through the bind operation is **v2** or higher.

The operation may be used to modify directory operational attributes. Only those directory operational attributes which are not classified **noUserModification** (and to which the user has effective modification access rights) may be modified.

NOTE 5 – Whether or not user modification is permitted, the Directory may change the values of directory operational attributes as a side effect of other Directory operations.

The operation may be used to modify collective attributes only if the service control **subentries** is **TRUE** and if the **object** is the subentry actually holding the collective attribute(s) to be modified.

NOTE 6 – Caution should therefore be exercised when modifying the information returned on reading an entry: some of the information may be from collective attributes, and cannot be modified in an operation directed at the entry itself. For example, it is not possible to delete a collective attribute from an (ordinary) entry via a **removeAttribute** entry modification to the entry (an **attributeError** with **problem noSuchAttributeOrValue** would be returned).

The operation may be used to modify an entry's Object Class attribute value if the values specify auxiliary object classes. However, an attempt to change an Object Class value which specifies an entry's structural object class shall result in an **updateError** with problem **objectClassModificationProhibited**. Any modification to auxiliary object classes shall leave the superclass chains consistent and correct with the resultant object class definition.

11.3.3 Modify Entry results

Should the request succeed, a **result** shall be returned. If no **selection** was specified in the operation argument and the result is not to be signed, the null result is returned. If no **selection** was specified (but the result is to be signed by the Directory), the entry component is omitted. If the result is to be signed by the Directory, the **SecurityParameters** (see 7.10) component of **CommonResultsSeq** (see 7.4) shall be included in the results. If the result is not to be signed by the Directory, no entry information shall be conveyed with the result.

11.3.4 Modify Entry errors

Should the request fail, one of the listed errors shall be reported. The circumstances under which the particular errors shall be reported are defined in clause 12.

11.3.5 Modify Entry operation decision points for basic access control

If **rule-based-access-control** is also applied, the order in which it is applied with respect to **basic-access-control** is a local matter, except that if access is denied to the entry, an attribute type or an attribute value by either mechanism, it shall not be overridden by the other mechanism. In this respect, *DiscloseOnError* permission of **basic-access-control** is a permission that shall not override a deny of **rule-based-access-control**.

If **basic-access-control** is in effect for the entry being modified, the following sequence of access controls applies:

1) *Modify* permission is required for the entry being modified. If this permission is not granted, the operation fails in accordance with 7.11.1.
2) For each of the specified **EntryModification** arguments applied in sequence, the following permissions are required:
   i) *Add* permission for the attribute type and for each of the values specified in an **addAttribute** parameter. If these permissions are not granted or the attribute already exists, the operation fails in accordance with 11.3.5.1, item a).
   ii) *Remove* permission for the attribute type specified in a **removeAttribute** parameter. If this permission is not granted, the operation fails in accordance with 11.3.5.1, item b).

   NOTE 1 – No specific permissions are required for any of the attribute values present within the attribute being removed.

   iii) *Add* permission on each of the attribute values specified in an **addValues** parameter. If these permissions are not granted or any of the attribute values already exist, the operation fails in accordance with 11.3.5.1, item c).
   iv) *Remove* permission on each of the values specified in a **removeValues** parameter. If these permissions are not granted, the operation fails in accordance with 11.3.5.1, item d).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

NOTE 2 – If the end result of a **removeValues** modification is to remove the last value of an attribute (which causes the attribute itself to be removed), Remove permission is also required on the specified attribute type.

v) *Add* and *Remove* permission on each of the values specified in an **alterValues** parameter. If these permissions are not granted, the operation fails in accordance with 11.3.5.1, item e).

vi) *Remove* permission on each of the values to be removed via a **resetValue** parameter. If at least one value is to be removed and these permissions are not granted, the operation fails in accordance with 11.3.5.1, item f).

11.3.5.1 Error returns

If the operation fails as defined in 11.3.5, the following procedure applies:

a) If the operation fails as defined in 11.3.5 item 2), subitem i), the valid error returns are one of: if the attribute already exists and *discloseOnError* or *add* is granted to that attribute, an **attributeError** with problem **attributeOrValueAlreadyExists** shall be returned; otherwise, a **securityError** with problem **insufficientAccessRights** or **noInformation** shall be returned.

b) If the operation fails as defined in 11.3.5 item 2), subitem ii), the valid error returns are one of: if *DiscloseOnError* permission is granted to the attribute being removed and the attribute exists, a **securityError** with problem **insufficientAccessRights** or **noInformation** shall be returned; otherwise, an **attributeError** with problem **noSuchAttributeOrValue** shall be returned.

c) If the operation fails as defined in 11.3.5 item 2), subitem iii), the valid error returns are one of: if an attribute value already exists and *discloseOnError* or *add* is granted to that attribute value, an **attributeError** with problem **attributeOrValueAlreadyExists** shall be returned; otherwise, *discloseOnError* permission at the attribute level shall be verified. If *discloseOnError* is granted to the attribute, a **securityError** with problem **insufficientAccessRights** or **noInformation** shall be returned; otherwise, an **attributeError** with problem **noSuchAttributeOrValue** shall be returned.

d) If the operation fails as defined in 11.3.5 item 2), subitem iv), the valid error returns are one of: if *DiscloseOnError* permission is granted to any of the attribute values being removed, a **securityError** with problem **insufficientAccessRights** or **noInformation** shall be returned; otherwise, an **attributeError** with problem **noSuchAttributeOrValue** shall be returned.

e) If the operation fails as defined in 11.3.5 item 2), subitem v), the valid error returns are one of: if *DiscloseOnError* permission is granted to any of the attribute values being altered, a **securityError** with problem **insufficientAccessRights** or **noInformation** shall be returned; otherwise, an **attributeError** with problem **noSuchAttributeOrValue** shall be returned.

f) If the operation fails as defined in 11.3.5 item 2), subitem vi), the valid error returns are one of: if *DiscloseOnError* permission is granted to any of the attribute values being removed, a **securityError** with problem **insufficientAccessRights** or **noInformation** shall be returned; otherwise, an **attributeError** with problem **noSuchAttributeOrValue** shall be returned.

11.3.6 Modify Entry operation decision points for rule-based access control

If **basic-access-control** is also applied, the order in which it is applied with respect to **rule-based-access-control** is a local matter, except that if access is denied to the entry, an attribute type or an attribute value by either mechanism, it shall not be overridden by the other mechanism. In this respect, *DiscloseOnError* permission of **basic-access-control** is a permission that shall not override a deny of **rule-based-access-control**.

If **rule-based-access-control**, **rule-and-basic-access-control**, or **rule-and-simple-access-control** is in effect for the entry being modified, the following sequence of access control applies:

1) If rule-based entry level permission is not granted to the target entry, then the operation fails with **nameError** with problem **noSuchObject** according to 7.11.2.4.

2) Entry level **basic-access-control** is applied according to 11.3.5.1.

3) Access must be granted to each of the attribute values (if any) that are removed. If **rule-based-access-control** permission is not granted to any attribute value that is to be removed, the operation fails with **attributeError** with problem **noSuchAttributeOrValue**.

4) Attribute level **basic-access-control** is applied as in 11.3.5 item 2).

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

11.4 Modify DN

11.4.1 Modify DN syntax

The Modify DN operation is used to change the Relative Distinguished Name of an entry, to change the primary Relative Distinguished Name of an entry, to add and subtract distinguished values of attributes, and/or to move an entry to a new superior in the DIT. It may be used with object entries, including compound entries or alias entries.

For family members, its use is restricted to the case where the affected family members stay within the same compound entry.

If the entry has subordinates, then all subordinates are renamed or moved accordingly (i.e., the subtree remains intact). The arguments of the operation may be signed (see 17.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2) by the requestor. If so requested, the Directory may sign the result.

NOTE 1 – First edition systems may use the operation only to change the Relative Distinguished Name of a leaf entry.

NOTE 2 – Second and subsequent edition systems may use the operation to move entries to a new superior only if the old superior, the new superior, the entry, and all its subordinates are in the one DSA.

NOTE 3 – The operation does not move entries to a new DSA; all entries remain in the original DSA.

NOTE 4 – The operation either succeeds or fails in its entirety; it shall not fail with some entries moved and some not moved. No intermediate states of the operation shall be externally visible to users of the Directory.

NOTE 5 – Some offline activity may be required following this operation to preserve consistency, for example to update attributes in any entries that hold Distinguished Name values that refer to the renamed or moved entry(ies).

NOTE 6 – The **modifyTimeStamp** attribute is not updated for entries subordinate to the renamed or moved entry.

```
modifyDN OPERATION ::= {
    ARGUMENT ModifyDNArgument
    RESULT   ModifyDNResult
    ERRORS   { nameError | serviceError | referral | securityError | updateError }
    CODE     id-opcode-modifyDN }

ModifyDNArgument ::= OPTIONALLY-PROTECTED {
    SET {
        object       [0]  DistinguishedName,
        newRDN       [1]  RelativeDistinguishedName,
        deleteOldRDN [2]  BOOLEAN DEFAULT FALSE,
        newSuperior  [3]  DistinguishedName OPTIONAL,
                          COMPONENTS OF CommonArguments } }

ModifyDNResult ::= CHOICE {
    null         NULL,
    information  OPTIONALLY-PROTECTED-SEQ {
        SEQUENCE {
            newRDN                RelativeDistinguishedName,
            COMPONENTS OF CommonResultsSeq } } }
```

11.4.2 Modify DN arguments

The **object** argument identifies the entry whose Distinguished Name is to be modified. Aliases in the name shall not be dereferenced. The **object** may be an alternative name and may include context information, as described in 9.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2.

The **newRDN** argument specifies the new RDN of the entry. If the operation moves the entry to a new superior without changing its RDN, the old RDN is supplied for this parameter.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

If an attribute value in the new RDN does not already exist in the entry (either as part of the old RDN or as a non-distinguished value) it is added. If it cannot be added, an error is returned.

For each attribute contributing to the RDN, **newRDN** may provide alternative distinguished values if those distinguished values are differentiated by context, as described in 9.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2. If so, **newRDN** shall be a primary RDN and shall include all distinguished values with their context lists for all attributes contributing to the RDN (including existing distinguished values that are to be retained as distinguished values). An **AttributeTypeAndDistinguishedValue** in **newRDN** which is provided without alternative distinguished values indicates a single distinguished value for that attribute.

If the **deleteOldRDN** flag is set, all attribute values in the old RDN that are not in the new RDN are deleted. This includes alternative distinguished values differentiated by contexts, if they exist in the old RDN but are not included in the new RDN. If this flag is not set, the old distinguished values shall remain in the entry (but are no longer distinguished values). The flag shall be set where a single value attribute in the RDN has its value changed by the operation. If an attribute value in the old RDN is the same as one in the new RDN except for their context lists, the one in the old RDN is replaced by the one in the new RDN. If this operation removes the last attribute value of an attribute, that attribute shall be deleted.

The **newSuperior** argument, if present, specifies that the entry is to be moved to a new superior in the DIT. The entry becomes an immediate subordinate of the entry with the indicated Distinguished Name, which must be an already existing object entry. The new superior shall not be the entry itself or any of its subordinates, or an alias, or such that the moved entry violates any DIT structure rules. It is possible that entries *subordinate* to the moved entry may violate the active subschema, in which case it is the responsibility of the Subschema Administrative Authority to make subsequent adjustments to these entries to make them consistent with the subschema, as described in clause 14 of ITU-T Rec. X.501 | ISO/IEC 9594-2.

If the argument is present, the **newSuperior** bit in the **criticalExtensions** parameter in **CommonArguments** shall be set, indicating that this extension is critical.

The **newSuperior** may be an alternative name and may include context information, as described in 9.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2.

The **CommonArguments** (see 7.3) includes a specification of the service controls and security parameters applying to the request. For the purposes of this operation the **dontDereferenceAlias** option and the **sizeLimit** component are not relevant and are ignored if provided. Aliases are never dereferenced by this operation. If the argument of this operation is to be signed by the requestor, the **SecurityParameters** (see 7.10) component shall be included in the arguments.

11.4.3 Modify DN results

Should the request succeed, a result shall be returned. If this result is to be signed by the Directory, the **SecurityParameters** (see 7.10) component of **CommonResultsSeq** (see 7.4), and the new RDN shall be included in the results. If the result is not to be signed by the Directory, no information shall be conveyed with the result.

11.4.4 Modify DN errors

Should the request fail, one of the listed errors shall be reported. The circumstances under which the particular errors shall be returned are defined in clause 12.

11.4.5 ModifyDN decision points for basic access control

If **rule-based-access-control** is also applied, the order in which it is applied with respect to **basic-access-control** is a local matter, except that if access is denied to the entry, an attribute type or an attribute value by either mechanism, it shall not be overridden by the other mechanism. In this respect, *DiscloseOnError* permission of **basic-access-control** is a permission that shall not override a deny of **rule-based-access-control**.

If **basic-access-control** is in effect for the entry being renamed, the following access controls apply:

- If the effect of the operation is to change the RDN of the entry, *Rename* permission is required for the entry being renamed (considered with its original name). If this permission is not granted, the operation fails in accordance with 11.4.5.1.
- If the effect of the operation is to move an entry to a new superior in the DIT, *Export* permission is required for the entry being considered with its original name, and *Import* permission is required for the entry being considered with its new name. If either of these permissions is not granted, the operation fails in accordance with 11.4.5.1.

NOTE 1 – The *Import* permission shall be provided as prescriptive ACI.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

NOTE 2 – No additional permissions are required even if, as a result of modifying the last RDN of the name, a new distinguished value needs to be added or an old one removed.

11.4.5.1 Error returns

If the operation fails as defined in 11.4.5, the procedure described in 7.11.1 is followed with respect to the entry being renamed (considered with its original name).

11.4.6 Modify DN operation decision points for rule-based access control

If **basic-access-control** is also applied, the order in which it is applied with respect to **rule-based-access-control** is a local matter, except that if access is denied to the entry, an attribute type or an attribute value by either mechanism, it shall not be overridden by the other mechanism. In this respect, *DiscloseOnError* permission of **basic-access-control** is a permission that shall not override a deny of **rule-based-access-control**.

If **rule-based-access-control**, **rule-and-basic-access-control**, or **rule-and-simple-access-control** is in effect for the entry being renamed, the following sequence of access control applies:

1) If rule-based RDN permission is not granted to the target entry, the operation fails with **nameError** with problem **noSuchObject** in accordance with 7.11.2.4.
2) Entry level **basic-access-control** is applied as in 11.4.5.
3) If the effect of the operation is to move the entry to a new superior in the DIT, rule-based RDN permission is required to the new superior, else the operation fails with **nameError** with problem **noSuchObject** in accordance with 7.11.2.4.

12 Errors

12.1 Error precedence

The Directory does not continue to perform an operation beyond the point at which it determines that an error is to be reported.

NOTE 1 – An implication of this rule is that the first error encountered can differ for repeated instances of the same query, as there is not a specific logical order in which to process a given query. For example, DSAs may be searched in different orders.

NOTE 2 – The rules of error precedence specified here apply only to the abstract service provided by the Directory as a whole. Different rules apply when the internal structure of the Directory is taken into account.

Should the Directory simultaneously detect more than one error, the following list determines which error is reported. An error higher in the list has a higher logical precedence than one below it, and is the error which is reported.

a) **nameError**;
b) **updateError**;
c) **attributeError**;
d) **securityError**;
e) **serviceError**.

The following errors do not present any precedence conflicts:

a) **abandonFailed**, because it is specific to one operation, Abandon, which can encounter no other error;
b) **abandoned**, which is not reported if an Abandon operation is received simultaneously with the detection of an error. In this case an **abandonFailed** error with problem **tooLate** is returned along with the report of the actual error encountered;
c) **referral**, which is not a "real" error, only an indication that the Directory has detected that the DUA should present its request to another access point.

12.2 Abandoned

This outcome may be reported for any outstanding directory enquiry operation (i.e., Read, Search, Compare, List) if the DUA invokes an Abandon operation with the appropriate **InvokeId**. If the parameters of the operation were signed (see 17.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2) by the requestor, then the Directory may sign the error parameters.

```
abandoned ERROR ::= { -- not literally an "error"
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
PARAMETER       OPTIONALLY-PROTECTED {
    SET {
        COMPONENTS OF   CommonResults} }
CODE            id-errcode-abandoned }
```

The **SecurityParameters** component (see 7.10) shall be included in the **CommonResults** (see 7.4) if the error is to be signed by the Directory.

12.3 Abandon Failed

The **abandonFailed** error reports a problem encountered during an attempt to abandon an operation. If the parameters of the operation were signed (see 17.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2) by the requestor, then the Directory may sign the error parameters.

```
abandonFailed  ERROR  ::= {
    PARAMETER       OPTIONALLY-PROTECTED {
        SET {
            problem                 [0]     AbandonProblem,
            operation               [1]     InvokeId,
            COMPONENTS OF           CommonResults } }
    CODE            id-errcode-abandonFailed }

AbandonProblem  ::=  INTEGER { noSuchOperation (1), tooLate (2), cannotAbandon (3) }
```

The various parameters have the following meanings.

The particular **problem** encountered is specified. Any of the following problems may be indicated:

a) **noSuchOperation** – When the Directory has no knowledge of the operation which is to be abandoned (this could be because no such invoke took place, or because the Directory has forgotten about it);

b) **tooLate** – When the Directory has already responded to the operation;

c) **cannotAbandon** – When an attempt has been made to abandon an operation for which this is prohibited (e.g., modify), or the abandon could not be performed.

The identification of the particular **operation** (invocation) to be abandoned.

The **SecurityParameters** component (see 7.10) shall be included in the **CommonResults** (see 7.4) if the error is to be signed by the Directory.

The information provided by the error problem can optionally be qualified by the use of the **notification** component of **CommonResults**.

12.4 Attribute Error

An **attributeError** reports an attribute-related problem. If the parameters of the operation were signed (see 17.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2) by the requestor, then the Directory may sign the error parameters.

```
attributeError  ERROR  ::= {
    PARAMETER       OPTIONALLY-PROTECTED {  SET {
        object  [0]     Name,
        problems        [1]     SET OF SEQUENCE {
            problem         [0]     AttributeProblem,
            type            [1]     AttributeType,
```


--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
                value   [2]     AttributeValue  OPTIONAL },
            COMPONENTS OF   CommonResults } }
        CODE        id-errcode-attributeError }

AttributeProblem  ::=  INTEGER {
        noSuchAttributeOrValue          (1),
        invalidAttributeSyntax                  (2),
        undefinedAttributeType          (3),
        inappropriateMatching                   (4),
        constraintViolation             (5),
        attributeOrValueAlreadyExists   (6),
        contextViolation                        (7) }
```

The various parameters have the following meanings.

The **object** parameter identifies the entry to which the operation was being applied when the error occurred. The name returned may include only the primary distinguished values for attributes containing multiple distinguished values differentiated by context (i.e., the DSA need not apply context selection as described in 7.7, as it does for successful operations).

One or more **problems** may be specified. Each **problem** (identified below) is accompanied by an indication of the attribute **type**, and, if necessary to avoid ambiguity, the **value**, which caused the problem:

a) **noSuchAttributeOrValue** – The named entry lacks one of the attributes or attribute values specified as an argument of the operation.

b) **invalidAttributeSyntax** – A purported attribute value, specified as an argument of the operation, does not conform to the attribute syntax of the attribute type.

c) **undefinedAttributeType** – An undefined attribute type was provided as an argument to the operation. This error may occur only in relation to **addEntry** or **modifyEntry** operations.

d) **inappropriateMatching** – An attempt was made, e.g., in a filter, to use a matching rule not defined for the attribute type concerned.

e) **constraintViolation** – An attribute value supplied in the argument of an operation does not conform to the constraints imposed by ITU-T Rec. X.501 | ISO/IEC 9594-2 or by the attribute definition (e.g., the value exceeds the maximum size allowed).

f) **attributeOrValueAlreadyExists** – An attempt was made to add an attribute which already existed in the entry, or a value which already existed in the attribute.

g) **contextViolation** – A context list or context supplied with an attribute value in the argument of an operation does not conform to the constraints imposed by ITU-T Rec. X.501 | ISO/IEC 9594-2, by the context definition (e.g., the context value is not of the correct syntax), or the DIT Context Use.

The **SecurityParameters** component (see 7.10) shall be included in the **CommonResults** (see 7.4) if the error is to be signed by the Directory.

The information provided by the error problem can optionally be qualified by the use of the **notification** component of **CommonResults**.

12.5 Name Error

A **nameError** reports a problem related to the name provided as an argument to an operation. If the parameters of the operation were signed (see 17.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2) by the requestor, then the Directory may sign the error parameters.

```
nameError  ERROR  ::= {
        PARAMETER       OPTIONALLY-PROTECTED {  SET {
            problem                     [0]     NameProblem,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        matched                 [1]     Name,
        COMPONENTS OF           CommonResults } }
    CODE        id-errcode-nameError }

NameProblem ::= INTEGER {
    noSuchObject            (1),
    aliasProblem            (2),
    invalidAttributeSyntax      (3),
    aliasDereferencingProblem   (4),
    contextProblem              (5) }
```

The various parameters have the following meanings.

The particular **problem** encountered. Any of the following problems may be indicated:

a) **noSuchObject** – The name supplied does not match the name of any object.

b) **aliasProblem** – An alias has been dereferenced which names no object.

c) **invalidAttributeSyntax** – An attribute type and its accompanying attribute value in an AVA in the name are incompatible.

d) **aliasDereferencingProblem** – An alias was encountered in a situation where it was not allowed or where access was denied.

e) **contextProblem** – A context type or value used in a name is not understood or is invalid, the use of a context variant name is not acceptable, or during name resolution a purported name matches the names of more than one DIT entry.

The **matched** parameter contains the name of the lowest entry (object or alias) in the DIT that was matched, and is a truncated form of the name provided or, if an alias has been dereferenced, of the resulting name. The name returned may include only the primary distinguished values for attributes containing multiple distinguished values differentiated by context (i.e., the DSA need not apply context selection as described in 7.7, as it does for successful operations).

NOTE – If there is a problem with the attribute types and/or values in the name offered in a Directory operation argument, this is reported via a **nameError** with problem **invalidAttributeSyntax** rather than as an **attributeError** or an **updateError**.

The **SecurityParameters** component (see 7.10) shall be included in the **CommonResults** (see 7.4) if the error is to be signed by the Directory.

The information provided by the error problem can optionally be qualified by the use of the **notification** component of **CommonResults**.

12.6 Referral

A **referral** redirects the service-user to one or more access points better equipped to carry out the requested operation. If the parameters of the operation were signed (see 17.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2) by the requestor, then the Directory may sign the error parameters.

```
referral ERROR ::= { -- not literally an "error"
    PARAMETER       OPTIONALLY-PROTECTED { SET {
            candidate               [0]     ContinuationReference,
            COMPONENTS OF           CommonResults } }
    CODE            id-errcode-referral }
```

The error has a single parameter which contains a **ContinuationReference** which can be used to progress the operation (see ITU-T Rec. X.518 | ISO/IEC 9594-4).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

If the DSA is responding to an LDAP request, the **accessPoints** component in **ContinuationReference** shall contain a valid **LabeledURI** value, in which case it will use that value to create an LDAP referral. If it does not contain a valid **LabeledURI** value, it shall not return a referral.

The **SecurityParameters** component (see 7.10) shall be included in the **CommonResults** (see 7.4) if the error is to be signed by the Directory.

Before acting on a continuation reference, the DUA shall check that an identical request to the one that would be generated from the continuation reference has not already been issued as a part of processing the same user request. If it has, the DUA shall not act on the continuation reference. This avoids loops.

12.7 Security Error

A **securityError** reports a problem in carrying out an operation for security reasons. If the parameters of the operation were signed (see 17.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2) by the requestor, then the Directory may sign the error parameters.

```
securityError ERROR ::= {
        PARAMETER       OPTIONALLY-PROTECTED { SET {
                        problem         [0]     SecurityProblem,
                        spkmInfo        [1]     SPKM-ERROR,
                                                COMPONENTS OF CommonResults } }
        CODE            id-errcode-securityError }

SecurityProblem ::= INTEGER {
        inappropriateAuthentication (1),
        invalidCredentials          (2),
        insufficientAccessRights    (3),
        invalidSignature                (4),
        protectionRequired          (5),
        noInformation               (6),
        blockedCredentials          (7),
--      invalidQOPMatch                 (8), obselete
        spkmError                   (9) }
```

The error has a single parameter, which reports the particular **problem** encountered. The following problems may be indicated:

a) **inappropriateAuthentication** – The level of security associated with the requestor's credentials is inconsistent with the level of protection requested, e.g., simple credentials were supplied while strong credentials were required.

b) **invalidCredentials** – The supplied credentials were invalid.

c) **insufficientAccessRights** – The requestor does not have the right to carry out the requested operation.

d) **invalidSignature** – The signature of the request was found to be invalid.

e) **protectionRequired** – The Directory was unwilling to carry out the requested operation because the argument was not signed.

f) **noInformation** – The requested operation produced a security error for which no information is available.

g) **blockedCredentials** – The credentials are blocked from consideration for security reasons (e.g., because an invalid password has been presented too many times in succession). The decision to return this error is governed by the security policy in effect for the DSA.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

h) **spkmError** – The supplied SPKM token was found to be invalid. The **spkmInfo** parameter contains an indication that this is an SPKM error token and the identifier of the SPKM context with which this error is associated.

The **SecurityParameters** component (see 7.10) shall be included in the **CommonResults** (see 7.4) if the error is to be signed by the Directory.

The information provided by the error problem can optionally be qualified by the use of the **notification** component of **CommonResults**.

12.8 Service Error

A **serviceError** reports a problem related to the provision of the service. If the parameters of the operation were signed (see 17.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2) by the requestor, then the Directory may sign the error parameters.

```
serviceError ERROR ::= {
    PARAMETER       OPTIONALLY-PROTECTED { SET {
                        problem             [0]     ServiceProblem,
                        COMPONENTS OF               CommonResults } }
    CODE            id-errcode-serviceError }

ServiceProblem ::= INTEGER {
    busy                            (1),
    unavailable                     (2),
    unwillingToPerform              (3),
    chainingRequired                        (4),
    unableToProceed                 (5),
    invalidReference                (6),
    timeLimitExceeded               (7),
    administrativeLimitExceeded     (8),
    loopDetected                    (9),
    unavailableCriticalExtension    (10),
    outOfScope                              (11),
    ditError                                (12),
    invalidQueryReference           (13),
    requestedServiceNotAvailable    (14),
    unsupportedMatchingUse          (15),
    ambiguousKeyAttributes          (16),
    saslBindInProgress              (17) }
```

The error has a single parameter which reports the particular **problem** encountered. The following problems may be indicated:

a) **busy** – The Directory, or some part of it, is presently too busy to perform the requested operation, but may be able to do so after a short while.

b) **unavailable** – The Directory, or some part of it, is currently unavailable.

c) **unwillingToPerform** – The Directory, or some part of it, is not prepared to execute this request, e.g., because it would lead to excessive consumption of resources or violates the policy of an Administrative Authority involved.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

d) **chainingRequired** – The Directory is unable to accomplish the request other than by chaining; however, chaining was prohibited by means of the **chainingProhibited** service control option.

e) **unableToProceed** – The DSA returning this error did not have administrative authority for the appropriate naming context and as a consequence was not able to participate in name resolution.

f) **invalidReference** – The DSA was unable to perform the request as directed by the DUA, (via **OperationProgress**) – This may have arisen due to using an invalid referral.

g) **timeLimitExceeded** – The Directory has reached the limit of time set by the user in a service control. No partial results are available to return to the user.

h) **administrativeLimitExceeded** – The Directory has reached some limit set by an administrative authority, and no partial results are available to return to the user.

i) **loopDetected** – The Directory is unable to accomplish this request due to an internal loop.

j) **unavailableCriticalExtension** – The Directory was unable to satisfy the request because one or more critical extensions were not available.

k) **outOfScope** – No referrals were available within the requested scope.

l) **ditError** – The Directory is unable to accomplish the request due to a DIT consistency problem.

m) **invalidQueryReference** – The parameters of the requested operation are invalid. This problem is reported if the **queryReference** in paged results is invalid.

NOTE – This problem is not supported by first edition systems.

n) **requestedServiceNotAvailable** – A search request failed within a service specific administrative area because no search-rule was available for the search or because the search violated an applicable search-rule. Additional diagnostic information may be returned together with this service problem. Such additional information for different situations is defined in clause 13.

o) **unsupportedMatchingUse** – An attempt was made, e.g., in a filter, to use a matching rule not supported by the DSA when the **performExactly** search option is set.

p) **ambiguousKeyAttributes** – A mapping-based matching rule was selected, but the mappable filter items provided multiple matches against the relevant mapping table. This error situation is accompanied by a notification attribute as indicated by the relevant matching-based matching rule.

q) **saslBindInProgress** – For some authentication mechanisms, it may be necessary for the requestor to invoke the **directoryBind** operation multiple times. This is indicated by the responder sending a **serviceError** with problem **saslBindInProgress**. This indicates that the responder requires the requestor to invoke a new **directoryBind** operation, with the same **SaslCredentials** mechanism, to continue the authentication process. If at any stage the requestor wishes to abort the process it may invoke a **directoryBind** operation with **SaslAbort** set to **TRUE**.

The **SecurityParameters** component (see 7.10) shall be included in the **CommonResults** (see 7.4) if the error is to be signed by the Directory.

The information provided by the error problem can optionally be qualified by the use of the **notification** component of **CommonResults**.

12.9 Update Error

An **updateError** reports problems related to attempts to add, delete, or modify information in the DIB. If the parameters of the operation were signed (see 17.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2) by the requestor, then the Directory may sign the error parameters.

```
updateError ERROR ::= {
    PARAMETER       OPTIONALLY-PROTECTED {
        SET {
            problem             [0]  UpdateProblem,
            attributeInfo       [1]  SET SIZE (1..MAX) OF CHOICE {
                attributeType        AttributeType,
                attribute            Attribute{{SupportedAttributes}} } OPTIONAL,
                                     COMPONENTS OF CommonResults } }
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        CODE        id-errcode-updateError }

UpdateProblem ::= INTEGER {
        namingViolation                         (1),
        objectClassViolation                    (2),
        notAllowedOnNonLeaf                     (3),
        notAllowedOnRDN                         (4),
        entryAlreadyExists                      (5),
        affectsMultipleDSAs                     (6),
        objectClassModificationProhibited   (7),
        noSuchSuperior                          (8),
        notAncestor                             (9),
        parentNotAncestor                       (10),
        hierarchyRuleViolation                  (11),
        familyRuleViolation                     (12) }
```

The problem parameter reports the particular problem encountered. The following problems may be indicated.

a) **namingViolation** – The attempted addition or modification would violate the structure rules of the DIT as defined in the Directory schema and ITU-T Rec. X.501 | ISO/IEC 9594-2. That is, it would place an entry as the subordinate of an alias entry, or in a region of the DIT not permitted to a member of its object class, or would define an RDN for an entry to include a forbidden attribute type.

b) **objectClassViolation** – The attempted update would produce an entry inconsistent with the rules for entry content; for example, its object class definition, the DIT content rules, or with the definitions of ITU-T Rec. X.501 | ISO/IEC 9594-2 as they pertain to object classes.

c) **notAllowedOnNonLeaf** – The attempted operation is only allowed on leaf entries of the DIT.

d) **notAllowedOnRDN** – The attempted operation would affect the RDN (e.g., removal of an attribute which is a part of the RDN).

e) **entryAlreadyExists** – An attempted **addEntry** or **modifyDN** operation names an entry which already exists.

NOTE 1 – This includes a conflict caused by RDNs which include multiple distinguished values differentiated by contexts, regardless of context, as described in 9.3 of ITU-T Rec. X.501 | ISO/IEC 9594-2.

f) **affectsMultipleDSAs** – An attempted update would need to operate on multiple DSAs where this operation is not permitted.

g) **objectClassModificationProhibited** – An operation attempted to modify the structural object class of an entry.

h) **noSuchSuperior** – An attempted ModifyDN operation names a new superior entry that does not exist.

i) **notAncestor** – An operation attempted to delete a compound entry without specifying the ancestor as the object.

j) **parentNotAncestor** – An operation attempted to establish an entry as an immediately hierarchical child under a family member that is not the ancestor.

k) **hierarchyRuleViolation** – An operation attempted to break a rule applicable to a hierarchical group: a hierarchical group has to be completely outside any service specific administrative area or has to be completely contained within a service specific administrative area; hierarchical group is confined to a single DSA.

l) **familyRuleViolation** – An operation attempted to break a rule applicable to families within a compound entry.

The **attributeInfo** parameter identifies the particular attribute type(s) and possibly value(s) causing a problem. If an **objectClassViolation** is being reported, an **attribute** item shall be present indicating the **objectClass** attribute type and listing the object class(es) that caused the problem; additional **attributeType** items may also be present (eg. to identify missing mandatory attributes or extraneous attributes).

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

NOTE 2 – The **updateError** is not used to report problems with attribute types, values, or constraint violations encountered in an **addEntry**, **removeEntry**, **modifyEntry**, or **modifyDN** operation. Such problems are reported via an **attributeError**.

The **SecurityParameters** component (see 7.10) shall be included in the **CommonResults** (see 7.4) if the error is to be signed by the Directory.

The information provided by the error problem can optionally be qualified by the use of the **notification** component of **CommonResults**.

13 Analysis of search arguments

This clause is only relevant for a Search operation starting its initial evaluation phase within a service specific administrative area.

This procedure has two purposes:

a) It provides the search-validation function (see 16.12 of ITU-T Rec. X.501 | ISO/IEC 9594-2). However, the search-validation function does not produce error information. If during the procedure an error is encountered, the evaluation stops and returns FALSE, otherwise it returns TRUE. A search-validation against an empty search-rule will always return TRUE.

b) It is the procedure to be used when no governing-search-rule can be located and where it is possible to identify a single search-rule the **SearchArgument** can be evaluated against to identify why the **search** request failed. When an error condition is found in this case, the evaluation stops, the necessary diagnostic information is supplied in the **notification** component of the **CommonResults** data type and a service error with problem **requestedServiceNotAvailable** is returned. What diagnostic information that is included is dependent on the type of error identified.

NOTE – According to the specification above, a search request may be evaluated twice against the same search-rule. How this could be optimized is not part of this specification, but is an implementation decision.

The procedure assumes that an implementation will not allow an invokable search-rule to:

– specify unsupported attribute types, context types, matching rules, matching restrictions, etc.;

– specify mapping-based matching algorithms that are unsupported or not relevant for the type of search for which the search-rule is governing;

– specify matching rule substitutions that would violate the search-rule;

– refer to optional search-rule features not supported by the implementation; or

– be inconsistent or erroneous.

13.1 General check of search filter

The evaluation is performed by first checking whether the filter violates some basic restrictions using the following procedure:

1) If there are attribute types represented in the filter but not represented by any request-attribute-profile in the **inputAttributeTypes** search-rule component, the **notification** shall contain:
   – a **searchServiceProblem** notification attribute with the value **id-pr-searchAttributeViolation**;
   – a **serviceType** notification attribute having as value the **serviceType** component of the search-rule; and
   – an **attributeTypeList** notification attribute having as values the object identifiers identifying the illegal attribute types.

2) If there are attribute types only represented by negated filter items, then the **notification** shall contain:
   – a **searchServiceProblem** notification attribute with the value **id-pr-attributeNegationViolation**;
   – a **serviceType** notification attribute having as value the **serviceType** component of the search-rule; and
   – an **attributeTypeList** notification attribute where the values are the object identifiers identifying the attribute types illegally negated in the filter.

3) Check that the condition specified in the **attributeCombination** is fulfilled with respect to the non-negated presence of attribute types. If mandatory attribute types, i.e., attribute types that unconditionally have to be represented by non-negated filter items in the filter, are missing in any subfilter, the **notification** shall contain:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

– a **searchServiceProblem** notification attribute with the value **id-pr-missingSearchAttribute**;
– a **serviceType** notification attribute having as value the **serviceType** component of the search-rule; and
– an **attributeTypeList** notification attribute having as values the object identifiers identifying the missing attribute types.

If a required combination is not present, the **notification** shall contain:

– a **searchServiceProblem** notification attribute with the value **id-pr-searchAttributeCombinationViolation**;
– a **serviceType** notification attribute having as value the **serviceType** component of the search-rule; and
– an **attributeCombinations** notification attribute identifying the missing combination(s).

4) For request-attribute-profiles having a **selectedValues** subcomponent but the set of values is empty, it is checked whether there is any filter item for those attribute types that does not meet one of the following requirements:
   – the filter item is of type **present** and the **contexts** subcomponent is not present in the request-attribute-profile; or
   – the filter item is of type **contextPresent** and the **contexts** subcomponent is present in the request-attribute-profile.

   If the above check fails for any filter item, the **notification** shall contain:
   – a **searchServiceProblem** notification attribute with the value **id-pr-searchValueNotAllowed**;
   – a **serviceType** notification attribute having as value the **serviceType** component of the search-rule; and
   – a **filterItem** notification attribute with the failing filters items as values.
5) For request-attribute-profiles having a **contexts** subcomponent, it is checked whether there are any filter items that refer to context types not included in this subcomponent. If so, the **notification** shall contain:
   – a **searchServiceProblem** notification attribute with the value **id-pr-searchContextViolation**;
   – a **serviceType** notification attribute having as value the **serviceType** component of the search-rule; and
   – a **contextTypeList** notification attribute having as values the object identifiers for the illegal context types.
6) If the **allowed** choice for the **subset** component is taken in the search-rule, it is checked whether the **subset** argument of the **SearchArgument** complies with that specification. If not, the **notification** shall contain:
   – a **searchServiceProblem** notification attribute with the value **id-pr-searchSubsetViolation**; and
   – a **serviceType** notification attribute having as value the **serviceType** component of the search-rule.

13.2 Check of request-attribute-profiles

If the above procedure did not yield any error, it has to be checked for each subfilter that any attribute type represented in that subfilter is also effectively present. This procedure does not specify any order in which subfilters should be evaluated. For an attribute type to be effectively present in a subfilter, it has to be represented by at least one non-negated filter item that complies with the corresponding request-attribute-profile. A non-negated filter item is evaluated using the below procedure.

The non-negated filter items are checked in the following order:

1) the filter items for the attribute types that unconditionally have to be represented are checked for each subfilter;
2) the filter items for the attribute types that conditionally have to be represented are checked for each subfilter; and
3) the remaining filter items are checked for each subfilter.

If a subfilter fails the evaluation, the evaluation stops and error information is returned as detailed below.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

If an attribute type in a subfilter is represented by several non-negated filter items, each such filter item is in principle checked until either a complying filter item is found or all filter items are checked. If a filter item fails during the procedure, it is dropped for further evaluation. It is the last filter item to fail for the attribute type that determines the diagnostic information returned.

A filter item is evaluated using the following procedure:

1) If the **selectedValues** component in the request-attribute-profile is absent; or if it present and non-empty, check whether the filter item is of type **equality**, **substrings**, **approximateMatch** or **extensibleMatch**. If not, the **notification** shall contain:
   - a **searchServiceProblem** notification attribute with the value **id-pr-searchValueRequired**;
   - a **serviceType** notification attribute having as value the **serviceType** component of the search-rule; and
   - an **attributeTypeList** notification attribute having as value the object identifier identifying the attribute type from the filter item.
2) If the **selectedValues** subcomponent in the corresponding request-attribute-profile is present and non-empty, check whether the filter item fails to match any value specified in that subcomponent. If so, the **notification** shall contain:
   - a **searchServiceProblem** notification attribute with the value **id-pr-invalidSearchValue**;
   - a **serviceType** notification attribute having as value the **serviceType** component of the search-rule; and
   - a **filterItem** notification attribute with the failing filter item as the only value.
3) If the **contexts** subcomponent is not present, continue with next subclause.
4) Check that the condition specified in the **contextCombination** subcomponent is fulfilled with respect to the presence of context types. If mandatory context types, i.e., context types that unconditionally have to be represented for the attribute type, are missing, the **notification** shall contain:
   - a **searchServiceProblem** notification attribute with the value **id-pr-missingSearchContext**;
   - a **serviceType** notification attribute having as value the **serviceType** component of the search-rule;
   - an **attributeTypeList** notification attribute having as a single value the object identifier identifying the attribute type from the filter item.
   - a **contextTypeList** notification attribute with the object identifiers identifying the missing context types.

   If a required combination is not present, the **notification** shall contain:
   - a **searchServiceProblem** notification attribute with the value **id-pr-searchContextCombinationViolation**;
   - a **serviceType** notification attribute having as value the **serviceType** component of the search-rule;
   - an **attributeTypeList** notification attribute having as only value the object identifier identifying the attribute type from the filter item.
   - a **contextCombinations** notification attribute identifying the missing combination(s).
5) Check if the context assertions for the attribute type in the subfilter are all included in the **contexts** subcomponent. If not, the **notification** shall contain:
   - a **searchServiceProblem** notification attribute with the value **id-pr-searchContextViolation**;
   - a **serviceType** notification attribute having as value the **serviceType** component of the search-rule;
   - an **attributeTypeList** notification attribute having as only value the object identifier identifying the attribute type from the filter item; and
   - a **contextTypeList** notification attribute having as values the object identifiers identifying the context types not allowed for the attribute type.
6) If context values are included for any of the context types in the **contexts** subcomponent of the request-attribute-profile, check whether any of the context assertions specified for the attribute type in the subfilter contains values not specified for the corresponding context types in **contexts** subcomponent. If so, the **notification** shall contain:
   - a **searchServiceProblem** notification attribute with the value **id-pr-searchContextValueViolation**;
   - a **serviceType** notification attribute having as value the **serviceType** component of the search-rule;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

- an **attributeTypeList** notification attribute having as only value the object identifier identifying the attribute type from the filter item; and
- a **contextList** notification attribute having as values the context assertions not allowed for the attribute type.

13.3 Check of controls and hierarchy selections

If the search request fails the test against the control and hierarchy selections as specified in 16.10.5 of ITU-T Rec. X.501 | ISO/IEC 9594-2, the procedure in this subclause is performed.

1) If the **defaultControls** component of the search-rule or the **hierarchyOptions** subcomponent of the **defaultControls** is absent, and the search request specifies hierarchy selections beside **self**, then the **notification** shall contain:
   - a **searchServiceProblem** notification attribute with the value **id-pr-hierarchySelectForbidden**; and
   - a **serviceType** notification attribute having as value the **serviceType** component of the search-rule.
2) If there are hierarchy select options in the request that are not allowed, or some selections are missing according to the combination of the **defaultControls** and **mandatoryControls** components of the search-rule, then the **notification** shall contain:
   - a **searchServiceProblem** notification attribute with the value **id-pr-invalidHierarchySelect**;
   - a **serviceType** notification attribute having as value the **serviceType** component of the search-rule; and
   - a **hierarchySelectList** notification attribute having as value a bitstring identifying the invalid hierarchy selection options.
3) If there are hierarchy select options in the request that are not supported by the DSA and which are not covered by 2), then the **notification** shall contain:
   - a **searchServiceProblem** notification attribute with the value **id-pr-unavailableHierarchySelect;**
   - a **serviceType** notification attribute having as value the **serviceType** component of the search-rule; and
   - a **hierarchySelectList** notification attribute having as value a bitstring identifying the unsupported hierarchy selection options.
4) If there are search control options (as defined by 10.2.1) in the request that are not allowed or some options are missing according to the combination of the **defaultControls** and **mandatoryControls** components of the search-rule, then the **notification** shall contain:
   - a **searchServiceProblem** notification attribute with the value **id-pr-invalidSearchControlOptions**;
   - a **serviceType** notification attribute having as value the **serviceType** component of the search-rule; and
   - a **searchControlOptionsList** notification attribute having as value a bitstring identifying the invalid search control options.
5) If there are service control options in the request not allowed or some options are missing according to the combination of the **defaultControls** and **mandatoryControls** components of the search-rule, then the **notification** shall contain:
   - a **searchServiceProblem** notification attribute with the value **id-pr-invalidServiceControlOptions**;
   - a **serviceType** notification attribute having as value the **serviceType** component of the search-rule; and
   - a **serviceControlOptionsList** notification attribute having as value a bitstring identifying the invalid service control options.

13.4 Check of matching use

In the search-validation procedure, this subclause represents the last step in the validation and it is assumed that the **search** request has passed all other validation steps. A search-rule failing this last step is put on the **MatchProblemSR** list (see 19.3.2.2.1, item 3) of ITU-T Rec. X.518 | ISO/IEC 9594-4).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

If the search request does not comply with the **matchingUse** requirement as specified in 16.10.2 of ITU-T Rec. X.501 | ISO/IEC 9594-2 for any of the request-attribute-profiles, then a **notification** for one of the failing request-attribute-profile shall contain:

- a **searchServiceProblem** notification attribute with the value **id-pr-attributeMatchingViolation** if the matching restriction is violated, or with the value **id-pr-unsupportedMatchingUse** if the matching rule is to be applied in an unsupported way;
- a **serviceType** notification attribute having as value the **serviceType** component of the search-rule;
- an **attributeTypeList** notification attribute having as only value the object identifier identifying the attribute type; and
- for the matching restriction that is violated, additional notification attributes as specified by specification for that matching restriction.

NOTE – When several request-attribute-profiles fail the validation, it is a local matter to select which one for which to create a **notification**.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Annex A

Abstract Service in ASN.1

(This annex forms an integral part of this Recommendation | International Standard)

This annex includes all of the ASN.1 type, value and information object definitions contained in this Directory Specification in the form of the ASN.1 module **DirectoryAbstractService**.

```
DirectoryAbstractService {joint-iso-itu-t ds(5) module(1) directoryAbstractService(2) 6}

DEFINITIONS ::=

BEGIN

-- EXPORTS All --

-- The types and values defined in this module are exported for use in the other ASN.1 modules contained
-- within the Directory Specifications, and for the use of other applications which will use them to access
-- Directory services. Other applications may use them for their own purposes, but this will not constrain
-- extensions and modifications needed to maintain or improve the Directory service.

IMPORTS

-- from ITU-T Rec. X.501 | ISO/IEC 9594-2

    attributeCertificateDefinitions, authenticationFramework, basicAccessControl,
    commonProtocolSpecification, directoryShadowAbstractService, distributedOperations,
    enhancedSecurity, id-at, informationFramework, selectedAttributeTypes, serviceAdministration
        FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 6}

    Attribute{}, ATTRIBUTE, AttributeType, AttributeTypeAssertion, AttributeValue,
    AttributeValueAssertion, CONTEXT, ContextAssertion, DistinguishedName,
    MATCHING-RULE, Name, OBJECT-CLASS, RelativeDistinguishedName,
    SupportedAttributes, SupportedContexts
        FROM  InformationFramework informationFramework

    RelaxationPolicy
        FROM  ServiceAdministration serviceAdministration

    AttributeTypeAndValue
        FROM BasicAccessControl  basicAccessControl

    OPTIONALLY-PROTECTED{ }, OPTIONALLY-PROTECTED-SEQ{ }
        FROM EnhancedSecurity enhancedSecurity

-- from ITU-T Rec. X.518 | ISO/IEC 9594-4
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        AccessPoint, ContinuationReference, Exclusions, OperationProgress, ReferenceType
                FROM  DistributedOperations  distributedOperations

-- from ITU-T Rec. X.519 | ISO/IEC 9594-5

        Code, ERROR, id-errcode-abandoned, id-errcode-abandonFailed,  id-errcode-attributeError,
        id-errcode-nameError, id-errcode-referral, id-errcode-securityError, id-errcode-serviceError,
        id-errcode-updateError, id-opcode-abandon, id-opcode-addEntry, id-opcode-compare,
        id-opcode-list,  id-opcode-modifyDN, id-opcode-modifyEntry, id-opcode-read,
        id-opcode-removeEntry, id-opcode-search, InvokeId, OPERATION
                FROM CommonProtocolSpecification commonProtocolSpecification

-- from ITU-T Rec. X.520 | ISO/IEC 9594-6

        DirectoryString {}, UnboundedDirectoryString
                FROM SelectedAttributeTypes selectedAttributeTypes

-- from ITU-T Rec. X.509 | ISO/IEC 9594-8

        AlgorithmIdentifier{}, CertificationPath, ENCRYPTED {}, HASH {}, SIGNED {}, SupportedAlgorithms
                FROM AuthenticationFramework authenticationFramework

        AttributeCertificationPath
                FROM AttributeCertificateDefinitions attributeCertificateDefinitions

-- from ITU-T Rec. X.525 | ISO/IEC 9594-9

        AgreementID
                FROM DirectoryShadowAbstractService  directoryShadowAbstractService

-- from RFC 2025

        SPKM-ERROR, SPKM-REP-TI, SPKM-REQ
                FROM SpkmGssTokens { iso (1) identified-organization (3) dod(6) internet (1)
                        security (5) mechanisms (5) spkm (1) spkmGssTokens (10) }    ;

-- Common data types --

CommonArguments  ::=  SET {
        serviceControls                 [30]    ServiceControls                 DEFAULT { },
        securityParameters              [29]    SecurityParameters    OPTIONAL,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        requestor                       [28]  DistinguishedName     OPTIONAL,
        operationProgress       [27]  OperationProgress
                                              DEFAULT { nameResolutionPhase notStarted },
        aliasedRDNs             [26]  INTEGER                       OPTIONAL,
        criticalExtensions      [25]  BIT STRING                    OPTIONAL,
        referenceType           [24]  ReferenceType                 OPTIONAL,
        entryOnly                       [23]  BOOLEAN                       DEFAULT TRUE,
        exclusions              [22]  Exclusions            OPTIONAL,
        nameResolveOnMaster     [21]  BOOLEAN                       DEFAULT FALSE,
        operationContexts       [20]  ContextSelection      OPTIONAL,
        familyGrouping                  [19]  FamilyGrouping                DEFAULT entryOnly }

FamilyGrouping ::= ENUMERATED {
        entryOnly               (1),
        compoundEntry           (2),
        strands                 (3),
        multiStrand             (4) }

CommonResults ::= SET {
        securityParameters      [30]  SecurityParameters                    OPTIONAL,
        performer                       [29]  DistinguishedName                     OPTIONAL,
        aliasDereferenced       [28]  BOOLEAN                               DEFAULT FALSE,
        notification            [27]  SEQUENCE SIZE (1..MAX) OF Attribute{{SupportedAttributes}}  OPTIONAL }

CommonResultsSeq ::= SEQUENCE {
        securityParameters      [30]  SecurityParameters                    OPTIONAL,
        performer                       [29]  DistinguishedName                     OPTIONAL,
        aliasDereferenced       [28]  BOOLEAN                               DEFAULT FALSE,
        notification            [27]  SEQUENCE SIZE (1..MAX) OF Attribute{{SupportedAttributes}}  OPTIONAL }

ServiceControls ::= SET {
        options                 [0]   ServiceControlOptions DEFAULT { },
        priority                        [1]   INTEGER { low (0),  medium (1),  high (2) }  DEFAULT medium,
        timeLimit                       [2]   INTEGER  OPTIONAL,
        sizeLimit                       [3]   INTEGER  OPTIONAL,
        scopeOfReferral                 [4]   INTEGER { dmd(0), country(1) } OPTIONAL,
        attributeSizeLimit      [5]   INTEGER  OPTIONAL,
        manageDSAITPlaneRef             [6]   SEQUENCE {
                dsaName                         Name,
                agreementID                     AgreementID } OPTIONAL,
        serviceType             [7]   OBJECT IDENTIFIER OPTIONAL,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        userClass                       [8]     INTEGER OPTIONAL }

ServiceControlOptions  ::=  BIT STRING {
                preferChaining                  (0),
                chainingProhibited      (1),
                localScope                      (2),
                dontUseCopy             (3),
                dontDereferenceAliases  (4),
                subentries                      (5),
                copyShallDo             (6),
                partialNameResolution   (7),
                manageDSAIT                     (8),
                noSubtypeMatch                  (9),
                noSubtypeSelection      (10),
                countFamily             (11),
                dontSelectFriends       (12),
                dontMatchFriends        (13) }

EntryInformationSelection  ::=  SET  {
        attributes                      CHOICE {
                allUserAttributes       [0]     NULL,
                select                  [1]     SET OF AttributeType
                -- empty set implies no attributes are requested --  } DEFAULT allUserAttributes : NULL,
        infoTypes               [2]     INTEGER {
                attributeTypesOnly              (0),
                attributeTypesAndValues         (1) } DEFAULT attributeTypesAndValues,
        extraAttributes                 CHOICE {
                allOperationalAttributes [3]    NULL,
                select                  [4]     SET SIZE (1..MAX) OF AttributeType } OPTIONAL,
        contextSelection            ContextSelection OPTIONAL,
        returnContexts                  BOOLEAN DEFAULT FALSE,
        familyReturn                FamilyReturn  DEFAULT
                                                { memberSelect  contributingEntriesOnly } }

ContextSelection  ::=  CHOICE  {
        allContexts                 NULL,
        selectedContexts        SET SIZE (1..MAX) OF TypeAndContextAssertion }

TypeAndContextAssertion  ::=  SEQUENCE  {
        type                        AttributeType,
        contextAssertions       CHOICE {
                preference              SEQUENCE OF ContextAssertion,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        all                         SET OF ContextAssertion } }

FamilyReturn ::= SEQUENCE {
    memberSelect ENUMERATED {
        contributingEntriesOnly     (1),
        participatingEntriesOnly    (2),
        compoundEntry               (3) },
    familySelect    SEQUENCE SIZE (1..MAX) OF OBJECT-CLASS.&id   OPTIONAL }

EntryInformation  ::=  SEQUENCE  {
    name                        Name,
    fromEntry                       BOOLEAN DEFAULT TRUE,
    information                 SET SIZE (1..MAX) OF CHOICE {
        attributeType               AttributeType,
        attribute                       Attribute{{SupportedAttributes}} } OPTIONAL,
    incompleteEntry         [3]     BOOLEAN DEFAULT FALSE, -- not in first edition systems
    partialName         [4]     BOOLEAN DEFAULT FALSE, -- not in first or second edition systems
    derivedEntry        [5]     BOOLEAN DEFAULT FALSE -- not in pre-fourth edition systems -- }

family-information  ATTRIBUTE  ::=  {
    WITH SYNTAX                 FamilyEntries
    USAGE                       directoryOperation
    ID                          id-at-family-information }

FamilyEntries  ::=  SEQUENCE  {
    family-class            OBJECT-CLASS.&id,       -- structural object class value
    familyEntries           SEQUENCE OF FamilyEntry }

FamilyEntry  ::=  SEQUENCE  {
    rdn                         RelativeDistinguishedName,
    information                 SEQUENCE OF CHOICE  {
        attributeType           AttributeType,
        attribute                   Attribute{{SupportedAttributes}} },
    family-info                 SEQUENCE SIZE (1..MAX) OF FamilyEntries OPTIONAL }

Filter  ::=  CHOICE  {
    item    [0]     FilterItem,
    and     [1]     SET OF Filter,
    or      [2]     SET OF Filter,
    not     [3]     Filter }
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
FilterItem ::= CHOICE {
    equality                [0] AttributeValueAssertion,
    substrings              [1] SEQUENCE {
        type                        ATTRIBUTE.&id ({ SupportedAttributes }),
        strings                     SEQUENCE OF CHOICE {
            initial             [0] ATTRIBUTE.&Type
                                        ({SupportedAttributes}{@substrings.type}),
            any                 [1] ATTRIBUTE.&Type
                                        ({SupportedAttributes}{@substrings.type}),
            final               [2] ATTRIBUTE.&Type
                                        ({SupportedAttributes}{@substrings.type}),
            control                 Attribute{{SupportedAttributes}} } },   -- Used to specify
interpretation of following
                                                                            -- items
    greaterOrEqual          [2] AttributeValueAssertion,
    lessOrEqual         [3] AttributeValueAssertion,
    present                 [4] AttributeType,
    approximateMatch    [5] AttributeValueAssertion,
    extensibleMatch         [6] MatchingRuleAssertion,
    contextPresent          [7] AttributeTypeAssertion }

MatchingRuleAssertion ::= SEQUENCE {
    matchingRule [1]    SET SIZE (1..MAX) OF MATCHING-RULE.&id,
    type                [2] AttributeType  OPTIONAL,
    matchValue  [3]     MATCHING-RULE.&AssertionType ( CONSTRAINED BY {
                -- matchValue shall be a value of  type specified by the &AssertionType field of
                -- one of the MATCHING-RULE information objects identified by matchingRule -- } ),
    dnAttributes [4]    BOOLEAN DEFAULT FALSE }

PagedResultsRequest ::= CHOICE {
    newRequest              SEQUENCE {
        pageSize                    INTEGER,
        sortKeys                    SEQUENCE SIZE (1..MAX) OF SortKey OPTIONAL,
        reverse             [1] BOOLEAN DEFAULT FALSE,
        unmerged                [2] BOOLEAN DEFAULT FALSE,
        pageNumber              [3] INTEGER OPTIONAL },
    queryReference              OCTET STRING,
    abandonQuery            [0] OCTET STRING }

SortKey ::= SEQUENCE {
    type                AttributeType,
    orderingRule  MATCHING-RULE.&id OPTIONAL }
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
SecurityParameters ::= SET {
        certification-path      [0]  CertificationPath          OPTIONAL,
        name                    [1]  DistinguishedName          OPTIONAL,
        time                    [2]  Time                       OPTIONAL,
        random                  [3]  BIT STRING                 OPTIONAL,
        target                  [4]  ProtectionRequest          OPTIONAL,
        response                [5]  BIT STRING                 OPTIONAL,
        operationCode           [6]  Code                       OPTIONAL,
        attributeCertificationPath [7] AttributeCertificationPath OPTIONAL,
        errorProtection         [8]  ErrorProtectionRequest     OPTIONAL,
        errorCode               [9]  Code                       OPTIONAL }

ProtectionRequest ::= INTEGER { none (0), signed (1) }

Time ::= CHOICE {
        utcTime                 UTCTime,
        generalizedTime         GeneralizedTime }

ErrorProtectionRequest ::= INTEGER { none (0), signed (1) }

-- Bind and unbind operations --

directoryBind OPERATION ::= {
        ARGUMENT DirectoryBindArgument
        RESULT          DirectoryBindResult
        ERRORS          { directoryBindError } }

DirectoryBindArgument ::= SET {
        credentials     [0]     Credentials OPTIONAL,
        versions        [1]     Versions DEFAULT {v1} }

Credentials ::= CHOICE {
        simple          [0]     SimpleCredentials,
        strong          [1]     StrongCredentials,
        externalProcedure [2]   EXTERNAL,
        spkm            [3]     SpkmCredentials,
        sasl                    [4]     SaslCredentials }

SimpleCredentials ::= SEQUENCE {
        name            [0]     DistinguishedName,
        validity        [1]     SET {
```


--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        time1           [0]     CHOICE {
                utc                     UTCTime,
                gt                      GeneralizedTime } OPTIONAL,
        time2           [1]     CHOICE {
                utc                     UTCTime,
                gt                      GeneralizedTime } OPTIONAL,
        random1         [2]     BIT STRING  OPTIONAL,
        random2         [3]     BIT STRING  OPTIONAL } OPTIONAL,
    password            [2]     CHOICE {
        unprotected             OCTET STRING,
        protected               HASH {OCTET STRING} } OPTIONAL}

StrongCredentials ::= SET {
    certification-path          [0]     CertificationPath OPTIONAL,
    bind-token                  [1]     Token,
    name                        [2]     DistinguishedName OPTIONAL,
    attributeCertificationPath  [3]     AttributeCertificationPath OPTIONAL }

SpkmCredentials ::= CHOICE {
    req     [0]     SPKM-REQ,
    rep     [1]     SPKM-REP-TI }

SaslCredentials ::= SEQUENCE {
    mechanism       [0]     DirectoryString { ub-saslMechanism },
    credentials     [1]     OCTET STRING OPTIONAL,
    saslAbort       [2]     BOOLEAN DEFAULT FALSE }

ub-saslMechanism INTEGER ::= 20     -- According to RFC 2222

Token ::= SIGNED { TokenContent }

TokenContent ::= SEQUENCE {
    algorithm               [0]     AlgorithmIdentifier{{SupportedAlgorithms}},
    name            [1]     DistinguishedName,
    time                    [2]     Time,
    random                  [3]     BIT STRING,
    response                [4]     BIT STRING OPTIONAL }

Versions ::= BIT STRING {v1(0), v2(1) }

DirectoryBindResult ::= DirectoryBindArgument
```


--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
directoryBindError ERROR ::= {
        PARAMETER               OPTIONALLY-PROTECTED {
                SET {
                        versions                [0]     Versions DEFAULT {v1},
                        error                           CHOICE {
                                serviceError    [1]     ServiceProblem,
                                securityError   [2]     SecurityProblem },
                        secutityParameters      [30]    SecurityParameters OPTIONAL } } }

BindKeyInfo ::= ENCRYPTED { BIT STRING }

-- Operations, arguments, and results --

read OPERATION ::= {
        ARGUMENT ReadArgument
        RESULT          ReadResult
        ERRORS          { attributeError | nameError | serviceError | referral | abandoned |
                        securityError }
        CODE            id-opcode-read }

ReadArgument ::= OPTIONALLY-PROTECTED {
        SET {
                object                  [0]     Name,
                selection                       [1]     EntryInformationSelection DEFAULT { },
                modifyRightsRequest     [2]     BOOLEAN DEFAULT FALSE,
                COMPONENTS OF                   CommonArguments } }

ReadResult ::= OPTIONALLY-PROTECTED {
        SET {
                entry           [0]     EntryInformation,
                modifyRights    [1]     ModifyRights OPTIONAL,
                COMPONENTS OF   CommonResults } }

ModifyRights ::= SET OF SEQUENCE {
        item                            CHOICE {
                entry           [0]     NULL,
                attribute               [1]     AttributeType,
                value           [2]     AttributeValueAssertion },
        permission              [3]     BIT STRING { add (0), remove (1), rename (2), move (3) } }

compare OPERATION ::= {
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        ARGUMENT CompareArgument
        RESULT          CompareResult
        ERRORS          { attributeError | nameError | serviceError | referral | abandoned |
                        securityError }
        CODE        id-opcode-compare }

CompareArgument ::= OPTIONALLY-PROTECTED {
        SET {
            object                  [0]     Name,
            purported               [1]     AttributeValueAssertion,
            COMPONENTS OF           CommonArguments } }

CompareResult ::= OPTIONALLY-PROTECTED {
        SET {
            name                            Name OPTIONAL,
            matched                 [0]     BOOLEAN,
            fromEntry               [1]     BOOLEAN DEFAULT TRUE,
            matchedSubtype          [2]     AttributeType OPTIONAL,
            COMPONENTS OF           CommonResults } }

abandon OPERATION ::= {
        ARGUMENT AbandonArgument
        RESULT          AbandonResult
        ERRORS          { abandonFailed }
        CODE        id-opcode-abandon }

AbandonArgument ::= OPTIONALLY-PROTECTED-SEQ {
        SEQUENCE {
            invokeID            [0]     InvokeId } }

AbandonResult ::= CHOICE {
        null            NULL,
        information     OPTIONALLY-PROTECTED-SEQ {
            SEQUENCE {
                invokeID                InvokeId,
                COMPONENTS OF   CommonResultsSeq } } }

list    OPERATION ::= {
        ARGUMENT ListArgument
        RESULT          ListResult
        ERRORS          { nameError | serviceError | referral | abandoned | securityError }
        CODE        id-opcode-list }
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
ListArgument ::= OPTIONALLY-PROTECTED {
        SET {
                object                  [0]     Name,
                pagedResults            [1]     PagedResultsRequest OPTIONAL,
                listFamily                      [2]     BOOLEAN DEFAULT FALSE,
                COMPONENTS OF           CommonArguments } }

ListResult ::= OPTIONALLY-PROTECTED {
        CHOICE {
                listInfo                        SET {
                        name                            Name  OPTIONAL,
                        subordinates                    [1]     SET OF SEQUENCE {
                                rdn                                     RelativeDistinguishedName,
                                aliasEntry                      [0]     BOOLEAN DEFAULT  FALSE,
                                fromEntry                       [1]     BOOLEAN DEFAULT  TRUE },
                        partialOutcomeQualifier [2]     PartialOutcomeQualifier OPTIONAL,
                        COMPONENTS OF                   CommonResults },
                uncorrelatedListInfo  [0]       SET OF ListResult } }

PartialOutcomeQualifier ::= SET {
        limitProblem                    [0]     LimitProblem  OPTIONAL,
        unexplored                      [1]     SET SIZE (1..MAX) OF ContinuationReference  OPTIONAL,
        unavailableCriticalExtensions   [2]     BOOLEAN DEFAULT  FALSE,
        unknownErrors                           [3]     SET SIZE (1..MAX) OF ABSTRACT-SYNTAX.&Type
OPTIONAL,
        queryReference                          [4]     OCTET STRING  OPTIONAL,
        overspecFilter                  [5]     Filter OPTIONAL,
        notification                    [6]     SEQUENCE SIZE (1 .. MAX) OF
                                                        Attribute{{SupportedAttributes}}  OPTIONAL,
        entryCount                              CHOICE {
                bestEstimate                    [7]     INTEGER,
                lowEstimate                     [8]     INTEGER,
                exact                           [9]     INTEGER } OPTIONAL,
        streamedResult                          [10]    BOOLEAN DEFAULT  FALSE }

LimitProblem ::= INTEGER {
        timeLimitExceeded (0), sizeLimitExceeded (1), administrativeLimitExceeded (2) }

search OPERATION ::= {
        ARGUMENT SearchArgument
        RESULT          SearchResult
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        ERRORS          { attributeError | nameError | serviceError | referral | abandoned |
                          securityError }
        CODE        id-opcode-search }

SearchArgument ::= OPTIONALLY-PROTECTED {
        SET {
            baseObject          [0]     Name,
            subset              [1]     INTEGER {
                baseObject(0), oneLevel(1), wholeSubtree(2) } DEFAULT baseObject,
            filter                      [2]     Filter DEFAULT and : { },
            searchAliases       [3]     BOOLEAN DEFAULT TRUE,
            selection                   [4]     EntryInformationSelection DEFAULT { },
            pagedResults        [5]     PagedResultsRequest OPTIONAL,
            matchedValuesOnly   [6]     BOOLEAN DEFAULT FALSE,
            extendedFilter      [7]     Filter OPTIONAL,
            checkOverspecified  [8]     BOOLEAN DEFAULT FALSE,
            relaxation                  [9]     RelaxationPolicy            OPTIONAL,
            extendedArea        [10]    INTEGER                     OPTIONAL,
            hierarchySelections [11]    HierarchySelections DEFAULT { self },
            searchControlOptions        [12]    SearchControlOptions DEFAULT { searchAliases },
            joinArguments               [13]    SEQUENCE SIZE (1..MAX) OF JoinArgument OPTIONAL,
            joinType                    [14]    ENUMERATED {
                innerJoin(0), leftOuterJoin(1), fullOuterJoin(2) } DEFAULT leftOuterJoin,
            COMPONENTS OF       CommonArguments } }

HierarchySelections ::= BIT STRING {
        self            (0),
        children        (1),
        parent      (2),
        hierarchy       (3),
        top             (4),
        subtree         (5),
        siblings        (6),
        siblingChildren (7),
        siblingSubtree (8),
        all             (9) }

SearchControlOptions ::= BIT STRING {
        searchAliases       (0),
        matchedValuesOnly   (1),
        checkOverspecified  (2),
        performExactly              (3),
```


--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
        includeAllAreas                 (4),
        noSystemRelaxation      (5),
        dnAttribute                     (6),
        matchOnResidualName             (7),
        entryCount                      (8),
        useSubset                       (9),
        separateFamilyMembers   (10),
        searchFamily            (11) }

JoinArgument ::= SEQUENCE {
        joinBaseObject          [0]     Name,
        domainLocalID           [1]     DomainLocalID OPTIONAL,
        joinSubset              [2]     ENUMERATED {
                baseObject(0), oneLevel(1), wholeSubtree(2) } DEFAULT baseObject,
        joinFilter              [3]     Filter  OPTIONAL,
        joinAttributes  [4]     SEQUENCE SIZE (1..MAX) OF JoinAttPair OPTIONAL,
        joinSelection   [5]     EntryInformationSelection }

DomainLocalID ::= UnboundedDirectoryString

JoinAttPair ::= SEQUENCE {
        baseAtt                 AttributeType,
        joinAtt         AttributeType,
        joinContext     SEQUENCE SIZE (1..MAX) OF JoinContextType OPTIONAL }

JoinContextType ::= CONTEXT.&id({SupportedContexts})

SearchResult ::= OPTIONALLY-PROTECTED {
        CHOICE {
                searchInfo                      SET {
                        name                            Name  OPTIONAL,
                        entries                 [0]     SET OF EntryInformation,
                        partialOutcomeQualifier [2]     PartialOutcomeQualifier OPTIONAL,
                        altMatching             [3]     BOOLEAN  DEFAULT FALSE,
                        COMPONENTS OF                   CommonResults },
                uncorrelatedSearchInfo  [0]     SET OF SearchResult } }

addEntry OPERATION ::= {
        ARGUMENT AddEntryArgument
        RESULT          AddEntryResult
        ERRORS          { attributeError | nameError | serviceError | referral | securityError |
```


--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
                        updateError }
    CODE        id-opcode-addEntry }

AddEntryArgument ::= OPTIONALLY-PROTECTED {
    SET {
        object              [0]     Name,
        entry               [1]     SET OF Attribute{{SupportedAttributes}},
        targetSystem        [2]     AccessPoint  OPTIONAL,
        COMPONENTS OF               CommonArguments } }

AddEntryResult ::= CHOICE {
    null                    NULL,
    information             OPTIONALLY-PROTECTED-SEQ {
                            SEQUENCE { COMPONENTS OF CommonResultsSeq } } }

removeEntry OPERATION ::= {
    ARGUMENT RemoveEntryArgument
    RESULT              RemoveEntryResult
    ERRORS              { nameError | serviceError | referral | securityError | updateError }
    CODE        id-opcode-removeEntry }

RemoveEntryArgument ::= OPTIONALLY-PROTECTED {
    SET {
        object              [0]     Name,
        COMPONENTS OF               CommonArguments } }

RemoveEntryResult ::= CHOICE {
    null                    NULL,
    information     OPTIONALLY-PROTECTED-SEQ {
        SEQUENCE { COMPONENTS OF CommonResultsSeq } } }

modifyEntry OPERATION ::= {
    ARGUMENT ModifyEntryArgument
    RESULT              ModifyEntryResult
    ERRORS              { attributeError | nameError | serviceError | referral | securityError |
                        updateError }
    CODE        id-opcode-modifyEntry }

ModifyEntryArgument ::= OPTIONALLY-PROTECTED {
    SET {
        object              [0]     Name,
        changes                     [1]     SEQUENCE OF EntryModification,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        selection                   [2]     EntryInformationSelection OPTIONAL,
        COMPONENTS OF               CommonArguments } }

ModifyEntryResult ::= CHOICE {
    null                NULL,
    information     OPTIONALLY-PROTECTED-SEQ {
        SEQUENCE {
            entry           [0]     EntryInformation OPTIONAL,
            COMPONENTS OF   CommonResultsSeq } } }

EntryModification   ::=     CHOICE {
    addAttribute        [0]     Attribute{{SupportedAttributes}},
    removeAttribute             [1]     AttributeType,
    addValues           [2]     Attribute{{SupportedAttributes}},
    removeValues        [3]     Attribute{{SupportedAttributes}},
    alterValues                 [4]     AttributeTypeAndValue,
    resetValue          [5]     AttributeType,
    replaceValues       [6]     Attribute{{SupportedAttributes}} }

modifyDN OPERATION ::= {
    ARGUMENT ModifyDNArgument
    RESULT          ModifyDNResult
    ERRORS          { nameError | serviceError | referral | securityError | updateError }
    CODE        id-opcode-modifyDN }

ModifyDNArgument ::= OPTIONALLY-PROTECTED {
    SET {
        object          [0]     DistinguishedName,
        newRDN              [1]     RelativeDistinguishedName,
        deleteOldRDN[2]     BOOLEAN DEFAULT FALSE,
        newSuperior [3]     DistinguishedName OPTIONAL,
                                COMPONENTS OF CommonArguments } }

ModifyDNResult ::= CHOICE {
    null                NULL,
    information     OPTIONALLY-PROTECTED-SEQ {
        SEQUENCE {
            newRDN                  RelativeDistinguishedName,
            COMPONENTS OF   CommonResultsSeq } } }

-- Errors and parameters --
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
abandoned ERROR ::= { -- not literally an "error"
        PARAMETER       OPTIONALLY-PROTECTED {
                SET {
                        COMPONENTS OF   CommonResults} }
        CODE            id-errcode-abandoned }

abandonFailed ERROR ::= {
        PARAMETER       OPTIONALLY-PROTECTED {
                SET {
                        problem                 [0]     AbandonProblem,
                        operation               [1]     InvokeId,
                        COMPONENTS OF           CommonResults } }
        CODE            id-errcode-abandonFailed }

AbandonProblem ::= INTEGER { noSuchOperation (1), tooLate (2), cannotAbandon (3) }

attributeError ERROR ::= {
        PARAMETER       OPTIONALLY-PROTECTED { SET {
                object [0]      Name,
                problems        [1]     SET OF SEQUENCE {
                        problem         [0]     AttributeProblem,
                        type            [1]     AttributeType,
                        value   [2]     AttributeValue  OPTIONAL },
                COMPONENTS OF   CommonResults } }
        CODE            id-errcode-attributeError }

AttributeProblem ::= INTEGER {
        noSuchAttributeOrValue          (1),
        invalidAttributeSyntax                  (2),
        undefinedAttributeType          (3),
        inappropriateMatching                   (4),
        constraintViolation             (5),
        attributeOrValueAlreadyExists   (6),
        contextViolation                        (7) }

nameError ERROR ::= {
        PARAMETER       OPTIONALLY-PROTECTED { SET {
                problem                 [0]     NameProblem,
                matched                 [1]     Name,
                COMPONENTS OF           CommonResults } }
        CODE            id-errcode-nameError }
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
NameProblem ::= INTEGER {
        noSuchObject                (1),
        aliasProblem                (2),
        invalidAttributeSyntax      (3),
        aliasDereferencingProblem   (4),
        contextProblem              (5) }

referral ERROR ::= { -- not literally an "error"
        PARAMETER       OPTIONALLY-PROTECTED { SET {
                candidate               [0]     ContinuationReference,
                COMPONENTS OF           CommonResults } }
        CODE            id-errcode-referral }

securityError ERROR ::= {
        PARAMETER       OPTIONALLY-PROTECTED { SET {
                problem         [0]     SecurityProblem,
                spkmInfo        [1]     SPKM-ERROR,
                                        COMPONENTS OF CommonResults } }
        CODE            id-errcode-securityError }

SecurityProblem ::= INTEGER {
        inappropriateAuthentication (1),
        invalidCredentials          (2),
        insufficientAccessRights    (3),
        invalidSignature            (4),
        protectionRequired          (5),
        noInformation               (6),
        blockedCredentials          (7),
--      invalidQOPMatch             (8), obsolete
        spkmError                   (9) }

serviceError ERROR ::= {
        PARAMETER       OPTIONALLY-PROTECTED { SET {
                problem         [0]     ServiceProblem,
                COMPONENTS OF           CommonResults } }
        CODE            id-errcode-serviceError }

ServiceProblem ::= INTEGER {
        busy                        (1),
        unavailable                 (2),
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        unwillingToPerform              (3),
        chainingRequired                (4),
        unableToProceed                 (5),
        invalidReference                (6),
        timeLimitExceeded               (7),
        administrativeLimitExceeded     (8),
        loopDetected                    (9),
        unavailableCriticalExtension    (10),
        outOfScope                      (11),
        ditError                        (12),
        invalidQueryReference           (13),
        requestedServiceNotAvailable    (14),
        unsupportedMatchingUse          (15),
        ambiguousKeyAttributes          (16),
        saslBindInProgress              (17) }

updateError  ERROR  ::=  {
        PARAMETER       OPTIONALLY-PROTECTED {
                SET {
                        problem         [0]     UpdateProblem,
                        attributeInfo   [1]     SET SIZE (1..MAX) OF CHOICE {
                                attributeType           AttributeType,
                                attribute               Attribute{{SupportedAttributes}} } OPTIONAL,
                                                        COMPONENTS OF CommonResults } }
        CODE            id-errcode-updateError }

UpdateProblem  ::=  INTEGER {
        namingViolation                 (1),
        objectClassViolation            (2),
        notAllowedOnNonLeaf             (3),
        notAllowedOnRDN                 (4),
        entryAlreadyExists              (5),
        affectsMultipleDSAs             (6),
        objectClassModificationProhibited (7),
        noSuchSuperior                  (8),
        notAncestor                     (9),
        parentNotAncestor               (10),
        hierarchyRuleViolation          (11),
        familyRuleViolation             (12) }

-- attribute types --
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

id-at-family-information OBJECT IDENTIFIER ::= {id-at 64}

END -- *DirectoryAbstractService*

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Annex B

Operational semantics for Basic Access Control

(This annex does not form an integral part of this Recommendation | International Standard)

This annex contains a number of charts that describe the semantics associated with Basic Access Control as it applies to the processing of a Directory operation (see Figures B.1 to B.16).

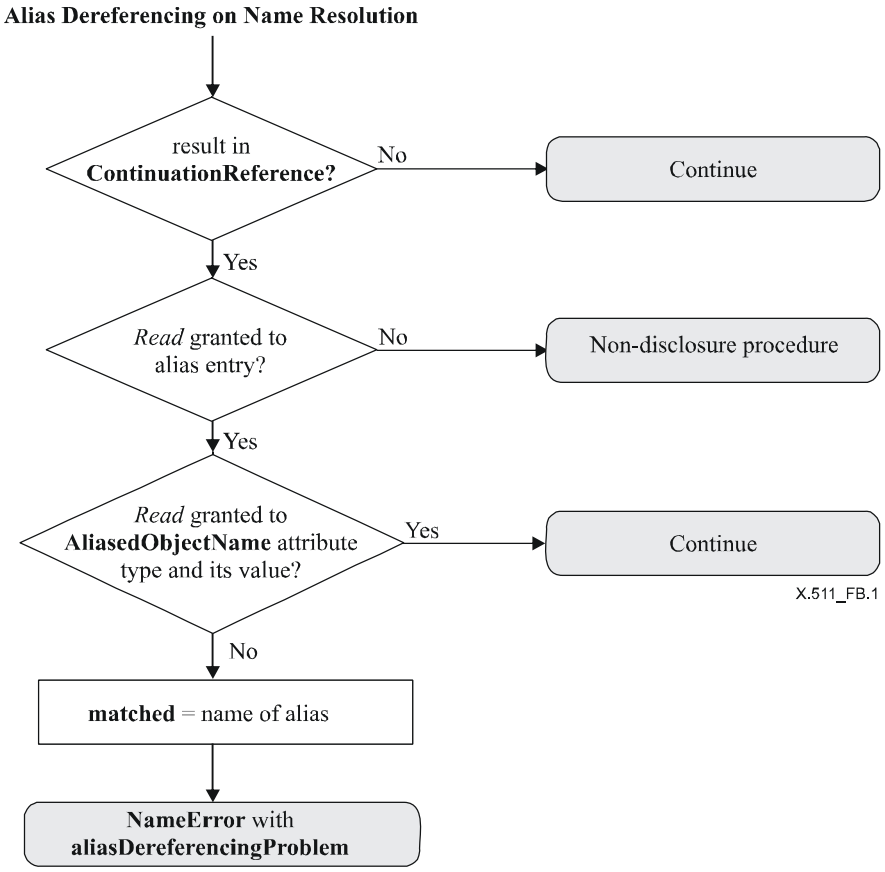

Figure B.1 – Alias Dereferencing in Name Resolution

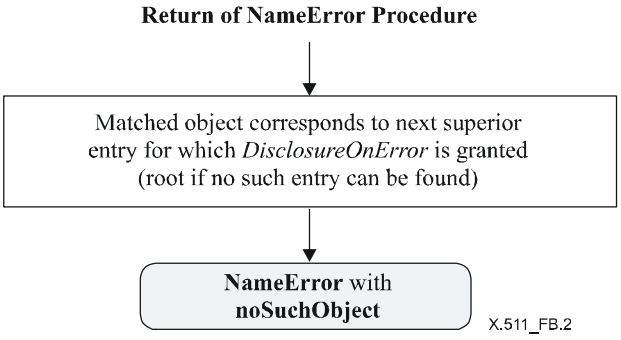

Figure B.2 – Return of Name Error

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

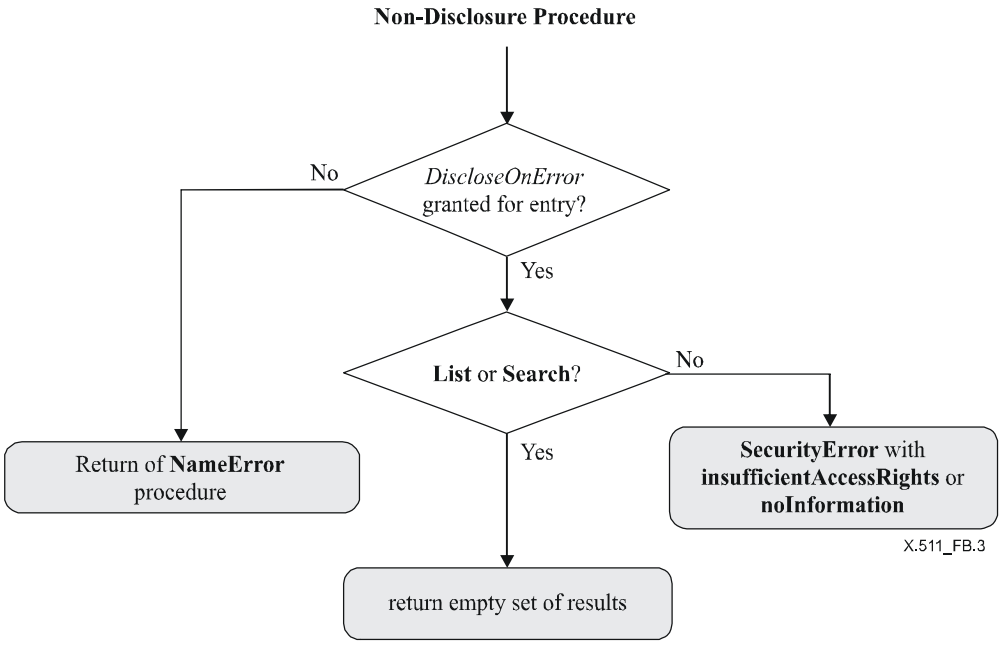

Figure B.3 – Non-Disclosure of the Existence of an Entry

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

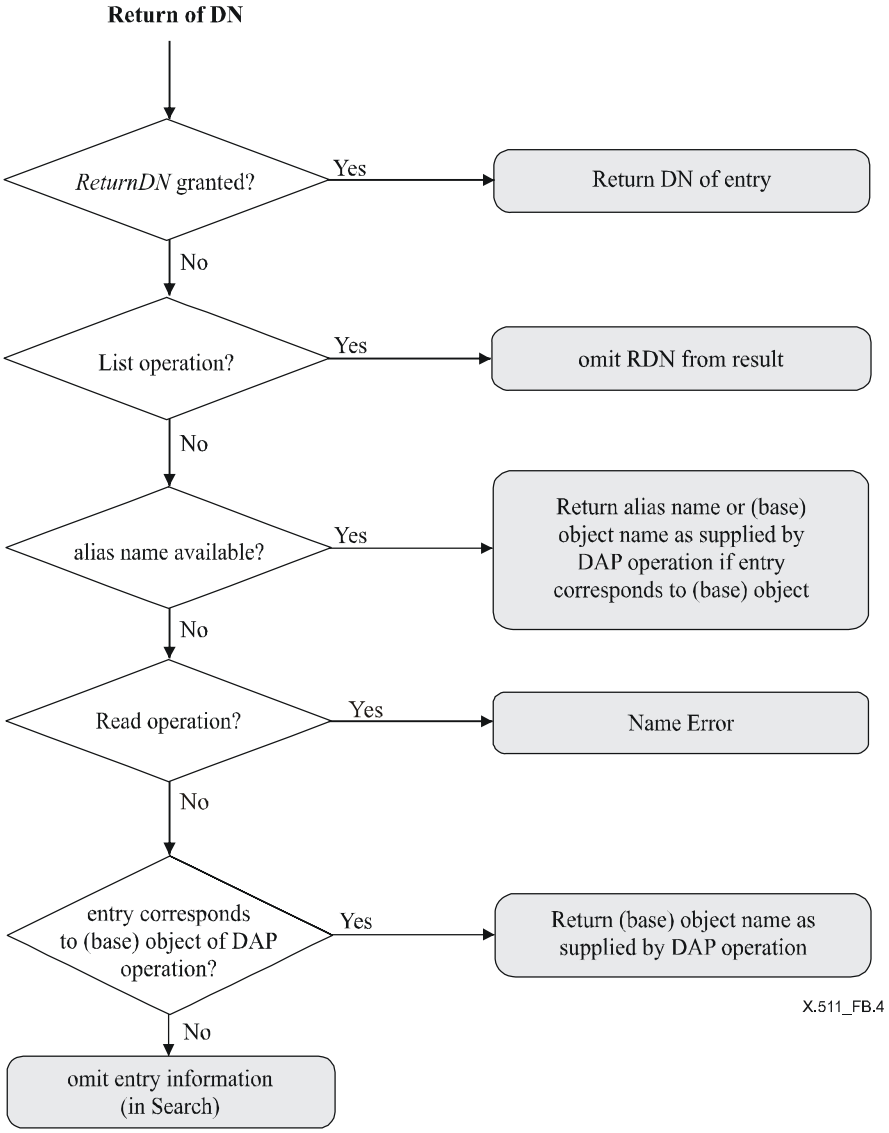

Figure B.4 – Return of Distinguished Name

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

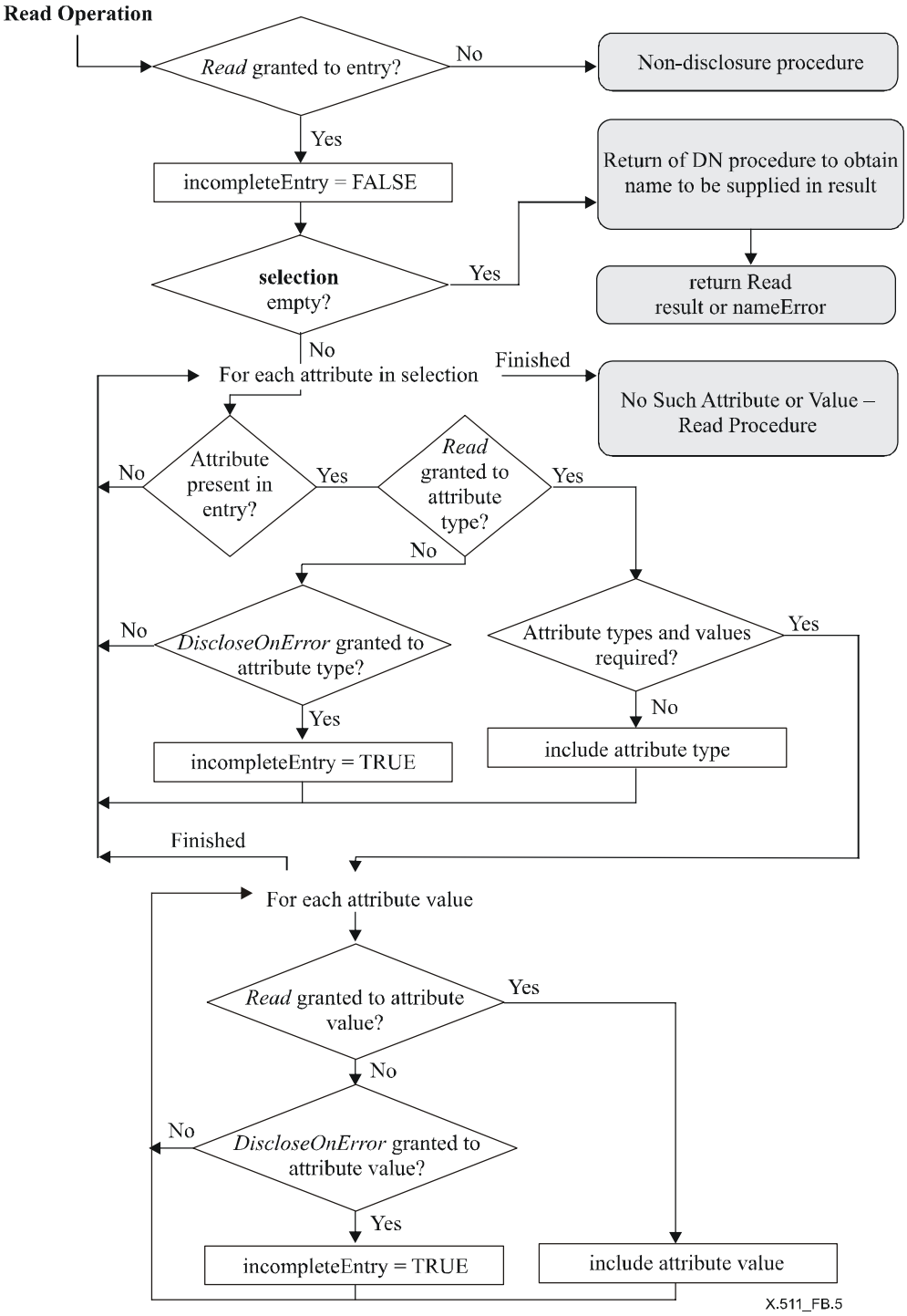

Figure B.5 – Read Operation

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

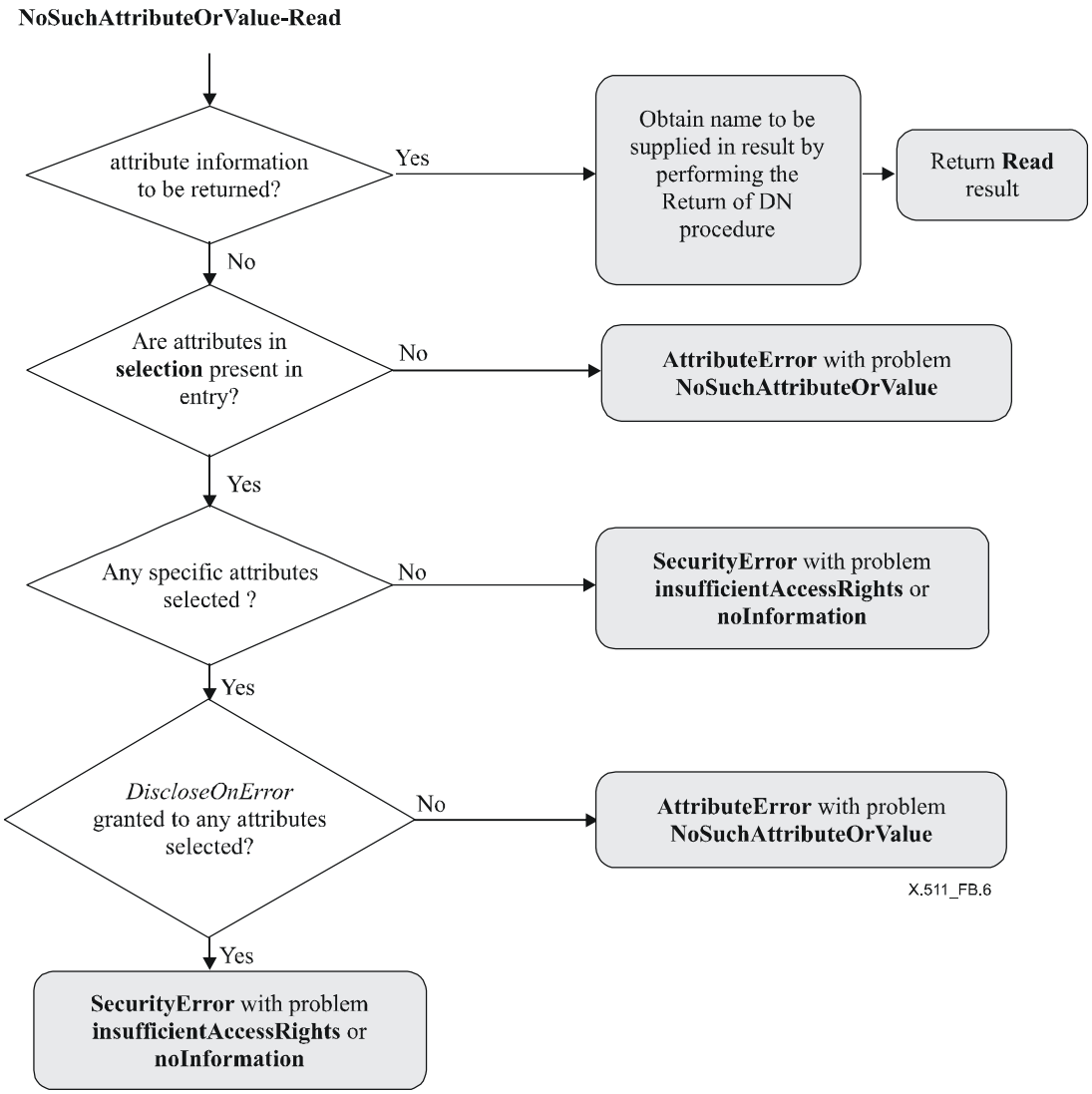

Figure B.6 – No Such Attribute Or Value for Read

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

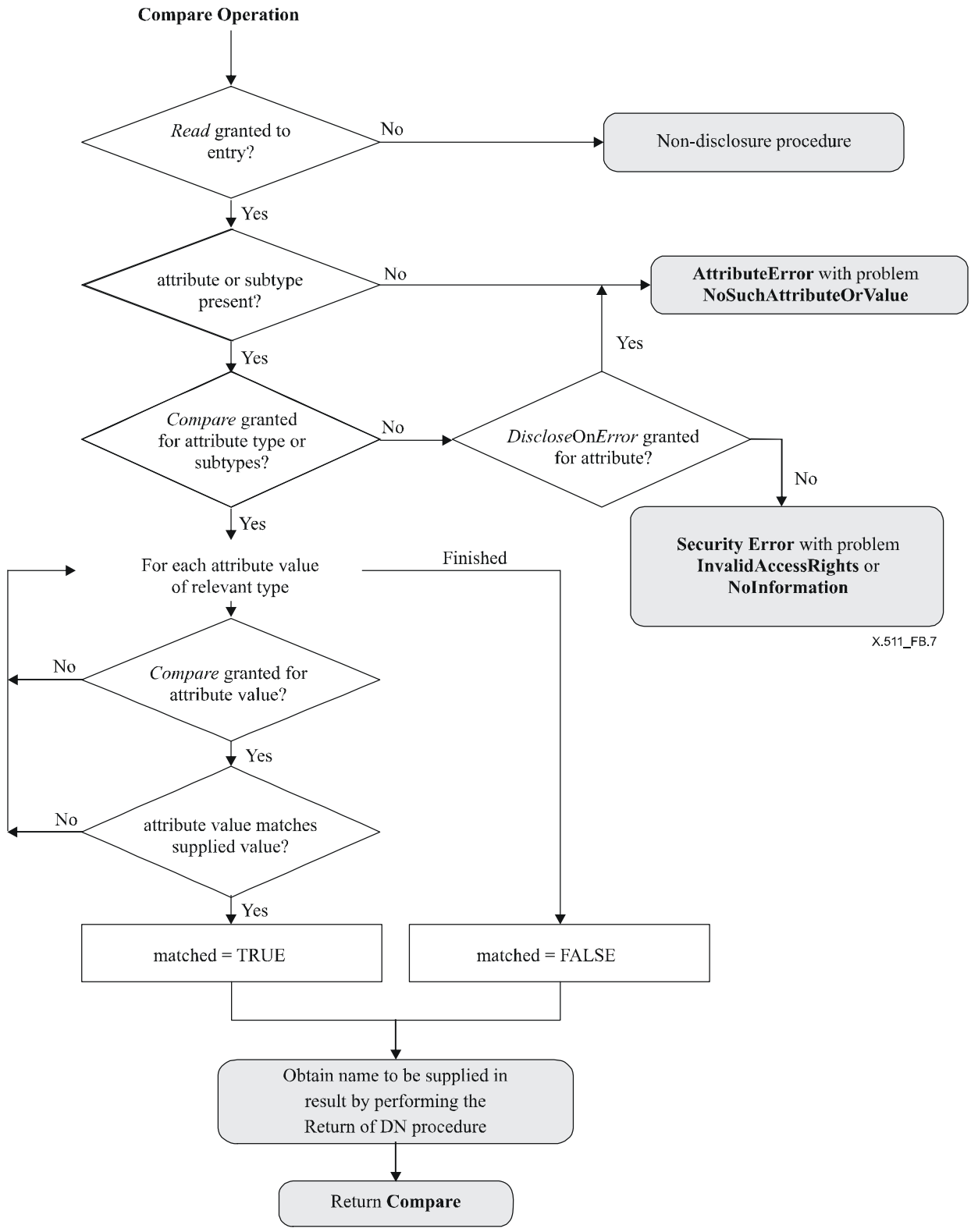

Figure B.7 – Compare Operation

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

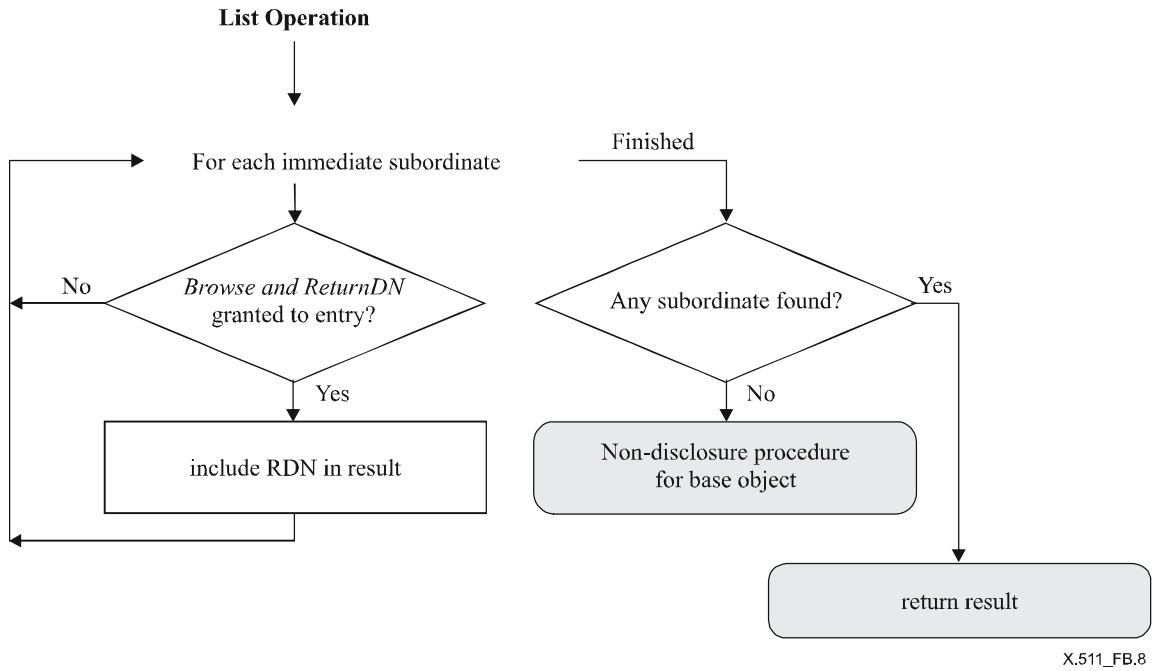

Figure B.8 – List Operation

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

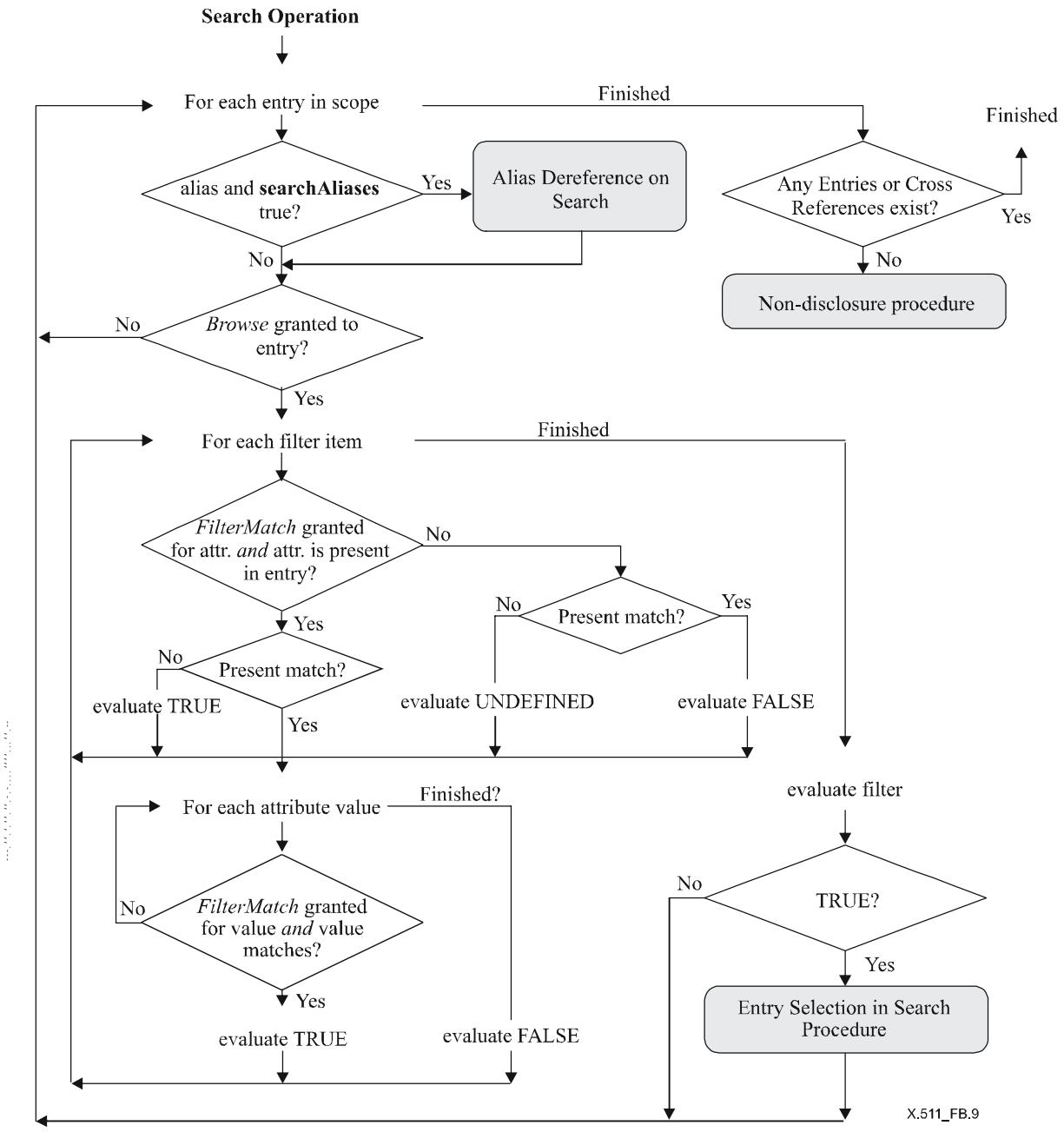

Figure B.9 – Search Operation

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

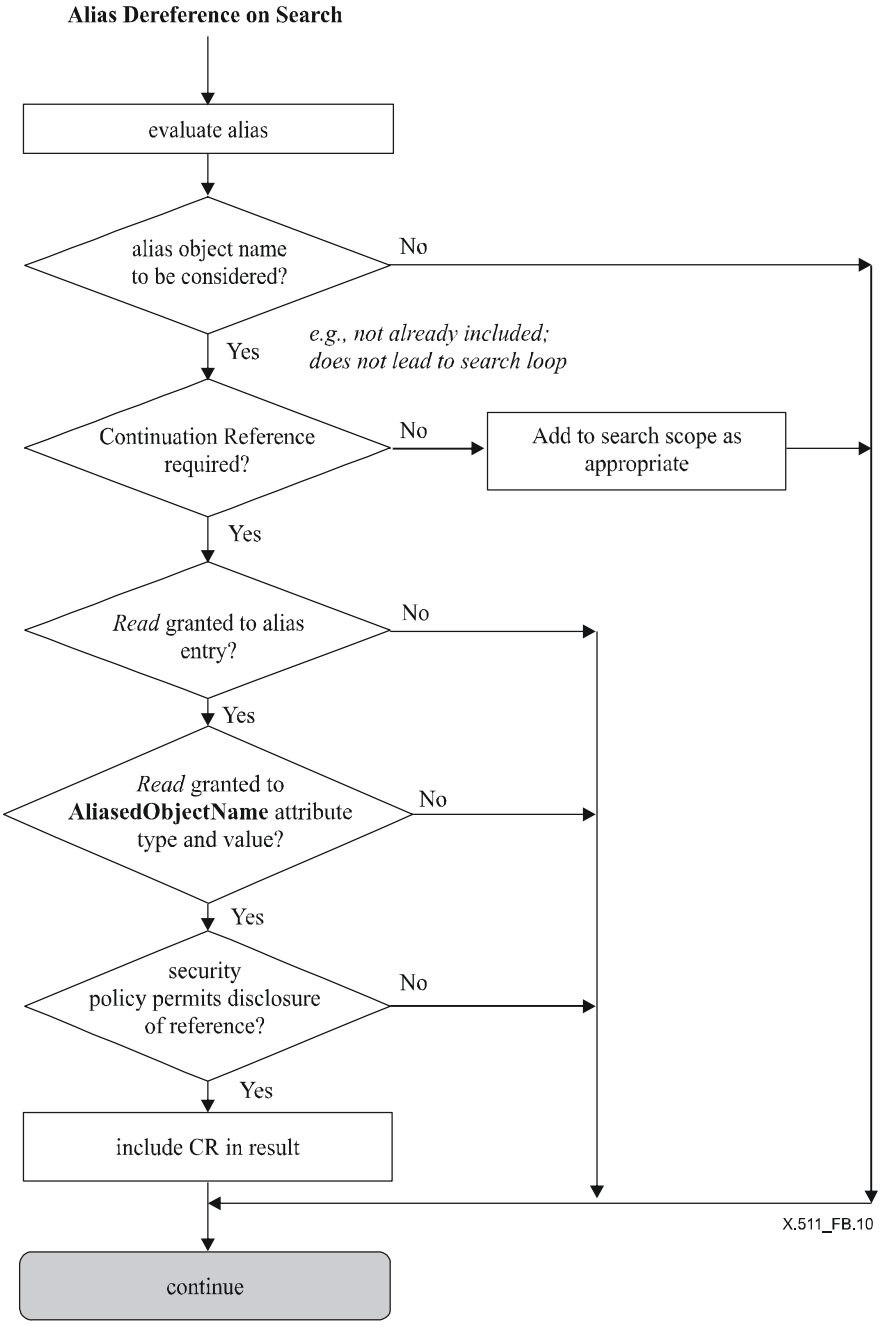

Figure B.10 – Alias Dereference in Search

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

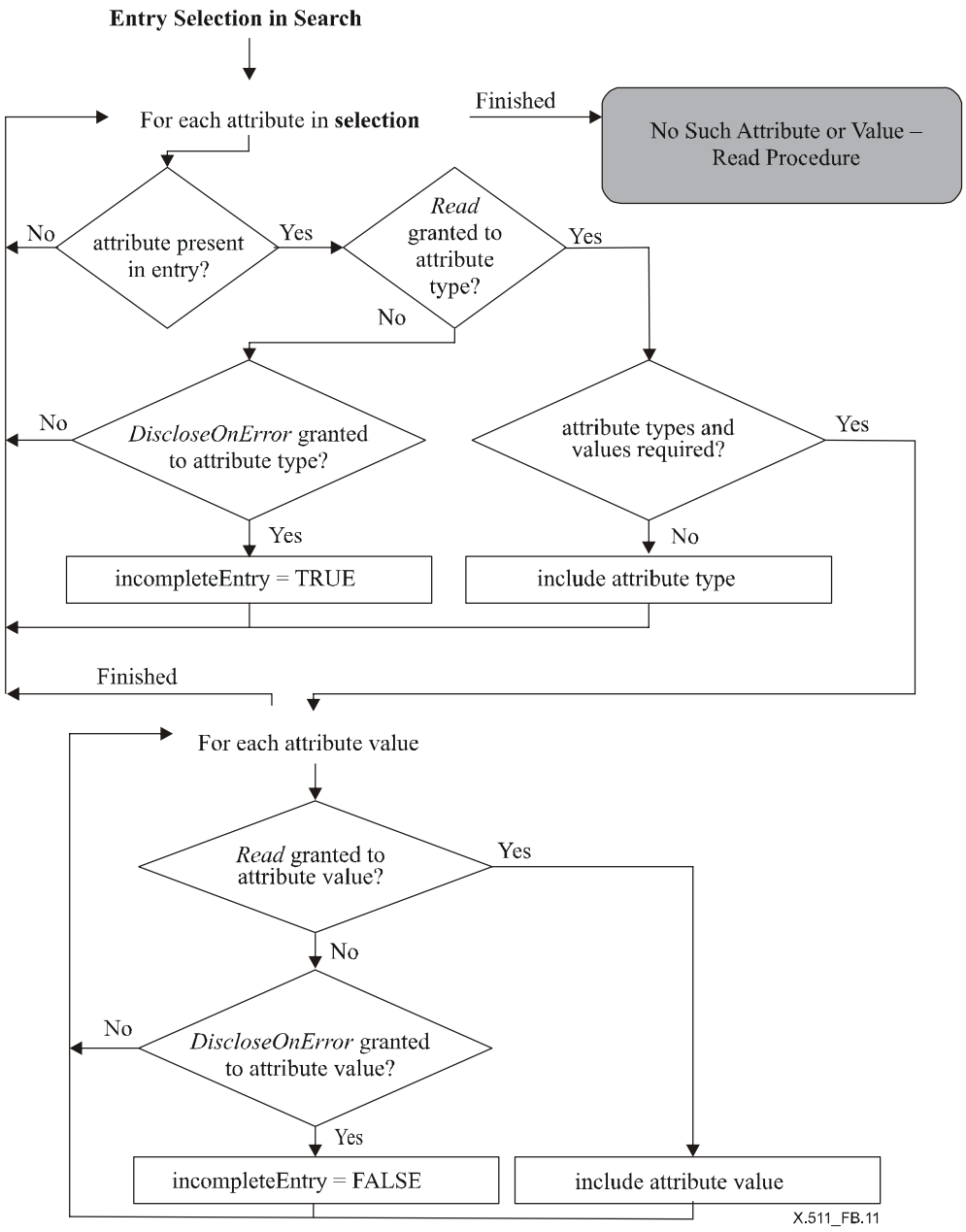

Figure B.11 – Entry Selection in Search

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

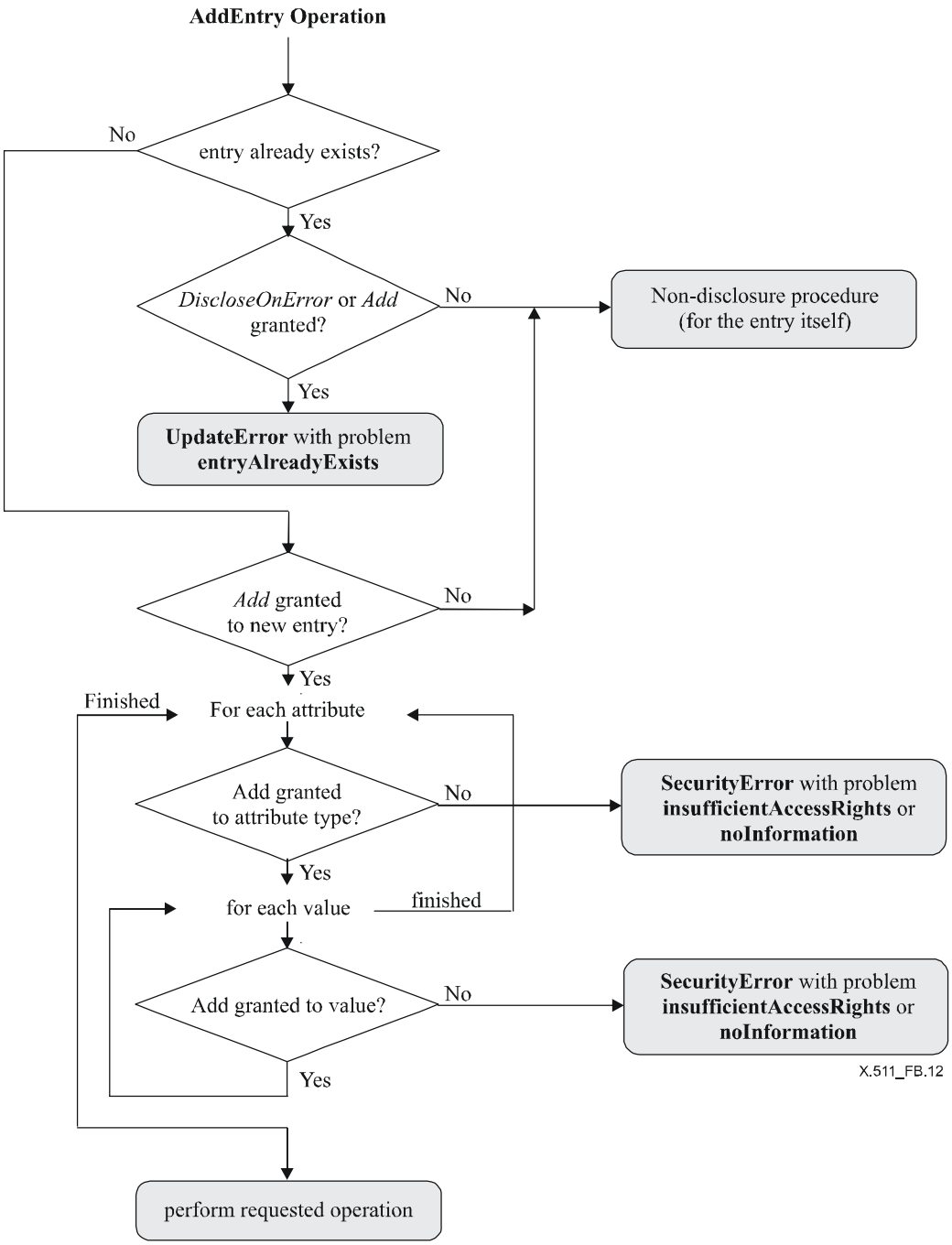

Figure B.12 – Add Entry Operation

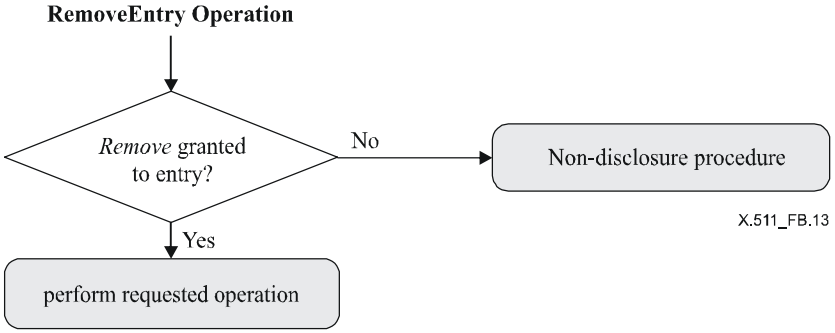

Figure B.13 – Remove Entry Operation

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

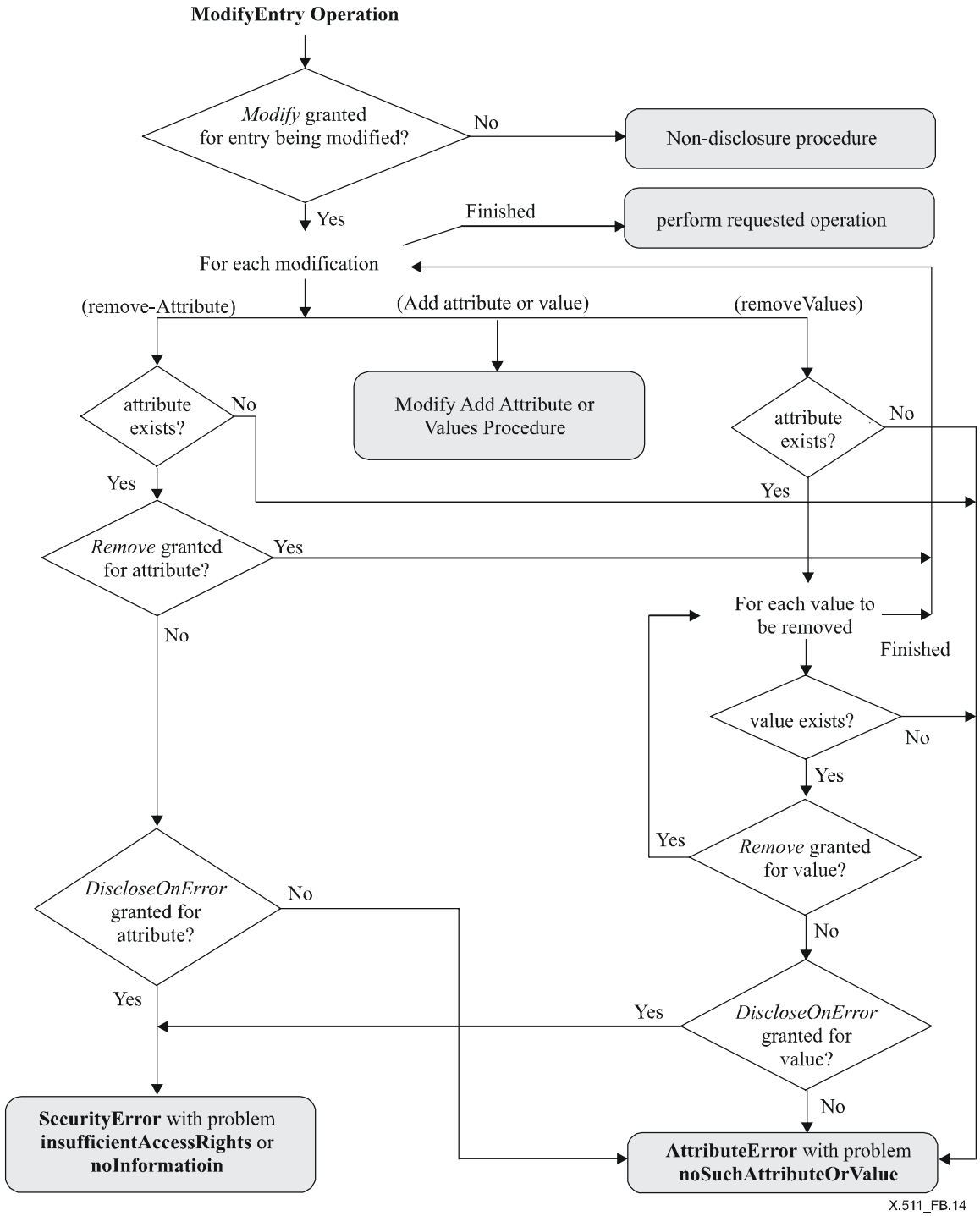

Figure B.14 – Modify Entry Operation

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

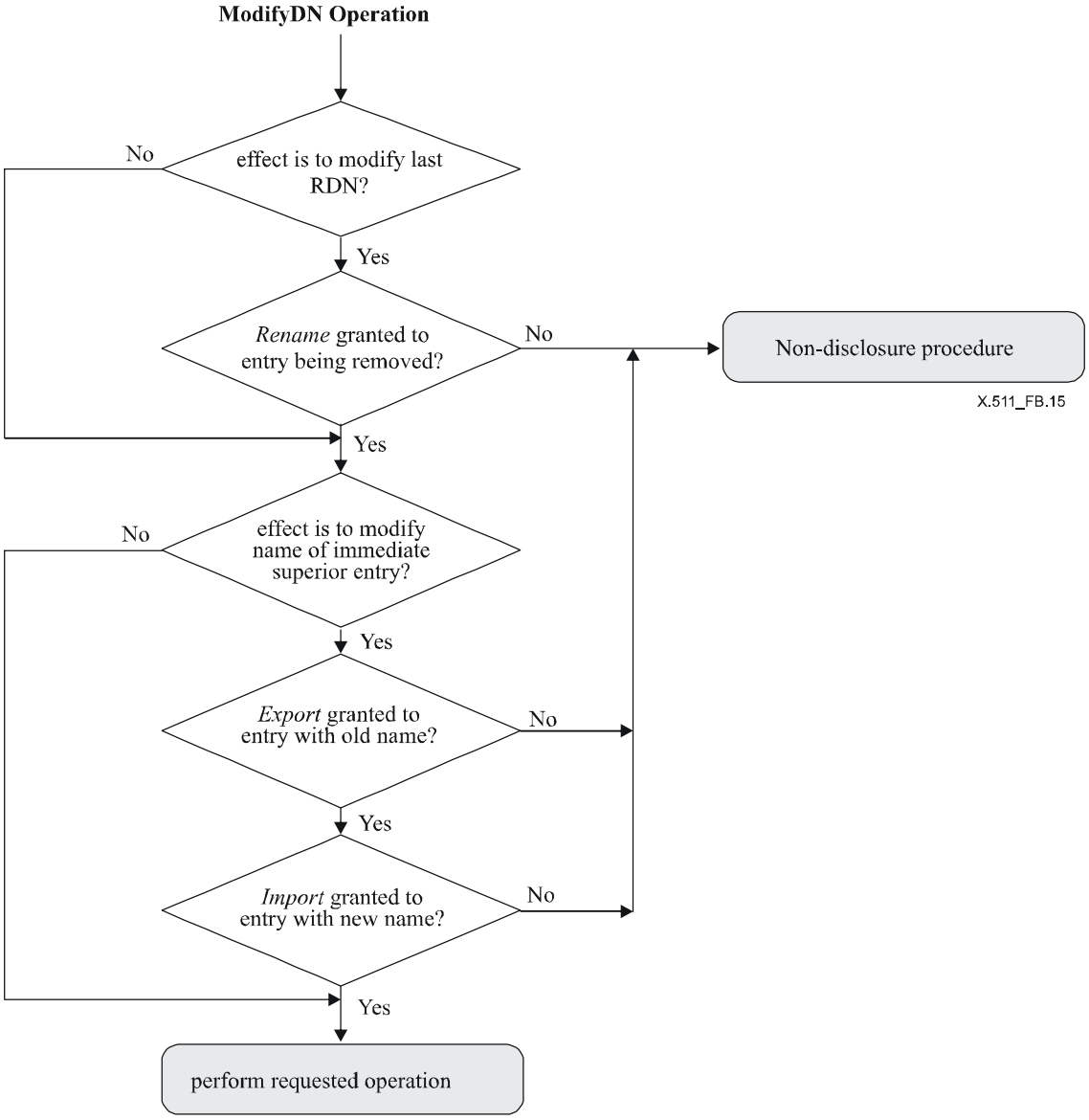

Figure B.15 – ModifyDN Operation

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

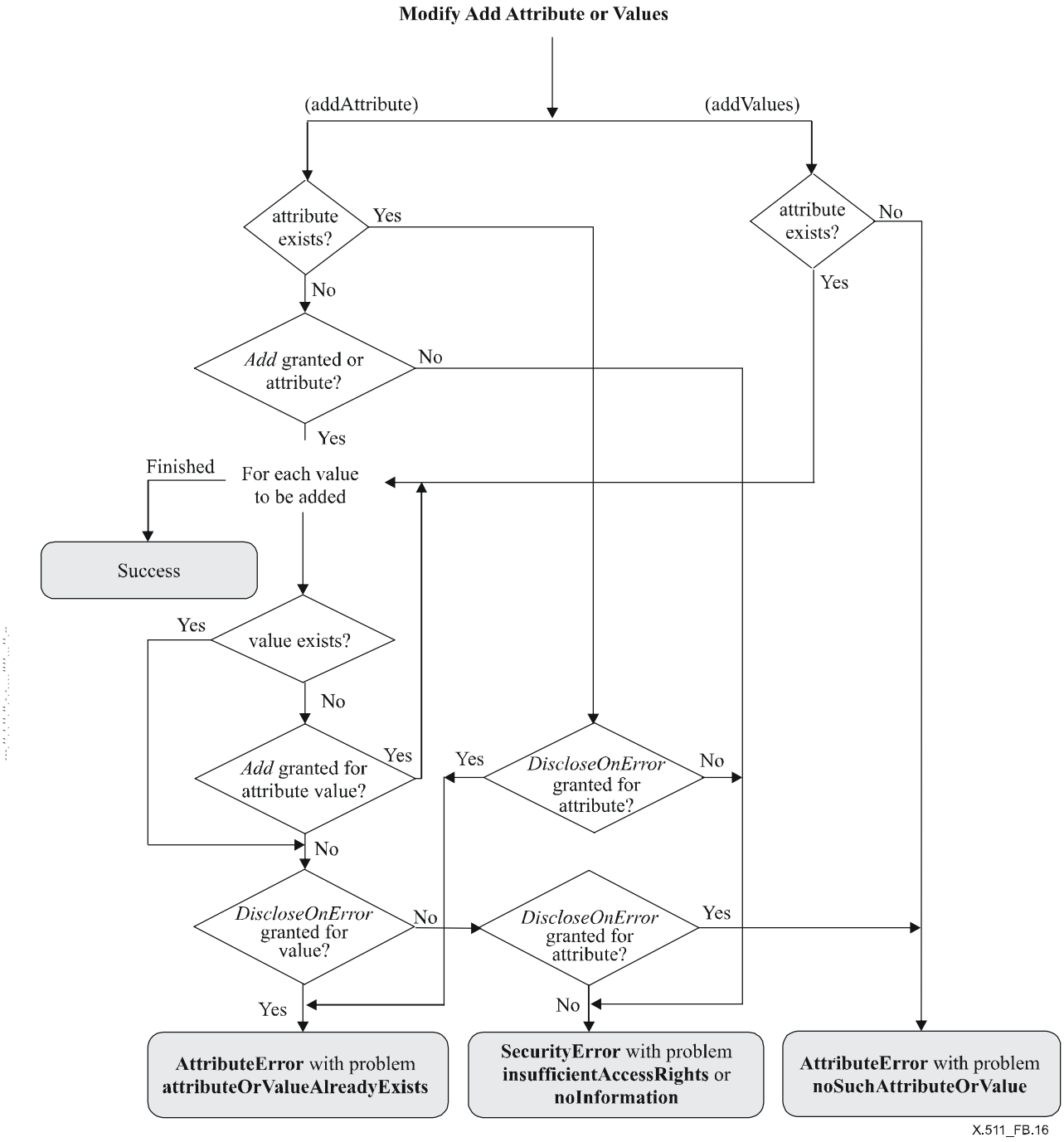

Figure B.16 – Modify Add Attribute or Values

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex C

Examples of searching families of entries

(This annex does not form an integral part of this Recommendation | International Standard)

C.1 Single family example

Suppose Charles Smith has multiple modes of communication: A land telephone, a fax, a mobile phone and e-mail, and each mode has its own associated parameters. Suppose further that Charles Smith has two e-mail accounts, one at his place of work and one at home, and that both offer POP3 mailboxes and SMTP servers. All this information can be held in a compound entry, with Charles Smith's member being the ancestor, with each mode of communication being a child member, and with each e-mail service being a child of the e-mail mode of communication. This is shown in Figure C.1 below. As all the members immediately subordinate to the ancestor has the same structural object class (**comAddr**), the compound entry consists of a single family.

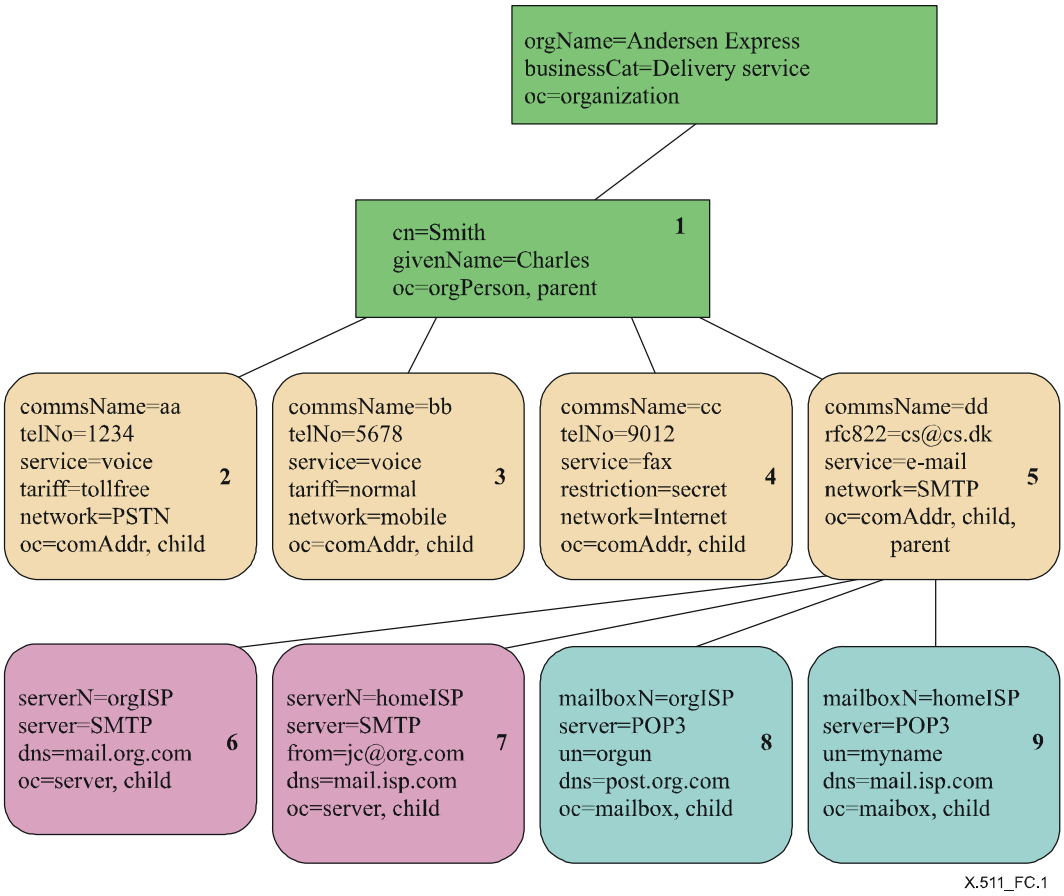

Figure C.1 – Charles Smith's family of entries

Suppose a **search** request is generated with a base object of {...o=Andersen Express}, a filter of {telNo=1234 & tariff=normal}, and a subset of **wholeSubtree** or **oneLevel**. With the **familyGrouping** parameter set to:

a) **entryOnly**: No members of the family would match the filter.

b) **strands** or **multiStrand**: No strand or multi-strand in the family would match the filter.

c) **compoundEntry**: Member 2 and member 3 would together match the filter and would be marked as contributing members. All the members are marked as participating members.

Nothing from this compound entry would be returned for case a) and b) above.

For the case c) above, the returned information would be dependent on the family return specification (e.g., given by **familyReturn** in **EntryInformationSelection**):

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) **contributingEntriesOnly**: The members marked as contributing members, i.e., members 2 and 3 would be returned.

ii) **participatingEntriesOnly** and **compoundEntry**: All the members in the compound entry would be returned.

C.2 Multiple families example

Suppose Charles Smith only has a land telephone and e-mail, but also has two postal addresses with associated parameters. All this information can be held in a compound entry, with Charles Smith's member being the ancestor and with each mode of communication or postal address being a child member. This is shown in Figure C.2 below. As the members immediately subordinate to the ancestor are of two different structural object classes (**comAddr** and **postAddr**), the compound entry consists of two families, where members 1, 2 and 3 constitute one family, and members 1, 4, 5, 6, 7, 8 and 9 constitute another family.

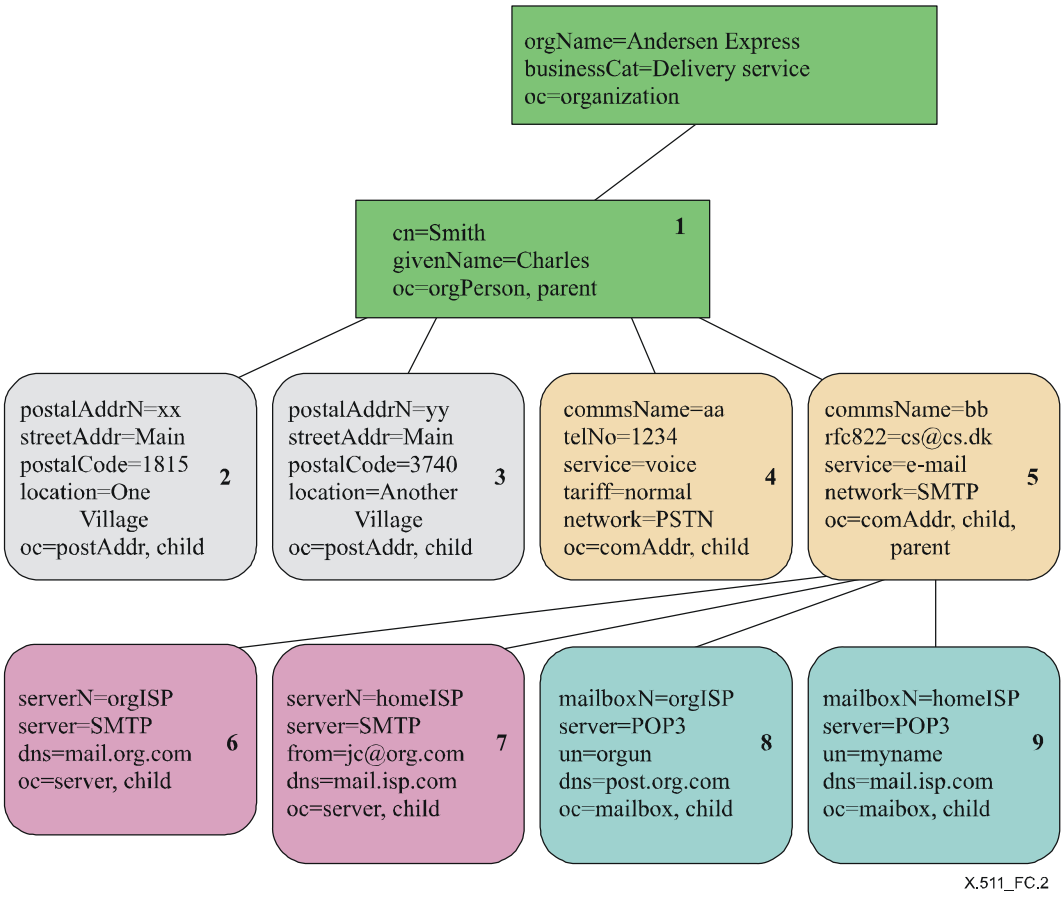

Figure C.2 – Charles Smith's families of entries

C.2.1 Filter example 1

Now suppose a Search request is generated with a base object of {...o=Andersen Express}, a filter of {telNo=1234 & service=e-mail & streetAddr=Main & postalCode=3740 }, and a subset of **wholeSubtree** or **oneLevel**. With the **familyGrouping** parameter set to:

a) **entryOnly**: No single member of the compound entry would match the filter.

b) **strands**: No single strand in any of the families would match the filter.

c) **multiStrand**: No combination of strands, one strand from each family, would match the filter.

d) **compoundEntry**: Members 2, 3, 4 and 5 would together match the filter and would be marked as contributing members. All the members are marked as participating members.

Nothing from this compound entry would be returned for case a), b) and c) above.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

For the case d) above the returned information would be dependent on the family return specification:

i) **contributingEntriesOnly**: The members marked as contributing members, i.e., members 2, 3, 4 and 5 would be returned.

ii) **participatingEntriesOnly** and **compoundEntry**: All the members within the compound entry would be returned.

C.2.2 Filter example 2

If we change the filter to {rfc822=cs@cs.dk & service=e-mail & streetAddr=Main & postalCode=1815}. With the familyGrouping parameter set to:

a) **entryOnly**: No single member of the compound entry would match the filter.

b) **strands**: No single strand in any of the families would match the filter.

c) **multiStrand**: The strand ending in member 2 together with any strand going through member 5 would match the filter. Members 2 and 5 have contributed to the matching and would be marked as contributing members. Members 1, 2, 5, 6, 7, 8 and 9 are marked as participating members.

d) **compoundEntry**: Members 2 and 5 would together match the filter and would be marked as contributing members. All the members are marked as participating members.

Nothing from this compound entry would be returned for case a) and b) above.

For the case c) above, the returned information would be dependent on the family return specification:

i) **contributingEntriesOnly**: The members marked as contributing members, i.e., members 2 and 5, would be returned.

ii) **participatingEntriesOnly**: The members marked as participating members are returned, i.e., the members 1, 2, 5, 6, 7, 8 and 9.

iii) **compoundEntry**: All the members of the compound entry would be returned.

For the case d) above the returned information would be dependent on the family return specification:

i) **contributingEntriesOnly**: The members marked as contributing members, i.e., members 2 and 5, would be returned.

ii) **participatingEntriesOnly** and **compoundEntry**: All the members of the compound entry would be returned.

C.2.3 Filter example 3

If we now change the filter to {rfc822=cs@cs.dk & service=e-mail}. With the **familyGrouping** parameter set to:

a) **entryOnly**: Member 5 alone would match the filter and this member would be marked as a contributing member and as a participating member.

b) **strands**: Any strand going through member 5 would match the filter. Member 5 would be marked as a contributing member. Members 1, 5, 6, 7, 8 and 9 would be marked as participating members.

c) **multiStrand**: Any strand going through member 5 together with any strand of the postal address family would match the filter. Entries 5 would be marked as a contributing member. Members 1, 2, 3, 5, 6, 7, 8 and 9 would be marked as participating members.

d) **compoundEntry**: Member 5 would match the filter and would be marked as a contributing member. All the members are marked as participating members.

For the case a) above the returned information would be dependent on the family return specification:

i) **contributingEntriesOnly** and **participatingEntriesOnly**: Member 5 would be returned.

ii) **compoundEntry**: All the members of the compound entry would be returned.

For the case b) above the returned information would be dependent on the family return specification:

i) **contributingEntriesOnly**: Member 5 would be returned.

ii) **participatingEntriesOnly**: All the members marked as participating members are returned, i.e., the members 1, 5, 6, 7, 8 and 9.

iii) **compoundEntry**: All the members of the compound entry would be returned.

For the case c) above the returned information would be dependent on the family return specification:

i) **contributingEntriesOnly**: Member 5 would be returned.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

ii) **participatingEntriesOnly**: All the members marked as participating members are returned, i.e., the member 1, 2, 3, 5, 6, 7, 8 and 9.

iii) **compoundEntry**: All the members of the compound entry would be returned.

For the case d) above the returned information would be dependent on the family return specification:

i) **contributingEntriesOnly**: Member 5 would be returned.

ii) **participatingEntriesOnly** and **compoundEntry**: All the members of the compound entry would be returned.

C.2.4 Filter example 4

If we finally change the filter to {cn=Smith & givenName=Charles}. The ancestor alone would match the filter.

a) **entryOnly**: Only the ancestor (member 1) would be marked as a contributing member and as a participating member.

b) **strands**, **multiStrand** and **compoundEntry**: The ancestor would be marked as a contributing member and all members would be marked as participating members.

For the case a) above the returned information would be dependent on the family return specification:

i) **contributingEntriesOnly** and **participatingEntriesOnly**: Member 1 would be returned.

ii) **compoundEntry**: All the members of the compound entry would be returned.

For the case b) above the returned information would be dependent on the family return specification:

i) **contributingEntriesOnly**: Member 1 would be returned.

ii) **participatingEntriesOnly** and **compoundEntry**: All the members of the compound entry would be returned.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Annex D

External ASN.1 module

(This annex does not form an integral part of this Recommendation | International Standard)

This annex provides an external ASN.1 module referenced by this Directory Specification. This module is provided for easy compilation of the ASN.1 modules defined by these Directory Specifications.

/ The follow module is an abstract of the module specified by RFC 2025. The import statement that imports has been changed to import only from the current Directory Specifications not to be dependent on modules from previous editions.*/*

```
SpkmGssTokens {iso(1) identified-organization(3) dod(6) internet(1) security(5)
        mechanisms(5) spkm(1) spkmGssTokens(10)}

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

-- EXPORTS ALL

IMPORTS

        Name
                FROM InformationFramework {joint-iso-itu-t(2) ds(5) module(1)
                        informationFramework(1) 6}

        Certificate, CertificateList, CertificatePair, AlgorithmIdentifier{}, SupportedAlgorithms, Validity
                FROM AuthenticationFramework {joint-iso-itu-t(2) ds(5) module(1)
                        authenticationFramework(7) 6};

-- types

SPKM-REQ ::= SEQUENCE {
        requestToken        REQ-TOKEN,
        certif-data         [0]     CertificationData OPTIONAL,
        auth-data           [1]     AuthorizationData OPTIONAL }

CertificationData ::= SEQUENCE {
        certificationPath               [0]     CertificationPath OPTIONAL,
        certificateRevocationList       [1]     CertificateList OPTIONAL
} -- at least one of the above shall be present

CertificationPath ::= SEQUENCE {
        userKeyId               [0]     OCTET STRING OPTIONAL,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        userCertif              [1]     Certificate OPTIONAL,
        verifKeyId              [2]     OCTET STRING OPTIONAL,
        userVerifCertif         [3]     Certificate OPTIONAL,
        theCACertificates       [4]     SEQUENCE OF CertificatePair OPTIONAL
} -- Presence of [2] or [3] implies that [0] or [1] must also be
-- present.  Presence of [4] implies that at least one of [0], [1],
-- [2], and [3] must also be present.

REQ-TOKEN ::= SEQUENCE {
        req-contents    Req-contents,
        algId           AlgorithmIdentifier{{SupportedAlgorithms}},
        req-integrity   Integrity -- "token" is Req-contents
}

Integrity ::= BIT STRING

-- If corresponding algId specifies a signing algorithm,
-- "Integrity" holds the result of applying the signing procedure
-- specified in algId to the BER-encoded octet string which results
-- from applying the hashing procedure (also specified in algId) to
-- the DER-encoded octets of "token".
-- Alternatively, if corresponding algId specifies a MACing
-- algorithm, "Integrity" holds the result of applying the MACing
-- procedure specified in algId to the DER-encoded octets of
-- "token"

Req-contents ::= SEQUENCE {
        tok-id                  INTEGER(256), -- shall contain 0100 (hex)
        context-id              Random-Integer,
        pvno                    BIT STRING,
        timestamp               UTCTime OPTIONAL, -- mandatory for SPKM-2
        randSrc                 Random-Integer,
        targ-name               Name,
        src-name        [0]     Name OPTIONAL,
        req-data                Context-Data,
        validity        [1]     Validity OPTIONAL,
        key-estb-set            Key-Estb-Algs,
        key-estb-req            BIT STRING OPTIONAL,
        key-src-bind            OCTET STRING OPTIONAL
  -- This field must be present for the case of SPKM-2
  -- unilateral authen. if the K-ALG in use does not provide
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
  -- such a binding (but is optional for all other cases).
  -- The octet string holds the result of applying the
  -- mandatory hashing procedure (in MANDATORY I-ALG;
  -- see Section 2.1) as follows:  MD5(src || context_key),
  -- where "src" is the DER-encoded octets of src-name,
  -- "context-key" is the symmetric key (i.e., the
  -- unprotected version of what is transmitted in
  -- key-estb-req), and "||" is the concatenation operation.
}

Random-Integer ::= BIT STRING

Context-Data ::= SEQUENCE {
        channelId               ChannelId OPTIONAL,
        seq-number      INTEGER OPTIONAL,
        options         Options,
        conf-alg                Conf-Algs,
        intg-alg                Intg-Algs,
        owf-alg                 OWF-Algs
}

ChannelId ::= OCTET STRING

Options ::= BIT STRING {
        delegation-state(0), mutual-state(1), replay-det-state(2), sequence-state(3),
        conf-avail(4), integ-avail(5), target-certif-data-required(6)}

Conf-Algs ::= CHOICE {
        algs    [0]     SEQUENCE OF AlgorithmIdentifier{{SupportedAlgorithms}},
        null    [1]     NULL }

Intg-Algs ::= SEQUENCE OF AlgorithmIdentifier{{SupportedAlgorithms}}

OWF-Algs ::= SEQUENCE OF AlgorithmIdentifier{{SupportedAlgorithms}}

Key-Estb-Algs ::= SEQUENCE OF AlgorithmIdentifier{{SupportedAlgorithms}}

SPKM-REP-TI ::= SEQUENCE {
        responseToken  REP-TI-TOKEN,
        certif-data    CertificationData OPTIONAL
        -- present if target-certif-data-required option was
}       -- set to TRUE in SPKM-REQ
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
REP-TI-TOKEN ::= SEQUENCE {
        rep-ti-contentsRep-ti-contents,
        algId                   AlgorithmIdentifier{{SupportedAlgorithms}},
        rep-ti-integ    Integrity -- "token" is Rep-ti-contents
}

Rep-ti-contents ::= SEQUENCE {
        tok-id                  INTEGER(512), -- shall contain 0200 (hex)
        context-id              Random-Integer,
        pvno            [0]     BIT STRING OPTIONAL,
        timestamp               UTCTime OPTIONAL, -- mandatory for SPKM-2
        randTarg                Random-Integer,
        src-name        [1]     Name OPTIONAL,
        targ-name               Name,
        randSrc                 Random-Integer,
        rep-data                Context-Data,
        validity        [2]     Validity OPTIONAL,
        key-estb-id             AlgorithmIdentifier{{SupportedAlgorithms}} OPTIONAL,
        key-estb-str            BIT STRING OPTIONAL
}

SPKM-REP-IT ::= SEQUENCE {
        responseToken           REP-IT-TOKEN,
        algId                   AlgorithmIdentifier{{SupportedAlgorithms}},
        rep-it-integ    Integrity -- "token" is REP-IT-TOKEN
}

REP-IT-TOKEN ::= SEQUENCE {
        tok-id                  INTEGER(768), -- shall contain 0300 (hex)
        context-id              Random-Integer,
        randSrc                 Random-Integer,
        randTarg                Random-Integer,
        targ-name               Name,
        src-name                Name OPTIONAL,
        key-estb-rep    BIT STRING OPTIONAL
}

SPKM-ERROR ::= SEQUENCE {
        errorToken      ERROR-TOKEN,
        algId                   AlgorithmIdentifier{{SupportedAlgorithms}},
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        integrity               Integrity -- "token" is ERROR-TOKEN
}

ERROR-TOKEN ::= SEQUENCE {
        tok-id          INTEGER(1024), -- shall contain 0400 (hex)
        context-id      Random-Integer }

SPKM-MIC ::= SEQUENCE {
        mic-header      Mic-Header,
        int-cksum               BIT STRING }

Mic-Header ::= SEQUENCE {
        tok-id                  INTEGER(257), -- shall contain 0101 (hex)
        context-id              Random-Integer,
        int-alg [0]     AlgorithmIdentifier{{SupportedAlgorithms}} OPTIONAL,
        snd-seq         [1]     SeqNum OPTIONAL }

SeqNum ::= SEQUENCE {
        num             INTEGER,
        dir-ind BOOLEAN }

SPKM-WRAP ::= SEQUENCE {
        wrap-header     Wrap-Header,
        wrap-body               Wrap-Body }

Wrap-Header ::= SEQUENCE {
        tok-id                  INTEGER(513), -- shall contain 0201 (hex)
        context-id              Random-Integer,
        int-alg [0]     AlgorithmIdentifier{{SupportedAlgorithms}} OPTIONAL,
        conf-alg        [1]     Conf-Alg OPTIONAL,
        snd-seq         [2]     SeqNum OPTIONAL }

Wrap-Body ::= SEQUENCE {
        int-cksum       BIT STRING,
        data            BIT STRING }

Conf-Alg ::= CHOICE {
        algId   [0]     AlgorithmIdentifier{{SupportedAlgorithms}},
        null    [1]     NULL }

SPKM-DEL ::= SEQUENCE {
        del-header      Del-Header,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        int-cksum       BIT STRING }

Del-Header ::= SEQUENCE {
        tok-id                  INTEGER(769), -- shall contain 0301 (hex)
        context-id              Random-Integer,
        int-alg [0]     AlgorithmIdentifier{{SupportedAlgorithms}} OPTIONAL,
        snd-seq         [1]     SeqNum OPTIONAL }

-- other types
-- from [RFC-1508]

MechType ::= OBJECT IDENTIFIER

InitialContextToken ::= [APPLICATION 0] IMPLICIT SEQUENCE {
        thisMech                MechType,
        innerContextToken       SPKMInnerContextToken
} -- when thisMech is SPKM-1 or SPKM-2

SPKMInnerContextToken ::= CHOICE {
        req             [0]     SPKM-REQ,
        rep-ti  [1]     SPKM-REP-TI,
        rep-it  [2]     SPKM-REP-IT,
        error           [3]     SPKM-ERROR,
        mic             [4]     SPKM-MIC,
        wrap            [5]     SPKM-WRAP,
        del             [6]     SPKM-DEL }

-- from [RFC-1510]

AuthorizationData ::=   SEQUENCE OF SEQUENCE {
        ad-type         INTEGER,
        ad-data         OCTET STRING }

-- object identifier assignments

md5-DES-CBC OBJECT IDENTIFIER ::=
        { iso(1) identified-organization(3) dod(6) internet(1) security(5)
        integrity(3) md5-DES-CBC(1) }

sum64-DES-CBC OBJECT IDENTIFIER ::=
        { iso(1) identified-organization(3) dod(6) internet(1) security(5)
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        integrity(3) sum64-DES-CBC(2) }

spkm-1 OBJECT IDENTIFIER ::=
        {  iso(1) identified-organization(3) dod(6) internet(1) security(5)
         mechanisms(5) spkm(1) spkm-1(1) }

spkm-2 OBJECT IDENTIFIER ::=
        { iso(1) identified-organization(3) dod(6) internet(1) security(5)
        mechanisms(5) spkm(1) spkm-2(2) }

END  -- SpkmGssTokens
```


--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Annex E

Amendments and corrigenda

(This annex does not form an integral part of this Recommendation | International Standard)

This edition of this Directory Specification includes the following amendment to the previous edition that was balloted and approved by ISO/IEC:

– Amendment 3 on Communications support enhancements.

This edition of this Directory Specification includes the following technical corrigenda correcting the defects documented in Defect Reports against the 5th edition of this Directory Specification:

– Technical Corrigendum 1 (covering Defect Reports 321, 323 and 324); and

– Technical Corrigendum 2 (covering Defect Reports 326 and 329).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

ICS 35.100.70

Price based on 119 pages

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale