INTERNATIONAL STANDARD

ISO/IEC 9594-3

Ninth edition
2020-11

Information technology — Open systems interconnection —

Part 3: The Directory: Abstract service definition

Reference number
ISO/IEC 9594-3:2020(E)

Documents
Document Sharing Hub
© ISO/IEC 2020

Full document details: ISO_IEC 9594-3_2020

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2020

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub

Full document details: ISO_IEC 9594-3_2020

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted (see www.iso.org/directives).

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see http://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by ITU-T as ITU-T X.511 (10/2019) and drafted in accordance with its editorial rules, in collaboration with Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 6, *Telecommunications and information exchange between systems*.

This ninth edition cancels and replaces the eighth edition (ISO/IEC 9594-3:2017), which has been technically revised.

A list of all parts in the ISO/IEC 9594 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

Documents
Document Sharing Hub

Full document details: ISO_IEC 9594-3_2020

CONTENTS

		Page
1	Scope	1
2	Normative references	1
2.1	Identical Recommendations | International Standards	1
2.2	Paired Recommendations | International Standards equivalent in technical content	1
2.3	Additional references	2
3	Definitions	2
3.1	OSI Reference Model security architecture definitions	2
3.2	Basic Directory definitions	2
3.3	Directory model definitions	2
3.4	Directory information base definitions	2
3.5	Directory entry definitions	2
3.6	Name definitions	3
3.7	Distributed operations definitions	3
3.8	Abstract service definitions	3
4	Abbreviations	4
5	Conventions	4
6	Overview of the Directory service	5
7	Information types and common procedures	5
7.1	Introduction	5
7.2	Information types defined elsewhere	5
7.3	Common arguments	6
7.4	Common results	9
7.5	Service controls	10
7.6	Entry information selection	12
7.7	Entry information	15
7.8	Filter	17
7.9	Paged results	20
7.10	Security parameters	22
7.11	Common elements of procedure for access control	23
7.12	Managing the DSA Information Tree	25
7.13	Procedures for families of entries	25
8	Directory authentication	26
8.1	Simple authentication procedure	26
8.2	Password policy	28
9	Bind, Unbind operations, Change Password and Administer Password operations	31
9.1	Directory Bind	31
9.2	Directory Unbind	34
10	Directory Read operations	34
10.1	Read	34
10.2	Compare	37
10.3	Abandon	40
11	Directory Search operations	40
11.1	List	40
11.2	Search	44
12	Directory Modify operations	55
12.1	Add Entry	55
12.2	Remove Entry	57
12.3	Modify Entry	59
12.4	Modify DN	63
12.5	Change Password	65
12.6	Administer Password	66

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

Page

13 Operations for LDAP messages 66
13.1 LDAP Transport operation 67
13.2 Linked LDAP operation 69
14 Errors 69
14.1 Error precedence 69
14.2 Abandoned 70
14.3 Abandon Failed 70
14.4 Attribute Error 71
14.5 Name Error 72
14.6 Referral 73
14.7 Security Error 73
14.8 Service Error 74
14.9 Update Error 76
15 Analysis of search arguments 77
15.1 General check of search filter 78
15.2 Check of request-attribute-profiles 79
15.3 Check of controls and hierarchy selections 80
15.4 Check of matching use 81
Annex A – Abstract Service in ASN.1 82
Annex B – Operational semantics for Basic Access Control 98
Annex C – Examples of searching families of entries 111
C.1 Single family example 111
C.2 Multiple families example 112
Annex D – External ASN.1 module 115
Annex E – Use of protected passwords for Bind operations 119
Annex F – Amendments and corrigenda 120

© ISO/IEC 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

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 ninth edition technically revises and enhances the eighth edition of this Recommendation | International Standard.

This ninth edition specifies versions 1 and 2 of the Directory protocols.

Rec. ITU-T X.511 (1993) | ISO/IEC 9594-3 (1995), Rec. ITU-T X.518 (1993) | ISO/IEC 9594-4 (1995) and Rec. ITU-T X.519 (1993) | ISO/IEC 9594-5 (1995) and their previous edition 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 nine editions, except for those specifically assigned to version 2, are accommodated using the rules of extensibility defined in Rec. ITU-T 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.

Annex D, 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, provides a suggested technique for Bind protected password.

Annex F, 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 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

Documents
Document Sharing Hub

Full document details: ISO_IEC 9594-3_2020

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

- Recommendation ITU-T X.500 (2019) | ISO/IEC 9594-1:2020, *Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services*.
- Recommendation ITU-T X.501 (2019) | ISO/IEC 9594-2:2020, *Information technology – Open Systems Interconnection – The Directory: Models*.
- Recommendation ITU-T X.509 (2019) | ISO/IEC 9594-8:2020, *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks*.
- Recommendation ITU-T X.518 (2019) | ISO/IEC 9594-4:2020, *Information technology – Open Systems Interconnection – The Directory: Procedures for distributed operation*.
- Recommendation ITU-T X.519 (2019) | ISO/IEC 9594-5:2020, *Information technology – Open Systems Interconnection – The Directory: Protocol specifications*.
- Recommendation ITU-T X.520 (2019) | ISO/IEC 9594-6:2020, *Information technology – Open Systems Interconnection – The Directory: Selected attribute types*.
- Recommendation ITU-T X.521 (2019) | ISO/IEC 9594-7:2020, *Information technology – Open Systems Interconnection – The Directory: Selected object classes*.
- Recommendation ITU-T X.525 (2019) | ISO/IEC 9594-9:2020, *Information technology – Open Systems Interconnection – The Directory: Replication*.
- Recommendation ITU-T X.680 (2015) | ISO/IEC 8824-1:2015, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*.
- Recommendation ITU-T X.681 (2015) | ISO/IEC 8824-2:2015, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification*.
- Recommendation ITU-T X.682 (2015) | ISO/IEC 8824-3:2015, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification*.
- Recommendation ITU-T X.683 (2015) | ISO/IEC 8824-4:2015, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications*.

2.2 Paired Recommendations | International Standards equivalent in technical content

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

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

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

2.3 Additional references

- Recommendation ITU-T X.200 (1994) | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The basic model*.
- IETF RFC 2025 (1996), *The Simple Public-Key GSS-API Mechanism (SPKM).*
- IETF RFC 4422 (2006), *Simple Authentication and Security Layer (SASL).*
- IETF RFC 4511 (2006), *Lightweight Directory Access Protocol (LDAP): The Protocol*.

3 Definitions

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

3.1 OSI Reference Model security architecture definitions

The following terms are defined in Rec. ITU-T X.800 | ISO 7498-2:

a) password.

3.2 Basic Directory definitions

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

a) *Directory*;
b) *Directory Information Base*;
c) *(Directory) User*.

3.3 Directory model definitions

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

a) *Directory System Agent*;
b) *Directory User Agent*.

3.4 Directory information base definitions

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

a) *alias entry*;
b) *ancestor*;
c) *compound entry*;
d) *(Directory) entry*;
e) *Directory Information Tree*;
f) *family (of entries)*;
g) *immediate superior*;
h) *immediately superior entry/object*;
i) *object*;
j) *object class*;
k) *object entry*;
l) *subordinate*;
m) *superior*.

3.5 Directory entry definitions

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

a) *attribute*;
b) *attribute type*;
c) *attribute value*;

© ISO/IEC 2020 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

d) *attribute value assertion*;

e) *context*;

f) *context type*;

g) *context value*;

h) *operational attribute*;

i) *matching rule*;

j) *user attribute*.

3.6 Name definitions

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

a) *alias, alias name*;

b) *distinguished name*;

c) *(directory) name*;

d) *purported name*;

e) *relative distinguished name*.

3.7 Distributed operations definitions

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

a) *bound DSA*;

b) *chaining*;

c) *initial performer*;

d) *LDAP requester*;

e) *referral*.

3.8 Abstract service definitions

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

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

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

3.8.3 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.

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

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

3.8.6 originator: The user that originated an operation.

3.8.7 participating 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.

3.8.8 Password expiration: The situation where a user password has reached the end of its validity period: the account is locked and the user has to change the password before doing any other directory operation.

3.8.9 Password quality attributes: Attributes that specify how a password shall be constructed. Password quality attributes include things like minimum length, mixture of characters (uppercase, lowercase, figures, punctuations, etc), and avoidance of trivial passwords.

3.8.10 Password history: List of old passwords and the times they were inserted in the history.

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

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

3.8.12 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.

3.8.13 reply: A DAP/DSP result or an error; or an LDAP result.

3.8.14 request: Information consisting of an operation code and associated components to convey a directory operation from a requester to a performer.

3.8.15 requester: A DUA, an LDAP client or a DSA sending a request to perform (i.e., invoke) an operation.

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

3.8.17 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).

4 Abbreviations

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

ACI	Access Control Information
AVA	Attribute Value Assertion
DIB	Directory Information Base
DIT	Directory Information Tree
DMD	Directory Management Domain
DSA	Directory System Agent
DUA	Directory User Agent
LDAP	Lightweight Directory Access Protocol
RDN	Relative Distinguished Name

5 Conventions

The term "Directory Specification" (as in "this Directory Specification") shall be taken to mean Rec. ITU-T X.511 | ISO/IEC 9594-3. The term "Directory Specifications" shall be taken to mean the Rec. ITU-T X.500 | ISO/IEC 9594-1, Rec. ITU-T X.501 | ISO/IEC 9594-2, Rec. ITU-T X.511 | ISO/IEC 9594-3, Rec. ITU-T X.518 | ISO/IEC 9594-4, Rec. ITU-T X.519 | ISO/IEC 9594-5, Rec. ITU-T X.520 | ISO/IEC 9594-6, Rec ITU-T X.521 | ISO/IEC 9594-7 and Rec. ITU-T X.525 | ISO/IEC 9594-9.

If an International Standard or ITU-T Recommendation is referenced within normal text without an indication of the edition, the edition shall be taken to be the latest one as specified in the normative references clause.

Prior to year 2020, the parts making up the Directory Specifications progressed together and can therefore collectively be identified as the Directory Specifications of a specific edition using the format: Rec. ITU-T X.5** (yyyy) | ISO/IEC 9594-*:yyyy (e.g.; Rec ITU-T X.5** (1993) | ISO/IEC 9594-*:1995).

This Directory Specification makes extensive use of Abstract Syntax Notation One (ASN.1) for the formal specification of data types and values, as it is specified in Rec. ITU-T X.680 | ISO/IEC 8824-1, ITU-T X.681 (2015) | ISO/IEC 8824-2, ITU-T X.682 (2015) | ISO/IEC 8824-3, ITU-T X.683 (2015) | ISO/IEC 8824-4 and Rec. ITU-T X.690 | ISO/IEC 8825-1.

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

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

© ISO/IEC 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

6 Overview of the Directory service

As described in Rec. ITU-T X.501 | ISO/IEC 9594-2, the services of the Directory are provided through access points to directory user agents (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.

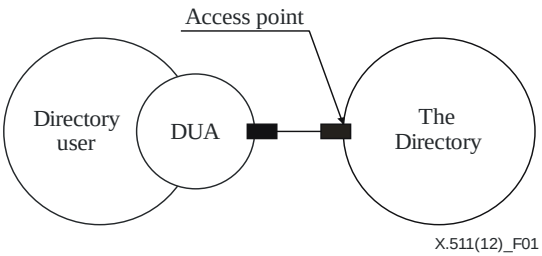


Figure 1 – Access to the Directory

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 10, 11, and 12, respectively. Conformance to Directory operations is specified in Rec. ITU-T 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. Clause 7.2 identifies these types and indicates the source of their definition. Each of the clauses (7.3 to 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 Rec. ITU-T 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`**.

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

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

a) **PresentationAddress**.

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

a) **Certificate**;

b) **SIGNED**;

c) **CertificationPath**.

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

a) **InvokeId**.

The following information types are defined in Rec. ITU-T 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,
  ... }
```

NOTE 1 – The above data type can only be used when included in set-constructs. An alternative data type **CommonArgumentsSeq** has been defined to be used in sequence-constructs (see Annex A).

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

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

The **requestor** component, when present, shall hold the distinguished name of the originator (requester) of the operation. If the distinguished name of the requester was established at bind time, the **requestor** component shall be equal to that distinguished name. Likewise, it shall be equal to the distinguished name in **subject** field of the end-entity public-key certificate of the requester if the **certification-path** component of the **SecurityParameters** is present.

NOTE 2 – The bound directory system agent (DSA) should check the equality of the distinguished names as indicated above (implementations based on Rec. ITU-T X.511 (2008) | ISO/IEC 9594-3:2008) or earlier editions may not do that).

NOTE 3 – If the distinguished name of the requester was not established at bind time and the **certification-path** component of the **SecurityParameters** is not present in the request, a possible value in the **requester** component should not be considered reliable for access control purposes.

The **operationProgress**, **referenceType**, **entryOnly**, **exclusions** and **nameResolveOnMaster** components are defined in Rec. ITU-T 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.

© ISO/IEC 2020 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

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 relative distinguished names (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 4 – This component is provided for compatibility with implementations based on Rec CCITT X.511 (1988) | IEC 9594:1990. 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 clause 7.6.1 and in clauses 8.9.2.2 and 12.8 of Rec. ITU-T 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 clause 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 clause 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 implementations based on the 1988 edition of the CCITT X.500 Recommendations series or the ISO/IEC 9594:1990 edition. 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 lightweight directory access protocol (LDAP) controls (if any) are shown in Table 1.

Table 1 – Extensions

Extension	Identifier	Operations	Criticality	Defined (clauses)	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	10.1	
pagedResultsRequest	6	List, Search	Non-critical	11.1	1.2.840.113556.1.4.319
matchedValuesOnly	7	Search	Non-critical	11.2	1.2.826.0.1.3344810.2.3
extendedFilter	8	Search	Non-critical	11.2	
targetSystem	9	Add Entry	Critical	12.1	
useAliasOnUpdate	10	Add Entry, Remove Entry, Modify Entry	Critical	12.1	
newSuperior	11	Modify DN	Critical	12.4	

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

Table 1 – Extensions

Extension	Identifier	Operations	Criticality	Defined (clauses)	LDAP control
manageDSAIT	12	All	Critical	7.5, 7.12	2.16.840.1.113730.3.4.2
Use of contexts	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	11.1.3 f)	
selectionOnModify	16	Modify Entry	Non-critical	12.3.2	
	17	Reserved		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	
	21-24	Reserved			
Service administration	25	Read, Search, ModifyEntry	Critical	11.2.2, 13, clause 16 of Rec. ITU-T X.501 | ISO/IEC 9594-2	
entryCount	26	Search	Non-critical	11.1.3	
hierarchySelections	27	Search	Non-critical	11.2.2	
relaxation	28	Search	Non-critical	7.8	
familyGrouping	29	Compare, Search, RemoveEntry	Non-critical Non-critical Critical	7.3.2, 7.8.3 & 10.2.2 11.2 12.2.2	
familyReturn	30	Read, Search, ModifyEntry	Non-critical Non-critical Non-critical	7.6.4, 7.7.1 & 10.1.3 11.2.3 12.3.3	
dnAttributes	31	Search	Non-critical	11.2.2	
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	12.3.1, 12.3.2	1.3.6.1.1.14
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.					

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),
```


© ISO/IEC 2020 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

```
  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 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.

The **SecurityParameters** component is specified in clause 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 clause 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 clause 6.13 of Rec. ITU-T X.520 | ISO/IEC 9594-6. Such notification attributes are not necessarily stored in directory entries.

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

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),
  allowWriteableCopy      (14)}
```

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

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 directory management domain (DMD).

d) **dontUseCopy** indicates that copied information as defined in Rec. ITU-T 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

© ISO/IEC 2020 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

copyShallDo is set and the Directory is not able to fully satisfy a query, it shall set **incompleteEntry** in 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 requester has access to the superior entry, then the existence of the context prefix entry will be indirectly disclosed to the requester 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 replication 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 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** service control (**low**, **medium**, or **high**) indicates the priority 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 queueing. There is no relationship implied with the use of priorities in underlying layers.

The **timeLimit** service control 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** service control 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 clause 7.9.

The **scopeOfReferral** service control 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** service control 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.

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

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** service control 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 requester 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 instance of the **EntryInformationSelection** data type indicates what information is being requested from an entry in a retrieval operation.

```
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,
    all                SET OF ContextAssertion,
    ...},
  ... }

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


© ISO/IEC 2020 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

The **attributes** component specifies the user and operational attributes for 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 for 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.

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 implementations based on Rec. CCITT X.511 (1988) | ISO/IEC 9594-3:1990) 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 clauses 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 clause 7.6.4).

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

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 clause 7.6.2), in order for that attribute value to be selected.

A **contextSelection** is said to govern one or more attribute types if any of the following conditions occur:

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

– the **ContextSelection** data type has a **selectedContexts** component which includes a set of **TypeAndContextAssertion** data types where the **type** component specifies an attribute type, including its subtypes, that is governed by the **contextAssertions** components; or

– the **ContextSelection** data type has a **selectedContexts** component which includes a **TypeAndContextAssertion** data type where the **type** component specifies the object identifier **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 directory information tree (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 clause 7.3 shall be applied if one is present and it governs the given attribute type as described above.

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 clause 14.7 of Rec. ITU-T 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 clause 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

© ISO/IEC 2020 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

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 clause 8.9.2.4 of Rec. ITU-T 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 clause 8.9.2.4 of Rec. ITU-T 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 clause 7.13.

The **memberSelect** component specifies which 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.

– **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 possibly have been explicitly unmarked 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 immediately 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 clause 16.10 of Rec. ITU-T 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,
  partialName       [4]   BOOLEAN DEFAULT FALSE,
  derivedEntry      [5]   BOOLEAN DEFAULT FALSE,
  ... }
```

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

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.

NOTE 1 – 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 2 – 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 chooses the same alias name for repeated requests by the same requester, 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.

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 (this component is not present for implementations based on Rec. CCITT X.511 (1988) | ISO/IEC 9594-3:1990).

The Directory shall complete the name resolution phase of operations in its entirety (including checking 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** component 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. This function is only available for implementations based on Rec. ITU-T X.511 (1997) | ISO/IEC 9594-3:1998 or later editions.

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. This function is only available for implementations based on Rec. ITU-T X.511 (2001) | ISO/IEC 9594-3:2001 or later editions.

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 a 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 the 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,
  ... }
```


© ISO/IEC 2020 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

```
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 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 clause 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,
  ... }

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
```

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

```
    -- one of the MATCHING-RULE information objects identified by matchingRule -- }),
  dnAttributes   [4]  BOOLEAN DEFAULT FALSE,
  ... }
```

A **Filter** is either a **FilterItem** (see clause 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 clause 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 identify 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 clause 8.7.2 of Rec. ITU-T X.520 | ISO/IEC 9594-6.

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 Rec. ITU-T X.520 | ISO/IEC 9594-6 for a description of the semantics of the presented value.

© ISO/IEC 2020 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

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 a 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 which has as value the attribute type for which the invalid matching rules were defined; and

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

NOTE 3 – An **extensibleMatch** is not permitted for implementations based on Rec. CCITT X.500 (1988) | ISO/IEC 9594-3:1990.

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 clause 8.9.2 of Rec. ITU-T 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 clause 8.9.2.2 of Rec. ITU-T 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 clause 7.13).

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.

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

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 a 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 clause 15.5.5 of Rec. ITU-T 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 may 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 clause 15.5.5 of Rec. ITU-T X.518 | 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,
  abandonQuer    [0]  OCTET STRING,
  ... }

SortKey ::= SEQUENCE {
  type          AttributeType,
  orderingRule  MATCHING-RULE.&id OPTIONAL,
  ... }
```


© ISO/IEC 2020 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

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 subordinates or 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 the 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 with 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 are 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 clause 10.3 of Rec. ITU-T 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 shall 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 shall return 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 that the user wants to start with a particular page rather than 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. An **abandoned** error with problem code **pagingAbandoned** shall be 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;

– 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.

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

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 implementations based on Rec. ITU-T X.511 (2005) | ISO/IEC 9594-3:2005 or later editions.

NOTE 6 – When a DAP association terminates, access to all associated paged results is lost. Paged results can only be accessed within the DAP application-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 requester 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,
  --                          [5]  Not to be used
  operationCode               [6]  Code OPTIONAL,
  --                          [7]  Not to be used
  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 public-key certificate framework defined in Rec. ITU-T X.509 | ISO/IEC 9594-8 is used in all Directory protocols defined in these Directory Specifications to optionally protect the operations including requests, responses and errors. Integrity protection is provided through the digital signature of the sender and the verification of that signature by the recipient using the sender's corresponding public-key certificate.

The **certification-path** component is defined in clause 7.7 in Rec. ITU-T X.509 | ISO/IEC 9594-8.). This component shall be present and contain the signer's public-key certificate if the request argument, response or error is signed. 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. The public-key certificate shall have the distinguished name in the subject field as specified by the name component.

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.

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.

© ISO/IEC 2020 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

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 requester'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 **operationCode** parameter is used to bind securely the operation code to the message (request arguments, results or errors). If this parameter is present, it shall take the value of the operation code for the operation and the receiver shall check that the value in this parameter is equal to the operation code in the received message. If not, the assumption is that the operation code has been changed and the message shall be discarded,

NOTE 2 –– The operation code is only securely bound to the message if the message is signed,

The **errorProtection** request may appear only in the request for an operation to be carried out, and indicates the requester'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 to 49 inclusive, the value shall have 2000 added to it.

– If the 2-digit value is 50 to 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 clause 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**.

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 a referral back 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 clause 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.

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

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 clause 10.2.3). If this permission is not granted, the Directory shall return an alias name for the entry instead, as described in clause 7.7, or it shall omit the **name** parameter altogether.

If a Read or Search operation, is not granted, the Directory shall return the name of an alias instead, as described in clause 7.7, or if no alias name is available, it 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**.

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 clause 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 requester 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.

© ISO/IEC 2020 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

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 requester 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 the same way as any other information, except that the access control information (ACI) for which the **`ProtectedItem`** is marked as is **`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 the DAP including the root DSE;
- attributes defined as no user modification may be modified (though the DSA can reply with a **`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 the 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;
   - 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 clause 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 which 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 matches 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

© ISO/IEC 2020 – All rights reserved

Documents Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

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 clause 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 can possibly 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 clause 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 originate 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.

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 clause 16.10.8 of Rec. ITU-T 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 Directory authentication

The Directory supports the authentication of users accessing the Directory via DUAs and the authentication of directory systems (DSAs) to users and to other DSAs. Depending on the environment, either simple or strong authentication may be used. The procedures to be used for simple and strong authentication in the Directory are described in the following subclauses.

8.1 Simple authentication procedure

Simple authentication is intended to provide local authorization based upon the distinguished name of a user, a bilaterally agreed (optional) password, and a bilateral understanding of the means of using and handling this password within a single domain. The utilization of simple authentication is primarily intended for local use only, i.e., for peer entity authentication between one DUA and one DSA or between one DSA and one DSA. Simple authentication may be achieved by several means:

a) the transfer of the user's distinguished name and (optional) password in the clear (non-protected) to the recipient for evaluation;

b) the transfer of the user's distinguished name, password and a random number and/or a timestamp, all of which are protected by applying a one-way function;

© ISO/IEC 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

c) the transfer of the protected information described in b) together with a random number and/or a timestamp, all of which is protected by applying a one-way function.

NOTE 1 – There is no requirement that the one-way functions applied be different.

NOTE 2 – The signalling of procedures for protecting passwords may be a matter for an extension to the document.

Where passwords are not protected, a minimal degree of security is provided for preventing unauthorized access. It should not be considered a basis for secure services. Protecting the user's distinguished name and password provides greater degrees of security. The algorithms to be used for the protection mechanism are typically non-enciphering one-way functions that are very simple to implement.

The general procedure for achieving simple authentication is shown in Figure 2.

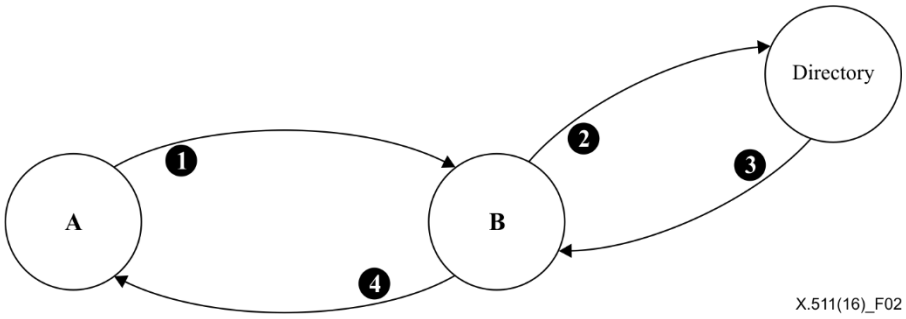


Figure 2 – The unprotected simple authentication procedure

The following steps are involved:

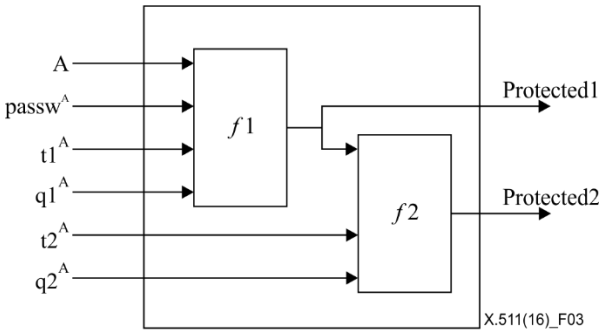
1) An originating user A sends its distinguished name and password to recipient user B.
2) B sends the purported distinguished name and password of A to the Directory, where the password is checked against that held as the **`UserPassword`** attribute within the directory entry for A (using the Compare operation of the Directory).
3) The Directory confirms (or denies) to B that the credentials are valid.
4) The success (or failure) of authentication may be conveyed to A.

The most basic form of simple authentication involves only step 1) and after B has checked the distinguished name and password, may include step 4).

8.1.1 Generation of protected identifying information

Figure 3 illustrates two approaches by which protected identifying information may be generated. *f*1 and *f*2 are one-way functions (either identical or different) and the timestamps and random numbers are optional and subject to bilateral agreements.

Annex E provides a suggested algorithm to be used for protected passwords.



A	User's distinguished name
t^A	Timestamps
$passw^A$	Password of A
q^A	Random numbers, optionally with a counter included

Figure 3 – Protected simple authentication

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

8.1.2 Procedure for protected simple authentication

Figure 4 illustrates the procedure for protected simple authentication.

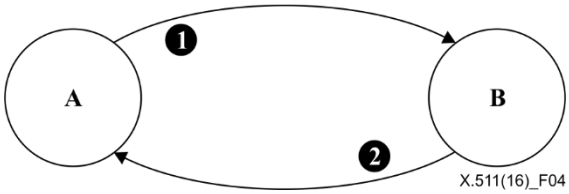


Figure 4 – The protected simple authentication procedure

The following steps are involved (initially using *f*1 only):

1) An originating user, user A, sends its protected identifying information (Authenticator1) to user B. Protection is achieved by applying the one-way function (f1) of Figure 3, where the timestamp and/or random number (when used) is used to minimize replay and to conceal the password.

   The protection of A's password is of the form:

   $\text{Protected1} = f1\ (t1^A, q1^A, A, \text{passw}^A)$

   The information conveyed to B is of the form:

   $\text{Authenticator1} = t1^A, q1^A, A, \text{Protected1}$

2) B verifies the protected identifying information offered by A by generating (using the distinguished name and optional timestamp and/or random number provided by A, together with a local copy of A's password) a local protected copy of A's password (of the form Protected1). B compares for equality the purported identifying information (Protected1) with the locally generated value.

3) B confirms or denies to A the verification of the protected identifying information.

The procedure can be modified to afford greater protection using *f*1 and *f*2. The main differences are as follows:

1) A sends its additionally protected identifying information (Authenticator2) to B. Additional protection is achieved by applying a further one-way function, *f*2, as illustrated in Figure 3. The further protection is of the form:

   $\text{Protected2} = f2\ (t2^A, q2^A, \text{Protected1})$

   The information conveyed to B is of the form:

   $\text{Authenticator2} = t1^A, t2^A, q1^A, q2^A, A, \text{Protected2}$

For comparison, B generates a local value of A's additionally protected password and compares it for equality with that of Protected2.

2) B confirms or denies to A the verification of the protected identifying information.

8.2 Password policy

8.2.1 Introduction

Password policy is a set of rules that controls how passwords are used and administered in the Directory. It improves the security of the Directory and makes it difficult for password cracking programs to break into the Directory. These rules ensure that users change their passwords periodically, that passwords meet quality requirements, that the reuse of old passwords is restricted, and that users are locked out after a certain number of failed attempts. This policy also forces the user to update its password after it has been set for the first time, or has been reset by a password administrator. However, in some cases, it is desirable to disallow users from adding and updating their own passwords.

A password is supposed not to be well known. If a password is frequently changed, the chance of misuse is minimized. Password policy administrators may deploy a password policy that causes passwords to expire after a given amount of time thus forcing users to change their passwords periodically. There must be a way to make users aware of the need to change their password before being locked out of their accounts. One or both of the following methods could be used:

– A warning may be returned to the user sometime before the password is due to expire. If the user ignores this warning before the expiration time, the account will be locked.

– The user may Bind to the directory a certain number of times after the password has expired. If the user fails to change the password following one of the 'grace' authentications, the account will be locked.

© ISO/IEC 2020 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

Password quality rules are rules for how a password shall be constructed. It is not the intention to provide a specification for password qualities, as requirements on quality may change over time. Password quality includes things like:

- minimum length;
- a mixture of characters (uppercase, lowercase, figures, punctuations, etc.); and
- avoidance of trivial passwords.

A particular quality rule requires specialized code within the implementation. It may therefore be of advantage to standardize password quality rules and assign object identifiers to such rules. An implementation may then claim support to one or more of such standardized quality rules.

An intruder may try to guess a password to get access to protected information. Currently, two different safeguards have been identified:

- Specification of the maximum number of failed attempts before a successful attempt within a given time span (which could be indefinitely): This approach allows for "denial of service attacks". One or more genuine users could have their access to the directory barred by the action of an attacker.
- The other mechanism is to insert a delay before returning information on authentication failure, and increasing this delay for repeated failed authentications on the same connection. This approach slows authentication, and makes brute force attacks impractical.

Password history is a mechanism to prevent password reuse. Previously used passwords should be stored to allow the Directory to ensure that a new password has not been previously used. Old passwords are stored for a time specified by the password policy, and after this time a password may be reused. The history is maintained in a **`userPwdHistory`** multi-valued operational attribute. A value is purged after a specific time, and the purged password may in principle be reused. The maximum time a password is kept in the **`userPwdHistory`** attribute is specified in the pwdMaxTimeInHistory operational attribute, and the minimum time is specified in the pwdMinTimeInHistory operational attribute. The number of passwords stored is limited by the **`pwdHistorySlots`** operational attribute and the password cannot be changed if there is no free slot in the history and no passwords in the history have been for less than the pwdMinTimeInHistory, so a user cannot revert to a "preferred password" simply by making lots of password changes.

The password policy can be used with clear passwords (using the **`clear`** alternative of the **`userPwd`** attribute), or with encrypted passwords (using the **`encrypted`** alternative of the **`userPwd`** attribute) or with another password attribute. All entries in the same specific password administrative area shall use the same password attribute type.

8.2.2 Operational attributes and procedures

The password policy uses specific operational attributes to register policy parameters, times and dates related to password management.

When a password value is first stored in the directory, in the **`userPwd`** attribute, the **`pwdStartTime`** operational attribute is set (Figure 5). The **`pwdExpiryTime`** operational attribute which contains the expiration of the password may either be automatically computed from the **`pwdExpiryAge`** operational attribute or set by explicit administrator action. It is an implementation option whether the value is dynamically computed by addition of the **`pwdExpiryAge`** to the **`pwdStartTime`** of the entry, in which case it does not need to be stored in the directory entry, or it is set by an administrator, in which case it shall be stored in the directory entry. The **`pwdEndTime`** operational attribute which contains the expiration of the account may either be automatically computed from the **`pwdMaxAge`** operational attribute or set by explicit administrator action. It is an implementation option whether the value is dynamically computed by addition of the **`pwdMaxAge`** to the **`pwdStartTime`** of the entry, in which case it does not need to be stored in the directory entry, or is set by an administrator, in which case it shall be stored in the directory entry.

The **`pwdStartTime`** operational attribute may also be set by an Administrator to specify that the account cannot be used before a given time.

When the user (or an administrator acting on behalf of the user) changes the **`userPwd`** attribute within the **`pwdMaxAge`** period, the **`pwdStartTime`** operational attribute should be updated. The **`pwdExpiryTime`** and the **`pwdEndTime`** operational attributes should be recomputed and updated to reflect the new password creation time.

NOTE – If a user does bind with the Directory for a long time, the values of pwdExpiryTime and pwdEndTime operational attributes may be exceeded and the account automatically locked.

© ISO/IEC 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

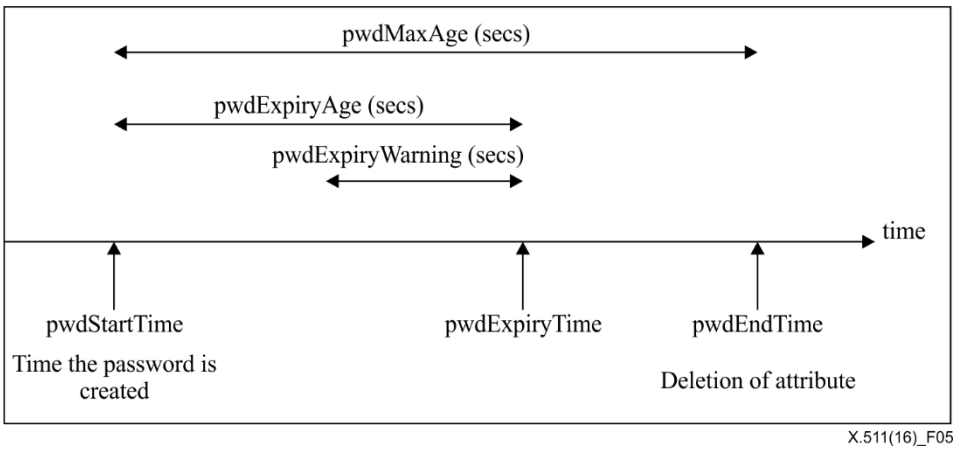


Figure 5 – Time chart for password attributes

When the user (or an administrator acting on behalf of the user) changes the value of the password, the new value is generally not immediately known by all the Directory servers because of replication delays. To prevent authentication problems, the previous password remains available for the **`pwdRecentlyExpiredDuration`** duration time (which shall be greater than the replication periods used in the Directory system).

When the user (or an administrator acting on behalf of the user) changes the value of the password, the old value should be copied into the recently expired password attribute. (The **`userPwd`** attribute is copied into the **`userPwdRecentlyExpired`**). When the recently expired password duration time is over, the recently expired password attribute (**`userPwdRecentlyExpired`**) should not be available to the user. If the user (or an administrator acting on behalf of the user) changes their password again during the recently expired password duration time, then their recently expired password should be overwritten and the duration should be set to start again (see Figure 6). Thus, a recently expired password will only be kept in the recently expired password attribute for the shorter of the recently expired password duration time or until the user changes their password again. However, it will be kept in the password history table.

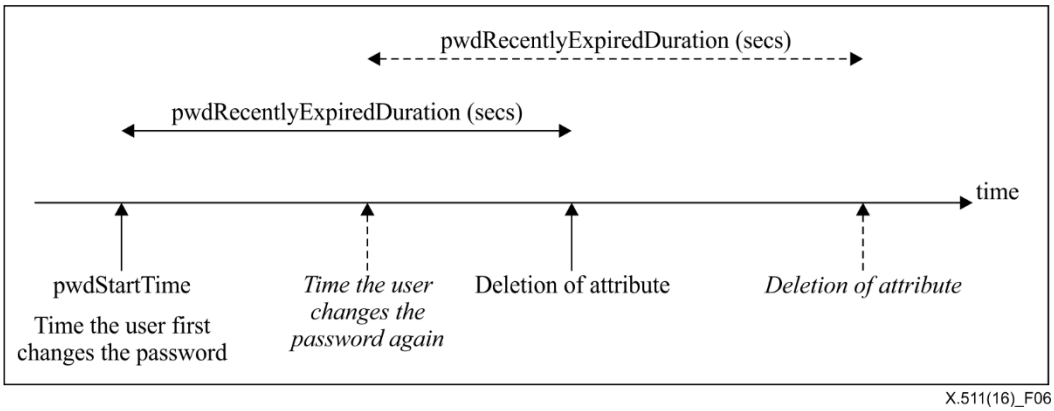


Figure 6 – Time line for recently expired passwords

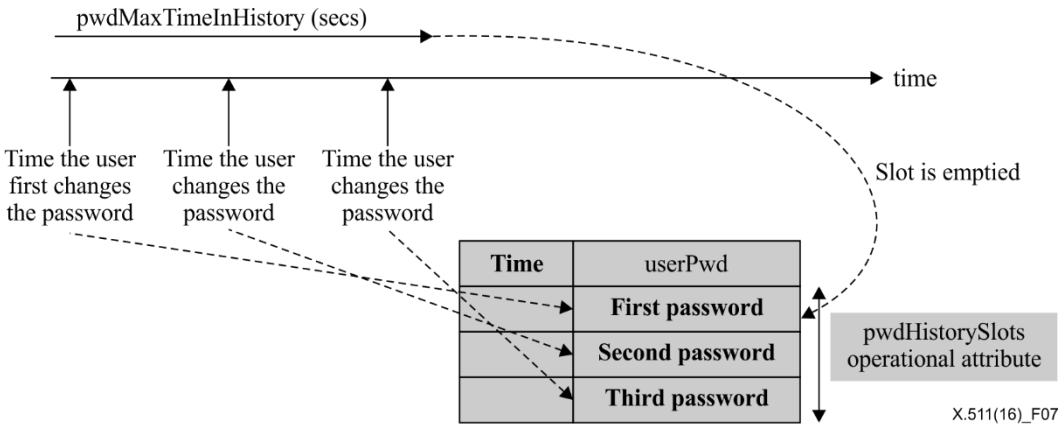
8.2.3 Password history

The password history attribute is used to prevent password reuse, by storing old values of the user's password so that the user cannot reuse the same password again whilst it is stored in the password history (see Figure 7). When the user (or an administrator acting on behalf of the user) changes their password, it may be copied into the password history (**`userPwdHistory`**) operational attribute along with the time that the password was changed. The password maximum time in history attribute (**`pwdMaxTimeInHistory`**) specifies the maximum duration (in seconds) that a password should remain in the password history. Once this time has expired for a particular password, then it is removed from the password history, and the user may use this password again.

The number of slots in the password history table (or password history attribute values) is defined in the **`pwdHistorySlots`** operational attribute. When all the slots are filled, the oldest password may be removed subject to it having been in the history for a minimum duration time (as specified in the **`pwdMinTimeInHistory`** attribute).

© ISO/IEC 2020 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020

NOTE – If an administrator has to change the password of a user when all the history slots are full and no passwords are older than **pwdMinTimeInHistory**, then the administrator might free two slots in the history table (i.e., delete two attribute values), reset the user's password to a temporary value (which is copied into the history), leaving one spare slot for the user to choose their own new password.



NOTE – If the Directory system changes its encryption algorithm then a user will be able to use the same password again since the encrypted password will be different.

Figure 7 – userPwdHistory attribute

9 Bind, Unbind operations, Change Password and Administer Password operations

The Directory Bind and Directory Unbind operations, defined in clauses 9.1 and 9.2 respectively, are used by the DUA at the beginning and end of a particular period of accessing the Directory.

9.1 Directory Bind

9.1.1 Directory Bind syntax

A Directory Bind operation is used at the beginning of a period of accessing the Directory.

```
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 {
    unprotected    OCTET STRING,
```

© ISO/IEC 2020 – All rights reserved

Documents Document Sharing Hub
Full document details: ISO_IEC 9594-3_2020