

Information technology — Open Systems Interconnection — The Directory: Procedures for distributed operation

(ITU-T Rec. X.518 (1993) | ISO/IEC 9594-4 : 1995)

ICS 35.100.70

National foreword

This British Standard reproduces verbatim ISO/IEC 9594-4 : 1995 and implements it as the UK national standard. BS ISO/IEC 9594-4 : 1990 remains current because it applies to existing products that are still being manufactured and which are not covered in this standard. It is envisaged that BS ISO/IEC 9594-4 : 1990 will be withdrawn in due course.

Customer Services will respond to orders for BS ISO/IEC 9594-4 by supplying the latest edition. If copies of BS ISO/IEC 9594-4 : 1990 are required, orders should state clearly that BS ISO/IEC 9594-4 : 1990 is required.

The UK participation in its preparation was entrusted to Technical Committee IST/21, Open Systems Interconnection, Data Management and Open Distributed Processing, which has the responsibility to:

- aid enquirers to understand the text;
- present to the responsible international/European committee any enquiries on the interpretation, or proposals for change, and keep the UK interests informed;
- monitor related international and European developments and promulgate them in the UK.

A list of organizations represented on this committee can be obtained on request.

Cross-references

The British Standards which implement international or European publications referred to in this document may be found in the BSI Standards Catalogue under the section entitled 'International Standards Correspondence Index', or using the 'Find' facility of the BSI Standards Electronic Catalogue.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

Summary of pages

This document comprises a front cover, an inside front cover, the ISO/IEC title page, pages ii to vi, pages 1 to 102, an inside back cover and a back cover.

This British Standard, having been prepared under the direction of the DISC Board, was published under the authority of the Standards Board and comes into effect on 15 February 1997

© BSI 1997

ISBN 0 580 26552 8

Amendments issued since publication

Amd. No.	Date	Text affected

INTERNATIONAL
STANDARD

ISO/IEC
9594-4

Second edition
1995-09-15

**Information technology — Open Systems
Interconnection — The Directory:
Procedures for distributed operation**

*Technologies de l'information — Interconnexion de systèmes ouverts
(OSI) — L'Annuaire: Procédures pour le fonctionnement réparti*



Reference number
ISO/IEC 9594-4:1995(E)

AVA **Documents**
Document Sharing Hub

Contents

	<i>Page</i>
SECTION 1 – GENERAL.....	1
1 Scope	1
2 Normative references	1
2.1 Identical Recommendations International Standards	1
2.2 Paired Recommendations International Standards equivalent in technical content	2
3 Definitions	2
3.1 OSI Reference Model Definitions.....	2
3.2 Basic Directory Definitions	2
3.3 Directory Model Definitions	2
3.4 DSA Information Model definitions	2
3.5 Directory replication definitions	3
3.6 Distributed operation definitions	3
4 Abbreviations	4
5 Conventions.....	5
SECTION 2 – OVERVIEW	5
6 Overview	5
SECTION 3 – DISTRIBUTED DIRECTORY MODELS	6
7 Distributed Directory System Model	6
8 DSA Interactions Model.....	7
8.1 Decomposition of a request.....	7
8.2 Uni-chaining	7
8.3 Multi-chaining.....	7
8.4 Referral	9
8.5 Mode Determination	9
SECTION 4 – DSA ABSTRACT SERVICE.....	11
9 Overview of DSA Abstract Service	11
10 Information types	11
10.1 Introduction.....	11
10.2 Information types defined elsewhere	11
10.3 Chaining Arguments	12
10.4 Chaining Results	13
10.5 Operation Progress.....	14
10.6 Trace Information	14
10.7 Reference Type	15
10.8 Access point information	15
10.9 Exclusions.....	15
10.10 Continuation Reference.....	16

Descriptors: data processing, information interchange, network interconnection, open systems interconnection, application layer, directories, procedure.

11	Bind and Unbind	17
11.1	DSA Bind.....	17
11.2	DSA Unbind	18
12	Chained operations.....	18
12.1	Chained operations.....	18
12.2	ChainedAbandon operation	19
13	Chained errors	19
13.1	Introduction.....	19
13.2	DSA Referral	19
SECTION 5 – DISTRIBUTED PROCEDURES		20
14	Introduction	20
14.1	Scope and Limits.....	20
14.2	Conformance.....	20
14.3	Conceptual model	20
14.4	Individual and cooperative operation of DSAs.....	20
14.5	Cooperative agreements between DSAs	21
15	Distributed Directory behavior.....	21
15.1	Cooperative fulfillment of operations	21
15.2	Phases of operation processing	21
15.3	Managing Distributed Operations	22
15.4	Loop handling	22
15.5	Other considerations for distributed operation.....	23
15.6	Authentication of Distributed Operations	24
16	The Operation Dispatcher	24
16.1	General Concepts	25
16.2	Procedures of the operation dispatcher	28
16.3	Overview of procedures.....	29
17	Request Validation	30
17.1	Introduction.....	30
17.2	Procedure parameters.....	30
17.3	Procedure definition.....	32
18	Name Resolution	33
18.1	Introduction.....	33
18.2	Find DSE procedure parameters	33
18.3	Procedures.....	34
19	Operation evaluation	43
19.1	Modification procedure.....	43
19.2	Single entry interrogation procedure.....	49
19.3	Multiple entry interrogation procedure	50

20	Continuation Reference procedures	59
20.2	Issuing chained sub-requests to a remote DSA.....	61
20.3	Procedures' parameters	61
20.4	Definition of the Procedures	62
20.5	Abandon procedure	70
21	Results Merging procedure	70
22	Procedures for distributed authentication.....	72
22.1	Originator authentication	73
22.2	Results authentication	73
	SECTION 6 – KNOWLEDGE ADMINISTRATION	74
23	Knowledge administration overview	74
23.1	Maintenance of Knowledge References.....	74
23.2	Requesting cross reference.....	75
23.3	Knowledge inconsistencies	76
24	Hierarchical operational bindings	77
24.1	Operational binding type characteristics.....	77
24.3	DSA procedures for hierarchical operational binding management	79
24.4	Procedures for operations	83
24.5	Use of application contexts	83
25	Non-specific hierarchical operational binding	84
25.1	Operational binding type characteristics.....	84
25.2	Operational binding information object class definition.....	85
25.3	DSA procedures for non-specific hierarchical operational binding management	85
25.4	Procedures for operations	87
25.5	Use of application contexts	87
	Annex A – ASN.1 for Distributed Operations	88
	Annex B – Example of distributed name resolution.....	91
	Annex C – Distributed use of authentication.....	93
C.1	Summary	93
C.2	Simple authentication.....	93
C.3	Distributed authentication model	93
C.4	DUA to DSA.....	94
C.5	Transference from the DAP to the DSP	94
C.6	Chaining through intermediate DSAs	94
C.7	Results authentication	94
	Annex D – Specification of hierarchical and non-specific hierarchical operational binding types.....	97
	Annex E – Knowledge maintenance example	99
	Annex F – Amendments and corrigenda	102

Foreword

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

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

International Standard ISO/IEC 9594-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 21, *Open systems interconnection, data management and open distributed processing*, in collaboration with ITU-T. The identical text is published as ITU-T Recommendation X.518.

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

This second edition technically revises and enhances ISO/IEC 9594-4:1990. It also incorporates technical corrigendum 1:1991, technical corrigendum 2:1992 and technical corrigendum 3:1993. Implementations may still claim conformance to the first edition of this part of ISO/IEC 9594. However, at some point, the first edition will no longer be supported (i.e. reported defects will no longer be resolved). It is recommended that implementations conform to this second edition as soon as possible.

ISO/IEC 9594 consists of the following parts, under the general title *Information technology — Open Systems Interconnection — The Directory*:

- *Part 1: Overview of concepts, models and services*
- *Part 2: Models*
- *Part 3: Abstract service definition*
- *Part 4: Procedures for distributed operation*
- *Part 5: Protocol specifications*
- *Part 6: Selected attribute types*
- *Part 7: Selected object classes*
- *Part 8: Authentication framework*
- *Part 9: Replication*

Annex A forms an integral part of this part of ISO/IEC 9594. Annexes B to F are for information only.

Introduction

This Recommendation | International Standard part together with other Recommendations | International Standards, has been produced to facilitate the interconnection of information processing systems to provide directory services. A set of such systems, together with the directory information which they hold, can be viewed as an integrated whole, called the *Directory*. The information held by the Directory, collectively known as the Directory Information Base (DIB), is typically used to facilitate communication between, with or about objects such as application entities, people, terminals and distribution lists.

The Directory plays a significant role in Open Systems Interconnection, whose aim is to allow, with a minimum of technical agreement outside of the interconnection standards themselves, the interconnection of information processing systems:

- from different manufacturers;
- under different managements;
- of different levels of complexity; and
- of different ages.

This Recommendation | International Standard specifies the procedures by which the distributed components of the Directory interwork in order to provide a consistent service to its users.

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

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

Annex A, which is an integral part of this Recommendation | International Standard, provides the ASN.1 module for directory distributed operations.

Annex B, which is not an integral part of this Recommendation | International Standard, describes an example of distributed name resolution.

Annex C, which is not an integral part of this Recommendation | International Standard, describes authentication in the distributed operations environment.

Annex D, which is an integral part of this Recommendation | International Standard, provides the definitions of the ASN.1 information object classes introduced in this Directory Specification.

Annex E, which is not an integral part of this Recommendation | International Standard, illustrates knowledge maintenance.

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.

INFORMATION TECHNOLOGY – OPEN SYSTEMS INTERCONNECTION – THE DIRECTORY: PROCEDURES FOR DISTRIBUTED OPERATION

SECTION 1 – GENERAL

1 Scope

This Recommendation | International Standard specifies the behavior of DSAs taking part in the distributed Directory application. The allowed behavior has been designed so as to ensure a consistent service given a wide distribution of the DIB across many DSAs.

The Directory is not intended to be a general purpose database system, although it may be built on such systems. It is assumed that there is a considerably higher frequency of queries than of updates.

2 Normative references

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

2.1 Identical Recommendations | International Standards

- ITU-T Recommendation X.500 (1993) | ISO/IEC 9594-1:1995, *Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services.*
- ITU-T Recommendation X.501 (1993) | ISO/IEC 9594-2:1995, *Information technology – Open Systems Interconnection – The Directory: Models.*
- ITU-T Recommendation X.511 (1993) | ISO/IEC 9594-3:1995, *Information technology – Open Systems Interconnection – The Directory: Abstract service definition.*
- ITU-T Recommendation X.519 (1993) | ISO/IEC 9594-5:1995, *Information technology – Open Systems Interconnection – The Directory: Protocol specifications.*
- ITU-T Recommendation X.520 (1993) | ISO/IEC 9594-6:1995, *Information technology – Open Systems Interconnection – The Directory: Selected attribute types.*
- ITU-T Recommendation X.521 (1993) | ISO/IEC 9594-7:1995, *Information technology – Open Systems Interconnection – The Directory: Selected object Classes.*
- ITU-T Recommendation X.509 (1993) | ISO/IEC 9594-8:1995, *Information technology – Open Systems Interconnection – The Directory: Authentication framework.*
- ITU-T Recommendation X.525 (1993) | ISO/IEC 9594-9:1995, *Information technology – Open Systems Interconnection – The Directory: Replication.*

- ITU-T Recommendation X.680 (1994) | ISO/IEC 8824-1:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*
- ITU-T Recommendation X.681 (1994) | ISO/IEC 8824-2:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.*
- ITU-T Recommendation X.682 (1994) | ISO/IEC 8824-3:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*
- ITU-T Recommendation X.683 (1994) | ISO/IEC 8824-4:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Parametrization of ASN.1 specifications.*
- ITU-T Recommendation X.880 (1994) | ISO/IEC 13712-1:1995, *Information technology – Remote Operations: Concepts, model and notation.*
- ITU-T Recommendation X.881 (1994) | ISO/IEC 13712-2:1995, *Information technology – Remote Operations: OSI realizations – Remote Operations Service Element (ROSE) service definition.*

2.2 Paired Recommendations | International Standards equivalent in technical content

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

3 Definitions

For the purpose of this Recommendation | International Standard the following definitions apply:

3.1 OSI Reference Model Definitions

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

- *application entity title.*

3.2 Basic Directory Definitions

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

- a) *(the) Directory;*
- b) *Directory Information Base.*

3.3 Directory Model Definitions

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

- a) *access point;*
- b) *alias;*
- c) *distinguished name;*
- d) *Directory Information Tree;*
- e) *Directory System Agent;*
- f) *Directory User Agent;*
- g) *relative distinguished name.*

3.4 DSA Information Model definitions

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

- a) *category;*
- b) *commonly usable;*
- c) *context prefix;*

- d) *cross reference*;
- e) *DIB fragment*;
- f) *DSA information tree*;
- g) *DSA Specific Entry (DSE)*;
- h) *DSE type*;
- i) *immediate superior reference*;
- j) *knowledge information*;
- k) *knowledge reference category*;
- l) *knowledge reference type*;
- m) *naming context*;
- n) *non-specific knowledge*;
- o) *non-specific subordinate reference*;
- p) *operational attribute*;
- q) *reference path*;
- r) *specific knowledge*;
- s) *subordinate reference*;
- t) *superior reference*.

3.5 Directory replication definitions

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

- a) *attribute completeness*;
- b) *shadowing operational binding*;
- c) *subordinate completeness*;
- d) *unit of replication*.

3.6 Distributed operation definitions

The following terms are defined in this Recommendation | International Standard:

- 3.6.1 base object:** The object or alias entry that is the target for an operation as issued by the originator.
- 3.6.2 chaining:** The generic term for uni-chaining or multi-chaining.
- 3.6.3 context prefix information:** Operational and user information supplied by the superior DSA to the subordinate DSA in a RHOB regarding DIT vertices superior to the subordinate context prefix.
- 3.6.4 distributed name resolution:** The process by which name resolution is performed in more than one DSA.
- 3.6.5 error:** Information sent from the performer to the requester conveying a negative outcome of a previously received request.
- 3.6.6 hard error:** A definite error which indicates that the operation cannot currently be performed without external intervention.
- 3.6.7 hierarchical operational binding (HOB):** Relationship between two master DSAs holding naming contexts, one of which is immediately subordinate to the other, in which the superior DSA holds a subordinate reference to the subordinate DSA.
- 3.6.8 modification operations:** These are the Directory Modify Operations, i.e. Modify Entry, Add Entry, Remove Entry and ModifyDN.
- 3.6.9 multi-chaining:** A mode of interaction in which a DSA processing a request itself sends multiple requests either in parallel or sequentially to a set of other DSAs.
- 3.6.10 multiple entry interrogation operations:** These are the Directory Search Operations, i.e. List and Search.
- 3.6.11 name resolution:** The process of locating an entry by sequentially matching each RDN in a purported name to a vertex of the DIT.

3.6.12 non-specific hierarchical operational binding (NHOB): Relationship between two master DSAs holding naming contexts, one of which is immediately subordinate to the other, in which the superior DSA holds a non-specific subordinate reference to the subordinate DSA.

3.6.13 NSSR decomposition: Decomposition of non-specific knowledge references into subrequests for other DSAs to pursue; these subrequests may be either chained to these DSAs by the DSA performing the decomposition, or a continuation reference identifying the DSAs may be returned to the requester for it to pursue, or the decomposing DSA may pursue some of the subrequests, leaving others unexplored for the requester to pursue.

3.6.14 operation progress: A set of values which denotes the extent to which name resolution has taken place.

3.6.15 originator: The DUA that has initiated a specific (distributed) operation.

3.6.16 performer: DSA receiving a request (i.e. to perform an operation).

3.6.17 procedure: An (informal) specification of how a DSA maps a given set of input arguments and its DSA information tree into a result.

NOTE – Input arguments and results may correspond to information received in a requested operation and information sent in a reply, or they may represent intermediate stages in the computation of a reply from a requested operation. In 14.2 the former variety of input arguments and results are termed external.

3.6.18 relevant hierarchical operational binding (RHOB): Either a HOB or a NHOB, depending on the context.

3.6.19 referral: An outcome which can be returned by a DSA which cannot perform an operation itself, and which identifies one or more other DSAs more able to perform the operation.

3.6.20 reply: A result or an error.

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

3.6.22 request decomposition: Decomposition of a request into subrequests for other DSAs to pursue; these subrequests may be either chained to these DSAs by the DSA performing the decomposition, or continuation references identifying the DSAs may be returned to the requester for it to pursue, or the decomposing DSA may pursue some of the subrequests, leaving others unexplored for the requester to pursue.

3.6.23 requester: A DUA or DSA sending a request to perform (i.e. invoke) an operation.

3.6.24 single entry interrogation operations: These are the Directory Read Operations, i.e. Read and Compare.

3.6.25 soft error: An error which may be transient, or which may indicate a localized problem, in which case the use of a different knowledge reference or access point may enable a result or hard error to be obtained.

3.6.26 subordinate DSA: Of the two DSAs sharing a HOB or a NHOB, the DSA holding the subordinate naming context.

3.6.27 subrequest: A request generated by request decomposition.

3.6.28 superior DSA: Of the two DSAs sharing a HOB or a NHOB, the DSA holding the superior naming context.

3.6.29 superior, subordinate DSA: Two master DSAs holding naming contexts, one of which is immediately subordinate to the other; the relationship between the two DSAs is managed explicitly via a HOB (or NHOB), or exists implicitly by virtue of the superior DSA holding a subordinate (or non-specific subordinate) reference to the subordinate DSA.

3.6.30 target object name: The name of an entry either to which the operation is to be directed at a particular stage of name resolution, or which is involved in the evaluation of the operation.

3.6.31 uni-chaining: A mode of interaction optionally used by a DSA which cannot perform an operation itself. The DSA *chains* by invoking an operation of another DSA and then relaying the outcome to the original requester.

4 Abbreviations

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

ASN.1	Abstract Syntax Notation One
DOP	Directory Operational Binding Management Protocol
DISP	Directory Information Shadowing Protocol

DMD	Directory Management Domain
DSE	DSA Specific Entry
HOB	Hierarchical Operational Binding
NHOB	Non-specific Hierarchical Operational Binding
NSSR	Non-specific Subordinate Reference
RHOB	Relevant Hierarchical Operational Binding

5 Conventions

With minor exceptions this Directory Specification has been prepared according to the “Presentation of ITU-TS/ISO/IEC common text” guidelines in the Guide for ITU-TS and ISO/IEC JTC 1 Cooperation, March 1993.

The term “Directory Specification” (as in “this Directory Specification”) shall be taken to mean ITU-T Rec. X.518 | ISO/IEC 9594-4. The term “Directory Specifications” shall be taken to mean the X.500-Series Recommendations and all parts of ISO/IEC 9594.

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

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

This Directory Specification defines directory operations using the Remote Operation notation defined in ITU-T Rec. X.880 | ISO/IEC 9072-1.

SECTION 2 – OVERVIEW

6 Overview

The Directory Abstract Service allows the interrogation, retrieval and modification of Directory information in the DIB. This service is described in terms of the abstract Directory object as specified in ITU-T Rec. X.511 | ISO/IEC 9594-3.

Necessarily, the specification of the abstract Directory object does not in any way address the physical realization of the Directory: in particular it does not address the specification of Directory System Agents (DSA) within which the DIB is stored and managed, and through which the service is provided. Furthermore, it does not consider whether the DIB is centralized, i.e. contained within a single DSA, or distributed over a number of DSAs. Consequently, the requirements for DSAs to have knowledge of, navigate to, and cooperate with other DSAs, in order to support the abstract service in a distributed environment is also not covered by the service description.

This Directory Specification specifies the refinement of the abstract Directory object, the refinement being expressed in terms of a set of one or more DSA objects which collectively constitute the distributed directory service.

In addition this Directory Specification specifies the permissible ways in which the DIB may be distributed over one or more DSAs. For the limiting case where the DIB is contained within a single DSA, the Directory is in fact centralized; for the case where the DIB is distributed over two or more DSAs, knowledge and navigation mechanisms are specified which ensure that the whole of the DIB is potentially accessible from all DSAs that hold constituent entries.

Portions of the DIB may also be replicated in multiple DSAs. The protocols described in this Directory Specification allow the use of replicated information to improve the availability, performance and efficiency of the distributed directory service. The use of replicated information is, to some extent, under the user’s control, through the use of service control options. The procedures described in this Directory Specification also indicate some of the opportunities for design optimizations when using the replicated information.

Additionally, request handling interactions are specified that enable particular operational characteristics of the Directory to be controlled by its users. In particular, the user has control over whether a DSA, responding to a directory inquiry pertaining to information held in other DSA(s), has the option of interrogating the other DSA(s) directly (chaining) or, whether it should respond with information about other DSA(s) which could further progress the inquiry (referral).

Generally, the decision by a DSA to chain or refer is determined by the service controls set by the user, and by the DSA's own administrative, operational or technical circumstances.

Recognizing that, in general, the Directory will be distributed, and that directory inquiries will be satisfied by an arbitrary number of cooperating DSAs which may arbitrarily chain or refer according to the above criteria, this Directory Specification specifies the appropriate procedures to be effected by DSAs in responding to distributed directory inquiries. These procedures will ensure that users of the distributed Directory service perceive it to be both user-friendly and consistent.

SECTION 3 – DISTRIBUTED DIRECTORY MODELS

7 Distributed Directory System Model

The Directory abstract service as defined in ITU-T Rec. X.511 | ISO/IEC 9594-3 models the Directory as an object which provides a set of directory services to its users. Users of the Directory access its services through an access point. The Directory may have one or more access points and each access point is characterized by the services it provides and the mode of interaction used to provide these services.

Figure 1 illustrates the distributed directory model which will be used as the basis for specifying the distributed aspects of the directory. It illustrates the Directory as comprising a set of one or more DSAs.

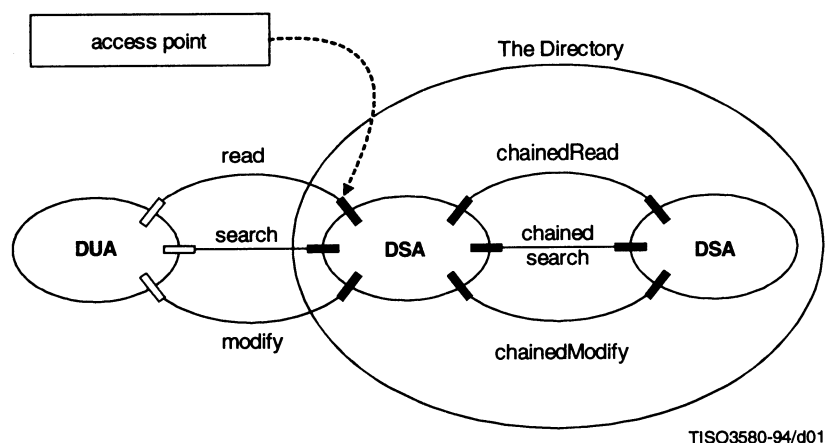


Figure 1 – Objects of the distributed Directory model

DSAs are specified in detail in the subsequent clauses of this Directory Specification. This clause merely states a number of their characteristics in order to serve as an introduction and to establish the relationship between this Directory Specification and the other Directory Specifications.

DSAs are defined in order that distribution of the DIB can be accommodated and that a number of physically distributed DSAs can interact in a prescribed, cooperative manner to provide directory services to the users of the directory (DUAs).

Figure 1 illustrates the relationship between the Directory abstract service and the DSA abstract service. The Directory abstract service defined in ITU-T Rec. X.511 | ISO/IEC 9594-3 is provided through a number of Directory operations. To realize this service, the DSAs that comprise the Directory interact with one another. The nature of this interaction is defined in terms of the service that one DSA may provide to another DSA, the DSA abstract service. The DSA abstract service is provided through a number of operations, termed chained operations, each having a counterpart in the Directory abstract service. Thus a given operation in the Directory abstract service, e.g. Read, may require that the DSA providing the service interact with one or more other DSAs using chained operations, e.g. Chained Read.

8 DSA Interactions Model

A basic characteristic of the Directory is that, given a distributed DIB, a user should potentially be able to have any service request satisfied (subject to security, access control, and administrative policies) irrespective of the access point at which the request originates. In accommodating this requirement, it is necessary that any DSA involved in satisfying a particular service request have some knowledge (as specified in ITU-T Rec. X.501 | ISO/IEC 9594-2) of where the requested information is located and either return this knowledge to the requester or attempt to have the request satisfied on its behalf. (The requester may either be a DUA or another DSA: in the latter case both DSAs shall support the DSP.)

Three modes of DSA interaction are defined to meet these requirements, namely “uni-chaining”, “multi-chaining”, and “referral”. Throughout the remainder of this Directory Specification, the generic term chaining is used to refer to uni-chaining and/or multi-chaining as appropriate to the context. “Chaining” refers to the attempt by a DSA to satisfy a request by sending one or more chained operations to other DSAs; “referral”, to the return of knowledge information to the requester, which may then itself interact with the DSA(s) identified in the knowledge information.

Uni-chaining or a referral interaction may result from a single request. Alternatively, the request may be decomposed into several subrequests prior to the interaction. Multi-chaining or referral interactions, or a mixture of the two, may result from a decomposed request. Two types of decomposition are defined; NSSR decomposition and request decomposition.

8.1 Decomposition of a request

8.1.1 NSSR decomposition

NSSR decomposition is the process of preparing identical requests ready for transfer (either sequentially or in parallel) to several subordinate DSAs as a result of encountering an NSSR during name resolution. Non-specific subordinate references do not hold the RDNs of the referenced subordinate naming contexts, so the referencing DSA is unable to tell which subordinate DSA holds which subordinate naming context(s). During name resolution a DSA encountering NSSRs shall send an identical request to each subordinate DSA (in the absence of shadowing). This may be done sequentially or in parallel. Typically, only one DSA will be able to continue with name resolution; the others will return the Service Error **unableToProceed**. In certain (rare) circumstances it is possible that more than one DSA will continue with name resolution, giving rise to duplicate results.

8.1.2 Request decomposition

Request decomposition, the other form of decomposing a request, is a process performed internally by a DSA prior to communication with one or more other DSAs. A request is decomposed into several, possibly different, sub-requests such that each of the sub-requests accomplishes a part of the original task. Request decomposition can be used only during operation evaluation of a List or Search. After request decomposition, each of the sub-requests may then be chained to other DSAs to continue the task, or a partial result (an embedded referral) may be returned to the requester. An example of the same sub-request being generated to different DSAs is when an entry has subordinate references and/or NSSRs that together reference more than one DSA. An example of different sub-requests being generated to the same or different DSAs is when two different entries are encountered during a Search (subtree), and each has a subordinate reference.

8.2 Uni-chaining

This mode of interaction (depicted in Figure 2) may be used by one DSA to pass on a request to another DSA when the former has knowledge about naming contexts held by the latter. Uni-chaining may be used to contact a single DSA pointed to in a cross reference, a subordinate reference, a superior reference, a supplier reference, or a master reference.

NOTE – In Figure 2, the order of interactions is defined by the numbers associated with the interaction lines.

8.3 Multi-chaining

This mode of interaction is used by a DSA for transferring several outgoing requests which have resulted from one incoming request, as a result of either request decomposition or NSSR decomposition.

8.3.1 Parallel multi-chaining

With parallel multi-chaining, the DSA transfers several outgoing requests simultaneously (see Figure 3a). Whilst parallel multi-chaining may give improved performance, it may under certain circumstances, e.g. in the presence of shadowing, cause duplicate results to be received.

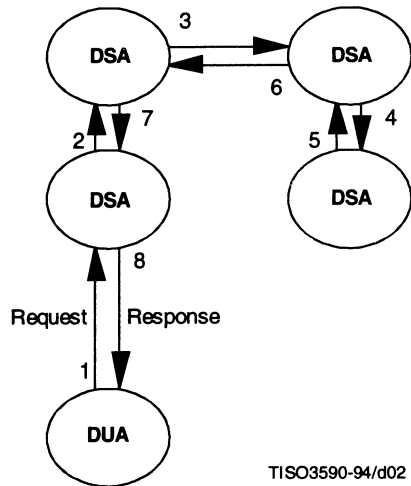


Figure 2 – Uni-chaining mode

8.3.2 Sequential multi-chaining

With sequential multi-chaining, the DSA transfers one outgoing request at a time and waits for the result or error of one request before sending the next (see Figure 3b). Whilst sequential multi-chaining may not be the quickest mode of interaction, it is unlikely that duplicate results will be received.

NOTE – A DSA may use a combination of parallel multi-chaining and sequential multi-chaining.

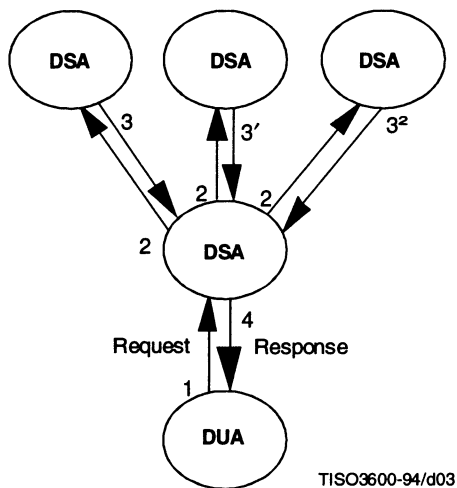
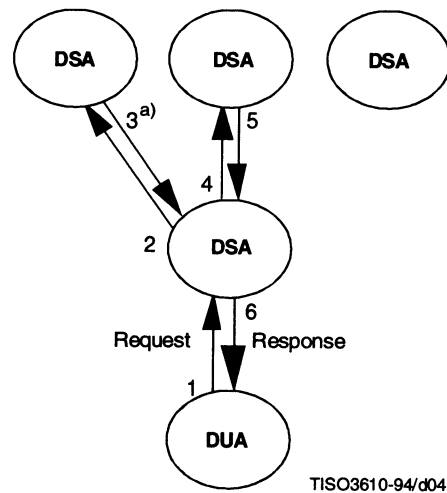


Figure 3a – Parallel Multi-chaining



TISO3610-94/d04

a) Unable to proceed.

**Figure 3b – Sequential Multi-chaining
(as a result of NSSR decomposition)**

8.4 Referral

A referral (depicted in Figures 4a and 4b) is returned by a DSA in response to a request from either a DUA or another DSA. The referral may constitute the whole response (in which case it is categorized as an error) or just part of the response. The referral contains a knowledge reference, which may be either a superior, subordinate, cross, non-specific subordinate, supplier, or master reference.

The DSA (Figure 4a) receiving the referral may use the knowledge reference contained therein, to subsequently chain or multi-cast (depending upon the type of reference) the original request to other DSAs. Alternatively, a DSA receiving a referral, may in turn pass the referral back in its response. A DUA (Figure 4b) receiving a referral may use it to contact one or more other DSAs to progress the request.

NOTE – In Figures 4a and 4b, the order of interactions is defined by the numbers associated with the interaction lines.

8.5 Mode Determination

If a DSA cannot itself fully resolve a request, it shall chain the request (or a request formed by decomposing the original one), to another DSA, unless:

- a) chaining is prohibited by the user via the service controls, in which case the DSA shall return a referral or a **chainingRequired ServiceError**; or
- b) the DSA has administrative, operational, or technical reasons for preferring not to chain, in which case the DSA shall return a referral.

NOTES

- 1 A “technical reason” for not chaining is that the DSA identified in the knowledge reference does not support the DSP.
- 2 If the **localScope** service control is set, then the DSA (or DMD) shall either resolve the request or return an error.
- 3 If the user prefers referrals, the user should set **chainingProhibited**.

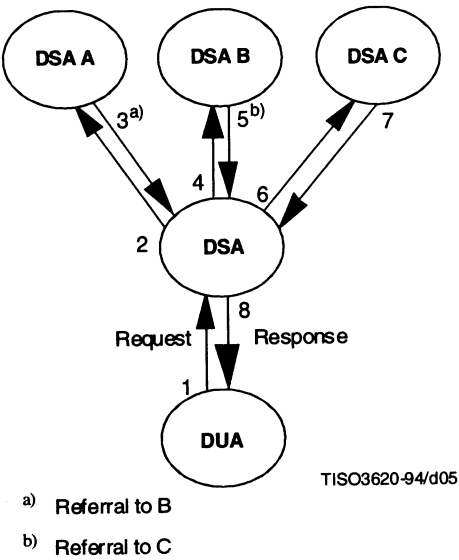


Figure 4a – Referral mode (DSA acts on referrals)

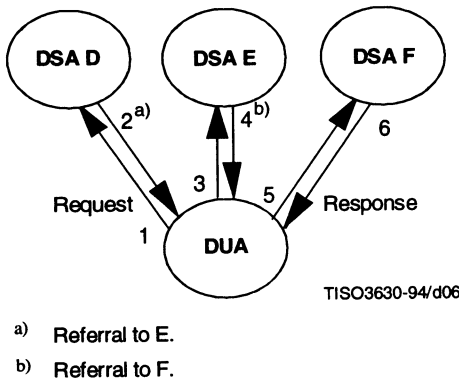


Figure 4b – Referral mode (DUA acts on referrals)

SECTION 4 – DSA ABSTRACT SERVICE

9 Overview of DSA Abstract Service

The service of the Directory is fully described in ITU-T Rec. X.511 | ISO/IEC 9594-3. When such a service is provided in a distributed environment, as modeled in clause 7, it can be regarded as being provided by means of a set of DSAs. This is illustrated in Figure 1.

For each operation defined in the Directory service, a corresponding “chained” operation is defined in the DSA abstract service for use between DSAs cooperating in the accomplishment of that Directory service operation. Thus, a DSA receiving a Read operation from a DUA might require the assistance of another DSA (e.g. a DSA holding the target entry or a copy of it) to satisfy it, and so send that DSA a Chained Read operation.

The information types exchanged in the DSA abstract service are defined in clause 10. The operations and errors of the DSA abstract service are defined in clauses 11 through 13.

10 Information types

10.1 Introduction

This clause identifies, and in some cases defines, a number of information types which are subsequently used in the definition of various of the operations of the DSA abstract service. 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 to merit being defined separately from the operation which uses them.

Several of the information types used in the definition of the DSA abstract service are actually defined elsewhere. Subclause 10.2 identifies these types and indicates the source of their definition. Subclauses 10.3 through 10.9 identify and define an information type.

10.2 Information types defined elsewhere

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

- **aliasedEntryName;**
- **DistinguishedName;**
- **Name;**
- **RelativeDistinguishedName.**

The following information types are defined in ITU-T Rec. X.511 | ISO/IEC 9594-3:

(Bind)

- **DirectoryBind**

(Operations)

- **Abandon**

(Errors)

- **abandoned;**
- **attributeError;**
- **nameError;**
- **securityError;**
- **serviceError;**
- **updateError.**

(Information Object Class)

- **OPTIONALLY-SIGNED**

(Data Type)

- **SecurityParameters**

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

- **PresentationAddress.**

10.3 Chaining Arguments

The **ChainingArguments** are present in each chained operation, to convey to a DSA the information needed to successfully perform its part of the overall task:

ChainingArguments	::=	SET {
originator	[0]	DistinguishedName OPTIONAL,
targetObject	[1]	DistinguishedName OPTIONAL,
operationProgress	[2]	OperationProgress DEFAULT { nameResolutionPhase notStarted },
traceInformation	[3]	TraceInformation,
aliasDereferenced	[4]	BOOLEAN DEFAULT FALSE,
aliasedRDNs	[5]	INTEGER OPTIONAL, <i>-- absent unless aliasDereferenced is TRUE</i>
returnCrossRefs	[6]	BOOLEAN DEFAULT FALSE,
referenceType	[7]	ReferenceType DEFAULT superior,
info	[8]	DomainInfo OPTIONAL,
timeLimit	[9]	UTCTime OPTIONAL,
securityParameters	[10]	SecurityParameters DEFAULT { },
entryOnly	[11]	BOOLEAN DEFAULT FALSE,
uniqueIdentifier	[12]	UniqueIdentifier OPTIONAL,
authenticationLevel	[13]	AuthenticationLevel OPTIONAL,
exclusions	[14]	Exclusions OPTIONAL,
excludeShadows	[15]	BOOLEAN DEFAULT FALSE,
nameResolveOnMaster	[16]	BOOLEAN DEFAULT FALSE }

The various components have the meanings as defined below:

- a) The **originator** component conveys the name of the (ultimate) originator of the request unless already specified in the security parameters. If **requester** is present in **CommonArguments**, this argument may be omitted.
- b) The **targetObject** component conveys the name of the object whose directory entry is being routed to. The role of this object depends on the particular operation concerned: it may be the object whose entry is to be operated on, or which is to be the base object for a request or sub request involving multiple objects (e.g. **ChainedList** or **ChainedSearch**). This component can be omitted only if it has the same value as the object or base object parameter in the chained operation, in which case its implied value is that value.
- c) The **operationProgress** component is used to inform the DSA of the progress of the operation, and hence of the role which it is expected to play in its overall performance. The information conveyed in this component is specified in 10.5.
- d) The **traceInformation** component is used to prevent looping among DSAs when chaining is in operation. A DSA adds a new element to trace information prior to chaining an operation to another DSA. On being requested to perform an operation, a DSA checks, by examination of the trace information, that the operation has not formed a loop. The information conveyed in this component is specified in 10.6.
- e) The **aliasDereferenced** component is a **BOOLEAN** value which is used to indicate whether or not one or more alias entries have so far been encountered and dereferenced during the course of distributed name resolution. The default value of **FALSE** indicates that no alias entry has been dereferenced.
- f) The **aliasedRDNs** component indicates how many of the RDNs in the **targetObject Name** have been generated from the **aliasedEntryName** attributes of one (or more) alias entries. The integer value is set whenever an alias entry is encountered and dereferenced. This component shall be present if and only if the **aliasDereferenced** component is **TRUE**.
- g) The **entryOnly** component is set to **TRUE** if the original operation was a search, with the subset argument set to **oneLevel** and an alias entry was encountered as an immediate subordinate of the **baseObject**. The DSA which successfully performs name resolution on the **targetObject** name, shall perform object evaluation on only the named entry.

- h) The **returnCrossRefs** component is a Boolean value which indicates whether or not knowledge references, used during the course of performing a distributed operation, are requested to be passed back to the initial DSA as cross references, along with a result or referral. The default value of **FALSE** indicates that such knowledge references are not to be returned.
- i) The **referenceType** component indicates, to the DSA being asked to perform the operation, what type of knowledge was used to route the request to it. The DSA may therefore be able to detect errors in the knowledge held by the invoker. If such an error is detected it shall be indicated by a **ServiceError** with the **invalidReference** problem. **ReferenceType** is described fully in 10.7.

NOTE – If the **referenceType** is missing then the value superior shall be assumed.

- j) The **info** component is used to convey DMD-specific information among DSAs which are involved in the processing of a common request. This component is of type **DomainInfo**, which is of unrestricted type:

DomainInfo ::= **ABSTRACT-SYNTAX.&Type**

- k) The **timeLimit** component, if present, indicates the time by which the operation is to be completed (see 17.4.4b).
- l) The **SecurityParameters** component is specified in ITU-T Rec. X.511 | ISO/IEC 9594-3. Its absence is deemed equivalent to there being an empty set of security parameters.
- m) **AuthenticationLevel** is optionally supplied when it is required to indicate the manner in which authentication has been carried out. The **AuthenticationLevel** element is described in ITU-T Rec. X.501 | ISO/IEC 9594-2.
- n) **UniqueIdentifier** is optionally supplied when it is required to confirm the originator name. The **UniqueIdentifier** element is described in ITU-T Rec. X.501 | ISO/IEC 9594-2.
- o) The **entryOnly** component is set to **TRUE** if the original operation was a Search with the **subset** argument set to **oneLevel**, and an alias entry was encountered as an immediate subordinate of the **baseObject**. The DSA which successfully performs name resolution on the **targetObject** name shall perform object evaluation on only the named entry.
- p) The **exclusions** component has significance only for Search operations; it indicates, if present, which subtrees of entries subordinate to the **targetObject** shall be excluded from the result of the Search operation (see 10.9).
- q) The **excludeShadows** component has significance only for Search and List operations; it indicates that the search shall be applied to entries and not to entry copies. This optional component may be used by a DSA as one way to avoid the receipt of duplicate results (see 20.1).
- r) The **nameResolveOnMaster** component only has significance during name resolution, and is only set if NSSRs have been encountered. If set to **TRUE**, it signals that subsequent name resolution, i.e. matching the remaining RDNs from **nextRDNTToBeResolved**, shall not employ entry copy information; subsequent resolution of each remaining RDN shall be done in the master DSA for the entry identified by that RDN (see 20.1).

10.4 Chaining Results

The **ChainingResults** are present in the result of each operation and provide feedback to the DSA which invoked the operation.

ChainingResults	::=	SET {
info	[0]	DomainInfo OPTIONAL,
crossReferences	[1]	SEQUENCE OF CrossReference OPTIONAL,
securityParameters	[2]	SecurityParameters DEFAULT { },
alreadySearched	[3]	Exclusions OPTIONAL }

The various components have the meanings as defined below:

- a) The **info** component is used to convey DMD-specific information among DSAs which are involved in the processing of a common request. This component is of type **DomainInfo**, which is of unrestricted type:
- b) The **crossReferences** component is not present in the **ChainingResults** unless the **returnCrossRefs** component of the corresponding request had the value **TRUE**. This component consists of a sequence of **CrossReference** items, each of which contains a **contextPrefix** and an **accessPoint** descriptor (see 10.8).

```

CrossReference ::= SET {
    contextPrefix [0] DistinguishedName,
    accessPoint [1] AccessPointInformation }

```

A **CrossReference** may be added by a DSA when it matches part of the **targetObject** argument of an operation with one of its context prefixes. The administrative authority of a DSA may have a policy not to return such knowledge, and will in this case not add an item to the sequence.

- c) The **SecurityParameters** component is specified in ITU-T Rec. X.511 | ISO/IEC 9594-3. Its absence is deemed equivalent to there being an empty set of security parameters.
- d) The **alreadySearched** component, if present, indicates which subordinate RDNs immediately subordinate to the **targetObject** have been processed as a part of a chained Search operation and therefore shall be excluded in a subsequent subrequest.

10.5 Operation Progress

An **OperationProgress** value describes the state of progress in the performance of an operation which several DSAs shall participate in.

```

OperationProgress ::= SET {
    nameResolutionPhase [0] ENUMERATED {
        notStarted (1),
        proceeding (2),
        completed (3) },
    nextRDNTToBeResolved [1] INTEGER OPTIONAL }

```

The various components have the meanings as defined below:

- a) The **nameResolutionPhase** component indicates what phase has been reached in handling the **targetObject** name of an operation. Where this indicates that name resolution has **notStarted**, then a DSA has not hitherto been reached with a naming context containing the initial RDN(s) of the name. If name resolution is **proceeding**, then the initial part of the name has been recognized, although the DSA holding the target object has not yet been reached. The **nextRDNTToBeResolved** indicates how much of the name has already been recognized (see 10.5.b). If name resolution is **completed**, then the DSA holding the target object has been reached, and performance of the operation proper is proceeding.
- b) The **nextRDNTToBeResolved** indicates to the DSA which of the RDNs in the **targetObject** name is the next to be resolved. It takes the form of an integer in the range one to the number of RDNs in the name. This component is only present if the **nameResolutionPhase** component has the value **proceeding**.

10.6 Trace Information

A **TraceInformation** value carries forward a record of the DSAs which have been involved in the performance of an operation. It is used to detect the existence of, or avoid, loops which might arise from inconsistent knowledge or from the presence of alias loops in the DIT.

```

TraceInformation ::= SEQUENCE OF TraceItem

TraceItem ::= SET {
    dsa [0] Name,
    targetObject [1] Name OPTIONAL,
    operationProgress [2] OperationProgress }

```

Each DSA which is propagating an operation to another adds a new item to the end of the sequence of **TraceItem**. Each such **TraceItem** contains:

- a) the Name of the **dsa** which is adding the item;
- b) the **targetObject** name which the DSA adding the item received on the incoming request. This parameter is omitted if the request being chained came from a DUA (in which case its implied value is the **object** or **baseObject** in **XOperation**), or if its value is the same as the (actual or implied) **targetObject** in the **ChainingArgument** of the outgoing request;
- c) the **operationProgress** which the DSA adding the item received on the incoming request.

10.7 Reference Type

A **ReferenceType** value indicates one of the various kinds of reference defined in ITU-T Rec. X.501 | ISO/IEC 9594-2.

```
ReferenceType ::= ENUMERATED {
    superior      (1),
    subordinate    (2),
    cross          (3),
    nonSpecificSubordinate (4),
    supplier       (5),
    master         (6),
    immediateSuperior (7),
    self           (8) }
```

10.8 Access point information

There are three types of access points:

- a) An **AccessPoint** value identifies a particular point at which access to the Directory, specifically to a DSA, can occur. The access point has a **Name**, that of the DSA concerned, and a **PresentationAddress**, to be used in OSI communications to that DSA.

```
AccessPoint ::= SET {
    ae-title      [0] Name,
    address       [1] PresentationAddress,
    protocolInformation [2] SET OF ProtocolInformation OPTIONAL }
```

- b) A **MasterOrShadowAccessPoint** value identifies an access point to the Directory. The **category**, either **master** or **shadow**, of the access point is dependent upon whether it points to a naming context or commonly usable replicated area.

```
MasterOrShadowAccessPoint ::= SET {
    COMPONENTS OF AccessPoint,
    category [3] ENUMERATED {
        master      (0),
        shadow      (1) } DEFAULT master }}
```

- c) A **MasterAndShadowAccessPoints** value identifies a set of access points to the Directory, i.e. a set of related DSAs. These access points share the property that each refers to a DSA holding entry information from a common naming context (or a common set of naming contexts mastered in one DSA when the value is a value of the **nonSpecificKnowledge** attribute. A **MasterAndShadowAccessPoints** value indicates the **category** of each **AccessPoint** value it contains. The access point of the master DSA of the naming context need not be included in the set.

```
MasterAndShadowAccessPoints ::= SET OF MasterOrShadowAccessPoint
```

An **AccessPointInformation** value identifies one or more access points to the Directory.

```
AccessPointInformation ::= SET {
    COMPONENTS OF MasterOrShadowAccessPoint ,
    additionalPoints [4] SET OF MasterOrShadowAccessPoint OPTIONAL }
```

In the case of 1988 edition DSAs producing an **AccessPointInformation** value, the optional component of the set is absent. In the case of 1988 edition DSAs interpreting an **AccessPointInformation** value, any **MasterAndShadowAccessPoints** values present are ignored.

In the case of 1993 edition DSAs, the **MasterOrShadowAccessPoint** value component produced for an **AccessPointInformation** value may be of category master or shadow, as determined by the knowledge selection procedure of the DSA producing the value. It may be viewed as a suggested access point provided by the DSA generating the value to the DSA receiving it. A set of **MasterAndShadowAccessPoints** values may optionally also be produced for an **AccessPointInformation** value. This constitutes additional information which may be employed by the receiving DSA's knowledge selection procedure to determine an alternative access point.

10.9 Exclusions

As defined in 10.3, the **exclusions** component of **ChainingArguments** is used to limit the scope of a Search operation by identifying a number of entries subordinate to the target object which, together with all of their subordinates, shall not be included in the processing of a Search operation. The **exclusion** component is defined as a value of the ASN.1 type **Exclusions**.

Exclusions ::= SET OF **RDNSequence**

Each **RDNSequence** value in the **Exclusions** set should identify the context prefix of a naming context subordinate to the target object. If a DSA receives a Search request with an **RDNSequence** value that does not conform to this constraint, the DSA may ignore that value. The **RDNSequence** is relative to the target object, and is not the distinguished name of the context prefix.

Exclusions can, besides being part of a user request, be used by DSAs to minimize duplicate information returned from Search sub-requests performed in the presence of shadowed information.

Figure 5 illustrates an example of the use of **Exclusions**. In this example, a DSA holds two replicated areas, one beneath the other. One starts with context prefix X, the other with context prefix C. An entry copy at Y has three subordinate references to naming contexts, A, B and C.

If, as an example, a subtree Search is performed in this DSA, starting with a base object within naming context X, the DSA can provide information from replicated areas X and C. The information from naming contexts A and B has to be provided via the subordinate references. When performing request decomposition, **ContinuationReferences**, to be used in either **partialResults** or chaining, will specify Y as the target object and C as a single element of an **Exclusions** set.

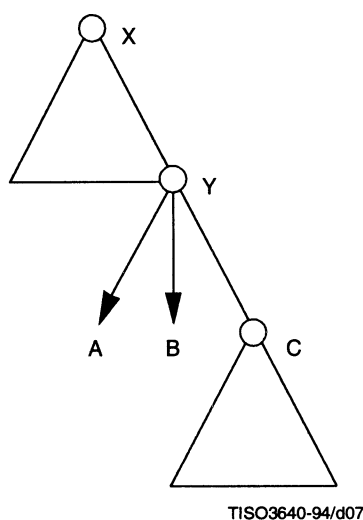


Figure 5 – Exclusions

10.10 Continuation Reference

A **ContinuationReference** describes how the performance of all or part of an operation can be continued at a different DSA or DSAs. It is typically returned as a referral when the DSA involved is unable or unwilling to propagate the request itself.

ContinuationReference	::=	SET {
targetObject	[0]	Name,
aliasedRDNs	[1]	INTEGER OPTIONAL,
operationProgress	[2]	OperationProgress,
rdnsResolved	[3]	INTEGER OPTIONAL,
referenceType	[4]	ReferenceType ,
accessPoints	[5]	SET OF AccessPointInformation,
entryOnly	[6]	BOOLEAN DEFAULT FALSE,
exclusions	[7]	Exclusions OPTIONAL,
returnToDUA	[8]	BOOLEAN DEFAULT FALSE,
nameResolveOnMaster	[9]	BOOLEAN DEFAULT FALSE}

The various components have the meanings as defined below:

- a) The **targetObject Name** indicates the name which is proposed to be used in continuing the operation. This might be different from the **targetObject Name** received on the incoming request if, for example, an alias has been dereferenced, or the base object in a search has been located.
- b) The **aliasedRDNs** component indicates how many (if any) of the RDNs in the target object name have been produced by dereferencing an alias. The argument is only present if an alias has been dereferenced.
- c) The **operationProgress** indicates the amount of name resolution which has been achieved, and which will govern the further performance of the operation by the DSAs named, should the DSA or DUA receiving the **ContinuationReference** wish to follow it up.
- d) The **rdnsResolved** component value (which need only be present if some of the RDNs in the name have not been the subject of full name resolution, but have been assumed to be correct from a cross reference) indicates how many RDNs have actually been resolved, using internal references only.
- e) The **referenceType** component indicates what type of knowledge was used in generating this continuation.
- f) The **accessPoints** component indicates the access points which are to be contacted to achieve this continuation. Only where non-specific subordinate references are involved can there be more than one **AccessPointInformation** item.
- g) The **entryOnly** component is set to **TRUE** if the original operation was a search, with the **subset** argument set to **oneLevel**, and an alias entry was encountered as an immediate subordinate of the **baseObject**. The DSA which successfully performs name resolution on the **targetObject** name, shall perform object evaluation on only the named entry.
- h) The **exclusions** component identifies a set of subordinate naming contexts that should not be explored by the receiving DSA.
- i) The **returnToDUA** element is optionally supplied when the DSA creating the continuation reference wishes to indicate that it is unwilling to return information via an intermediate DSA (e.g. for security reasons), and wishes to indicate that information may be directly available via an operation over DAP between the originating DUA and the DSA. When **returnToDUA** is set to **TRUE**, **referenceType** may be set to **self**.
- j) The **nameResolveOnMaster** element is optionally supplied when the DSA creating the continuation reference has encountered NSSRs. If set to **TRUE**, it signals that subsequent name resolution, i.e. matching the remaining RDNs from **nextRDNTToBeResolved**, shall not employ entry copy information; subsequent resolution of each remaining RDN shall be done in the master DSA for the entry identified by that RDN (see 20.1).

11 Bind and Unbind

DSABind and **DSAUnbind**, respectively, are used by a DSA at the beginning and at the end of a period of accessing another DSA.

11.1 DSA Bind

A **DSABind** operation is used to begin of a period of cooperation between two DSAs providing the Directory service.

DSABind	::=	BIND
ARGUMENT		DirectoryBindArgument
RESULT		DirectoryBindResult
BIND-ERROR		DirectoryBindError

The components of the **DSABind** are identical to their counterparts in the **DirectoryBind** (see ITU-T Rec. X.511 | ISO/IEC 9594-3 with the following differences.

- The **Credentials** of the **DirectoryBindArgument** allows information identifying the AE-Title of the initiating DSA to be sent to the responding DSA. The AE-Title shall be in the form of a Directory Distinguished Name.
- The **Credentials** of the **DirectoryBindResult** allows information identifying the AE-Title of the responding DSA to be sent to the initiating DSA. The AE-Title shall be in the form of a Distinguished Name.

11.2 DSA Unbind

A **DSAUnbind** is used to end a period of cooperation between two DSAs providing the Directory service.

DSAUnbind ::= **UNBIND**

There are no arguments, results or errors.

12 Chained operations

For each of the operations used to access the Directory abstract service, there is an operation used between cooperating DSAs in an one-to-one correspondence. The names of the operations have been chosen to reflect that correspondence by prefixing the names of operations used between cooperating DSAs with the term "Chained".

The arguments, results, and errors of the chained operations are, with one exception, formed systematically from the arguments, results, and errors of the corresponding operations in the Directory abstract service (as described in 14.3). The one exception is the **ChainedAbandon** operation, which is syntactically equivalent to its Directory service counterpart (described in 14.4).

12.1 Chained operations

A DSA, having received an operation from a DUA, may elect to construct a chained form of that operation to propagate to another DSA. A DSA, having received a chained form of an operation may also elect to chain it to another DSA. The DSA invoking a chained form of an operation may optionally sign the argument of the operation; the DSA performing the operation, if so requested, may sign the result of the operation.

The chained form of an operation is specified using the parameterized type **chained {}**.

```

chained { OPERATION : operation } OPERATION ::= {
  ARGUMENT OPTIONALLY-SIGNED { SET {
    chainedArgument          ChainingArguments,
    argument                  [0] operation.&ArgumentType }}
  RESULT OPTIONALLY-SIGNED { SET {
    chainedResult            ChainingResults,
    result                    [0] operation.&ResultType }}
  ERRORS { operation.&Errors EXCEPT (referral | dsaReferral) }
  CODE operation.&operationCode }
  NOTES

```

1 The operations of the Directory abstract service which may be used as the actual parameter of **chained {}** include the **abandoned** error. The presence of this error among the set of possible errors of a chained operation reflects the possibility discussed in 14.4, that a **ChainedAbandon** can be generated for a **ChainedModify** operation when a linked association fails.

2 The definitive specification of the DSA abstract service in Annex A applies this parameterized type to construct all the chained operations of the abstract service.

The argument of the derived operation has the components:

- chainedArgument** – This is a value of **ChainingArguments** which contains that information, over and above the original DUA-supplied argument, which is needed in order for the performing DSA to carry out the operation. This information type is defined in 10.3.
- argument** – This is a value **operation.&Argument** and consists of the original DUA-supplied argument, as specified in the appropriate clause of ITU-T Rec. X.511 | ISO/IEC 9594-3.

Should the request succeed, the result of the derived operation has the components:

- chainedResult** – This is a value of **ChainingResults** which contains that information, over and above that to be supplied to the originating DUA, which may be needed by previous DSAs in a chain. This information type is defined in 10.4.
- result** – This is a value **operation.&Result** and consists of the result which is being returned by the performer of this operation, and which is intended to be passed back in the result to the originating DUA. This information is as specified in the appropriate clause of ITU-T Rec. X.511 | ISO/IEC 9594-3.

Should the request fail, one of the errors of the set **operation.&Errors** will be returned, except that **dsaReferral** is returned instead of **referral**. The set of errors which may be reported are as described for the corresponding operation in ITU-T Rec. X.511 | ISO/IEC 9594-3. The error **dsaReferral** is described in 13.2

12.2 ChainedAbandon operation

A **ChainedAbandon** operation is used by one DSA to indicate to another that it is no longer interested in having a previously invoked distributed operation performed. This may be for any of a number of reasons, of which the following are examples:

- the operation which led to the DSA originally chaining has itself been abandoned, or has implicitly been aborted by the breakdown of an association;
- the DSA has obtained the necessary information in another way, e.g. from a faster responding DSA involved in the parallel multi-chaining.

A DSA is never obliged to issue a **ChainedAbandon**, or indeed to actually abandon an operation if requested to do so.

If **ChainedAbandon** actually succeeds in stopping the performance of an operation, then a result will be returned, and the subject operation will return an **abandoned** error. If the **ChainedAbandon** does not succeed in stopping the operation, then it itself will return an **abandonFailed** error.

13 Chained errors

13.1 Introduction

For the most part, the same errors can be returned in the DSA abstract service which can be returned in the Directory abstract service. The exceptions are that the **dsaReferral** 'error' is returned (see 13.2), instead of **Referral**, and the following service problems have the same abstract syntax but different semantics:

- a) **invalidReference** – The DSA returning this error detected an error in the calling DSA's knowledge as specified in the **referenceType** chaining argument;
- b) **loopDetected** – The DSA returning this error detected a loop in the knowledge information in the Directory.

The precedence of the errors which may occur is as for their precedence in the Directory abstract service, as specified in ITU-T Rec. X.511 | ISO/IEC 9594-3.

13.2 DSA Referral

The **dsaReferral** error is generated by a DSA when, for whatever reason, it doesn't wish to continue performing an operation by chaining the operation to one or more other DSAs. The circumstances where it may return a referral are described in 8.3.

```
dsaReferral      ERROR      ::=  {
    PARAMETER      SET {
        reference      [0]      ContinuationReference,
        contextPrefix  [1]      DistinguishedName OPTIONAL }
    CODE            id-errcode-dsaReferral }
```

The various parameters have the meanings as described below:

- a) The **ContinuationReference** contains the information needed by the invoker to propagate an appropriate further request, perhaps to another DSA. This information type is specified in 10.9.
- b) If the **returnCrossRefs** component of the **ChainingArguments** for this operation had the value **TRUE**, and the referral is being based upon a subordinate or cross-reference, then the **contextPrefix** parameter may optionally be included. The administrative authority of any DSA will decide which knowledge references, if any, can be returned in this manner (the others, for example, may be confidential to that DSA).

SECTION 5 – DISTRIBUTED PROCEDURES

14 Introduction

14.1 Scope and Limits

This clause specifies the procedures for distributed operation of the Directory which are performed by DSAs. Each DSA individually performs the procedures described below; the collective action of all DSAs produces the full set of services provided to users by the Directory.

14.2 Conformance

The description of DSA procedures in this section is based on the models in clauses 8 and 9 of ITU-T Rec. X.501 | ISO/IEC 9594-2 and clauses 7 and 8. The flow charts and their corresponding textual descriptions are one means of mapping a given set of external (DAP and/or DSP) inputs to a DSA into one or more external outputs (i.e. a result, error, referral or chained requests) produced by that DSA, depending on the particular DSA information tree held by that DSA.

It is probable that the Directory will be distributed across DSAs implemented according to different editions of the Directory Specifications, e.g. 1988 and 1993 editions. The DUA initiating the request will be unaware as to which edition the DSA or DSAs satisfying the DUA's request will have been implemented. Therefore to allow operation in such a heterogeneous environment, a DSA shall be implemented according to the rules of extensibility defined in clause 7 of ITU-T Rec. 519 | ISO/IEC 9594-5.

A DSA implementation shall be functionally equivalent to the external behavior specified by these procedures described here. The algorithms used by a particular DSA implementation to derive the correct output(s) from the given inputs and DSA information tree held are not standardized.

14.2.1 Interaction between 1988 edition and 1993 edition DSAs

If the modify operations evaluate across DSA boundaries (i.e. **AddEntry** with **TargetSystem**, **Remove** or **Rename** a context prefix), then this directory Specification only specifies how two 1993 edition DSAs shall behave. The interaction between two 1988 edition DSAs, or between a 1988 edition DSA and a 1993 edition DSA, is outside the scope of the Directory Specifications. When mixed edition DSAs have a hierarchical operational binding, knowledge of each other's edition may allow a consistent error to be given to the user.

14.3 Conceptual model

The complexity of the Directory's distributed operation gives rise to a need for conceptual modeling using both narrative and pictorial descriptive techniques. However, neither the narrative nor graphic diagrams should be construed as a formal description of distributed Directory operation.

14.4 Individual and cooperative operation of DSAs

The model views DSA operation from two separate perspectives, which, taken together, provide a complete, operational picture of the Directory.

- a) **DSA-centered perspective** – In this perspective the set of procedures that support the directory is described from the viewpoint of a single DSA. This makes it possible to provide a definitive specification of each procedure and to fully account for their interrelationships and overall control structure. Clauses 16 through 22 describe the DSA procedures from a DSA-centered perspective.
- b) **operation-centered perspective** – The DSA-centered view provides complete detail but makes it difficult to understand the structure of individual operations, which may undergo processing by multiple DSAs. Consequently Clause 15 adopts a primarily operation-centered view to introduce the processing phases applicable to each.

To support the distributed operation of the directory, each DSA shall perform actions needed to realize the intent of each operation and additional actions needed to distribute that realization across multiple DSAs. Clause 15 explores the distinction between these two kinds of actions. In clauses 16 through 22 both kinds of actions are specified in detail.

14.5 Cooperative agreements between DSAs

All DSAs which are in a subordinate/superior relationship due to the naming contexts that they hold, have hierarchical and/or non-specific hierarchical operational bindings between them, depending upon the types of knowledge reference held by the subordinate DSA.

Hierarchical and non-specific hierarchical operational bindings between DSAs may be administered using the procedures of clauses 24 and 25 of this specification, or by other means (e.g. telephone).

A DSA holding entries which are within the administrative area of its superior DSA shall administer the sub-schema and shall control access to the entries as required by the administrative authority. The regulation of entries within an administrative area may be performed as defined in ITU-T Recommendation X.501 | ISO/IEC 9594-2 or may be by local mechanisms.

15 Distributed Directory behavior

15.1 Cooperative fulfillment of operations

Each DSA is equipped with procedures capable of completely fulfilling all Directory operations. In the case that a DSA contains the entire DIB all operations are, in fact completely carried out within that DSA. In the case that the DIB is distributed across multiple DSAs the completion of a typical operation is fragmented, with just a portion of that operation carried out in each of potentially many cooperating DSAs.

In the distributed environment, the typical DSA sees each operation as a transitory event: the operation is invoked by a DUA or some other DSA; the DSA carries out processing on the object and then directs it toward another DSA for further processing.

An alternative view considers the total processing experienced by an operation during its fulfillment by multiple, cooperating DSAs. This perspective reveals the common processing phases that apply to all operations.

15.2 Phases of operation processing

Every Directory operation may be thought of as comprising three distinct phases:

- a) the *Name Resolution* phase in which the name of the object on whose entry a particular operation is to be performed is used to locate the DSA which holds the entry;
- b) the *Evaluation phase* in which the operation specified by a particular directory request (e.g. **Read**) is actually performed;
- c) the *Results Merging phase* in which the results of a specified operation are returned to the requesting DUA. If a chaining mode of interaction was chosen, the Results Merging phase may involve several DSAs, each of which chained the original request or sub request (as defined in 15.3.1 Request Decomposition) to another DSA during either or both of the preceding phases.

In the case of the operations Read, Compare, List, Search, Modify Entry, ModifyDN and Remove Entry, name resolution takes place on the object name provided in the argument of the operation. In the case of Add Entry name resolution's target entry is the immediately superior entry of that provided in the argument of the operation - it can be easily derived by removing the final RDN from the name provided in the operation argument. (This is done via local argument **m** in the FindDSE procedure of 18.2.5)

An operation on a particular entry may initially be directed at any DSA in the Directory. That DSA uses its knowledge, possibly in conjunction with other DSAs, to process the operation through the three phases.

15.2.1 Name Resolution phase

Name Resolution is the process of sequentially matching each RDN in a purported Name to an arc (or vertex) of the DIT, beginning logically at the Root and progressing downwards in the DIT. However, because the DIT is distributed between arbitrarily many DSAs, each DSA may only be able to perform a fraction of the name resolution process. A given DSA performs its part of the Name Resolution process by traversing its local DSA information tree. This process is described in clause 18 and the accompanying diagrams (see Figures 9 through 12). Based on its local DSA information tree, and the knowledge information contained therein, a DSA is able to infer whether the resolution can be continued by one or more other DSAs, or whether the name is erroneous.

15.2.2 Evaluation phase

When the name resolution phase has completed, the actual operation required (e.g. **Read** or **Search**) is performed.

Operations that involve a single entry interrogation - Read and Compare - may be carried out entirely within the DSA in which the entry is located.

Operations that involves multiple entries interrogation - List and Search - need to locate subordinates of the target, which may or may not reside in the same DSA. If they do not all reside in the same DSA, operations need to be directed to the DSAs specified in the subordinate, non-specific subordinate, supplier or master references (as appropriate) to complete the evaluation process.

15.2.3 Results merging phase

The results merging phase is entered once some of the results of the evaluation phase are available.

In those cases where the operation affected only a single entry, the result of the operation can simply be returned to the requesting DUA. In those cases where the operation has affected multiple entries on multiple DSAs, results need to be combined.

The permissible responses returned to a requester after results merging include:

- a) a complete result of the operation;
- b) a result which is not complete because some parts of the DIT remain unexplored (applies to List and Search only). Such a *partial result* may include continuation references for those parts of the DIT not explored;
- c) an error (a referral being a special case);
- d) and if the requester was a DSA, a **ChainingResult**.

15.3 Managing Distributed Operations

Information is included in the argument of each operation which a DSA may be asked to perform indicating the progress of each operation as it traverses various of the DSAs of the Directory. This makes it possible for each DSA to perform the appropriate aspect of the processing required, and to record the completion of that aspect before directing the operation outward toward further DSAs.

Additional procedures are included in the DSA to physically distribute the operations and support other needs arising from their distribution.

15.3.1 Request Decomposition

Request decomposition is a process performed internally by a DSA prior to communication with one or more other DSAs. A request is decomposed into several subrequests such that each of the latter accomplishes a part of the original task. Request decomposition can be used, for example, in the search operation, after the base object has been found. After decomposition, each of the subrequests may then be uni-chained or multi-chained to other DSAs, to continue the task.

15.3.2 DSA as Request Responder

A DSA that receives a request can check the progress of that request using the **operationProgress** parameter. This will determine whether the operation is still in the name resolution phase or has reached the evaluation phase, and what portion of the operation the DSA should attempt to satisfy. If the DSA cannot fully satisfy the request it shall either pass (by uni-chaining or multi-chaining) the operation on to one or more DSAs which can help to fulfill the request, or return a referral to another DSA, or terminate the request with an error.

15.3.3 Completion of Operations

Each DSA that has initiated an operation or propagated an operation to one or more other DSAs shall keep track of that operation's existence until each of the other DSAs has returned a result or error, or the operation's maximum time limit has expired. This requirement applies to all operations, propagation modes and processing phases. It ensures the orderly closing down of distributed operations that have propagated out into the Directory.

15.4 Loop handling

The DIT may be in a state that can cause looping. As an example, looping can occur during name resolution where dereferencing one or more aliases brings the resolution back to the same branch of the DIT. Another potential cause of looping is through misconfigured knowledge references.

Within the context of a particular directory operation, a loop occurs if at any time the operation returns to a previous *state*, where *state* is defined by the following components:

- the name of the DSA currently processing the operation;
- the name of the **targetObject** as contained within the argument of the operation;
- the **operationProgress** as contained within the argument of the operation and as defined in 10.5.

This does not mean that an operation cannot be processed multiple times by a particular DSA. However, it does mean that the DSA will not process the same operation in the same state multiple times.

Looping is controlled using the **traceInformation** argument as defined in 10.6, which records the sequence of states a particular operation has gone through. Two strategies are defined to determine whether looping has occurred, or is about to occur. These are loop detection and loop avoidance, and they are described in 15.4.1 and 15.4.2 respectively.

Loop detection is mandatory and loop avoidance is optional.

15.4.1 Loop detection

On receipt of a directory operation a DSA shall initially validate the operation to ensure that it can be progressed. An important task of validation is to check for loops, by determining whether the current state of the operation appears in the sequence of previous states recorded in the **traceInformation** argument for that operation. This step of loop checking is loop detection.

15.4.2 Loop avoidance

Loop avoidance requires that a DSA, immediately prior to forwarding an operation to another DSA as part of a chaining procedure, determines whether the consequential state of the operation (which is the **traceItem** that the receiving DSA will add to **traceInformation** when it receives it) appears on the sequence of previous states recorded in the **traceInformation** argument for the original incoming operation.

In the case where referrals are received or acted upon, loop avoidance and loop detection cannot be achieved purely by examining **traceInformation**. In this case, each time a DSA acts on a referral, it needs to store the consequential state of the operation (i.e. the **traceItem** that the receiving DSA is going to add when it receives the request) along with a record of the incoming request. Before acting on or returning a referral, a DSA needs to check through this list, in order to check that an identical request has not been previously sent whilst trying to service the incoming operation.

15.5 Other considerations for distributed operation

15.5.1 Service controls

Some service controls need special consideration in the distributed environment in order that the operation is processed the way that was requested.

- a) **chainingProhibited** – A DSA consults this service control when determining the mode of propagation of an operation. If it is set then the DSA always uses referral mode. If, however, it is not set, the DSA can choose whether to use chaining or referral depending on its capabilities.
- b) **timeLimit** – A DSA needs to take account of this service control to ensure that the time limit is not exceeded in that DSA. A DSA requested to perform an operation by a DUA, initially heeds the **timeLimit** expressed by the DUA as the available elapsed time in seconds for completion of the operation. If chaining is required, the **timeLimit** is included in the chaining argument to be passed to the next DSA(s). In this case the same value of the limit is used for each chained request, and is the (UTC) time by which the operation shall complete to meet the originally specified constraint. On receiving **ChainingArguments** with a **timeLimit** specified, the receiving DSA respects this limit.
- c) **sizeLimit** – A DSA needs to take account of this service control to ensure that the list of results does not exceed the size specified. The limit, as included in the common argument of the original request, is conveyed unchanged as the request is chained. If request decomposition is required, the same value is included in the argument to be passed to the next DSA, the full limit is used for each subrequest. When the results are returned the requester DSA resolves the multiple results and applies the limit to the total to ensure that only the requested number are returned. If the limit had been exceeded, this is indicated in the reply.
- d) **priority** – In all modes of propagation, each DSA is responsible for ensuring that the processing of operations is ordered so as to support this service control if present.
- e) **localScope** – The operation is limited to a locally defined scope and each DSA shall not propagate the request outside of this.

- f) **scopeOfReferral** – If the DSA returns a referral or partial result to a **List** or **Search** operation, then the embedded **ContinuationReferences** shall be within the requested scope.

All other service controls need to be respected, but their use does not require any special consideration in the distributed environment.

15.5.2 Extensions

If a DSA encounters an extended operation in the name resolution phase of processing and determines that the operation should be chained to one or more DSAs, it shall include unchanged in the chained operation any extensions present.

NOTE – An Administrative Authority may determine that it is appropriate to return a **serviceError** with problem **unwillingToPerform** if it does not wish to propagate an extension.

If a DSA encounters an extension in the execution phase of processing, two possibilities may arise. If the extension is not critical, the DSA shall ignore the extension. If the extension is critical, the DSA shall return a **serviceError** with problem **unavailableCriticalExtension**. A critical extension to a multiple object operation may result in both results and service errors of this variety. A DSA merging such results and errors shall discard these service errors and employ the **unavailableCriticalExtension** component of **PartialOutcomeQualifier** as described in ITU-T Rec. X.511 | ISO/IEC 9594-3.

15.5.3 Alias dereferencing

Alias dereferencing is the process of creating a new target object name, by replacing the alias entry distinguished name part of the original target object name with the **AliasedEntryName** attribute value from the alias entry. The **object** name in the operation is not affected by alias dereferencing.

15.6 Authentication of Distributed Operations

Users of the Directory together with administrative authorities that provide directory services may, at their discretion, require that directory operations be authenticated. For any particular directory operation the nature of the authentication process will depend upon the security policy in force.

Two sets of authentication procedures are available which collectively enable a range of authentication requirements to be met. One set of procedures are those provided by Bind: these facilitate authentication between two directory application-entities for the purposes of establishing an association. The Bind procedures accommodate a range of authentication exchanges from a simple exchange of identities to strong authentication.

In addition to the peer entity authentication of an association as provided by Bind, additional procedures are defined within the directory to enable individual operations to be authenticated. Two distinct sets of directory authentication procedures are defined. One facilitates originator authentication services, which address the authentication, by a DSA, of the initiator of the original service request. The second set facilitate results authentication services which address the authentication, by an initiator, of any results that are returned.

For originator authentication two procedures are defined, one based upon a simple exchange of identities, termed **identity based authentication**, and one based upon digital signature techniques, termed **signature based authentication**. The former of these procedures is rudimentary in nature since the identity exchange is based upon the exchange of distinguished names which are transmitted in the clear.

For authentication of results a single **results authentication** procedure is defined, based upon digital signature techniques; due to the generally complex nature of results collation a simpler, identity-based procedure is not defined.

Authentication of error responses is **not** supported by these procedures.

The services described below are to be considered as augmenting those provided by the Bind service; Bind procedures are assumed to have been effected successfully prior to authentication of directory operations.

The procedures to be effected by a DSA in providing originator and results authentication are specified in clause 22.

16 The Operation Dispatcher

The Operation Dispatcher is the main controlling procedure in a DSA. It guides each operation through the three phases of processing a request. The Operation Dispatcher therefore makes use of a set of procedures to fully process the request as shown in Figure 6.

16.1 General Concepts

16.1.1 Procedures

Each of the procedures employed by the operation dispatcher consists of a definition of its conceptual interface in terms of its parameters i.e. arguments, results and errors, and a description of the procedure steps itself. The behavior of the procedures is described by flowcharts and text. Within a flow chart the used symbols have the following semantics (see Figure 7):

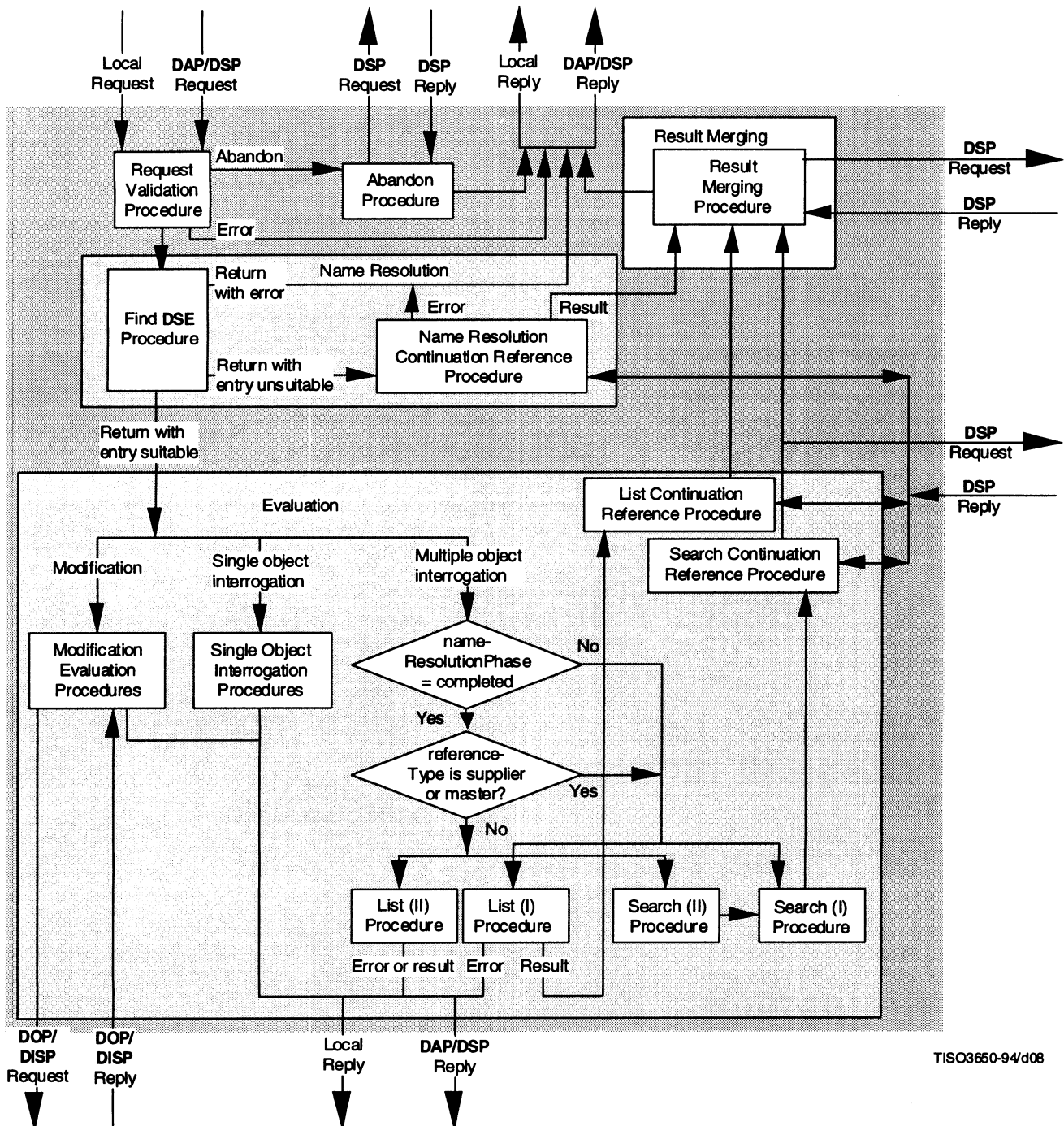


Figure 6 – Operation Dispatcher

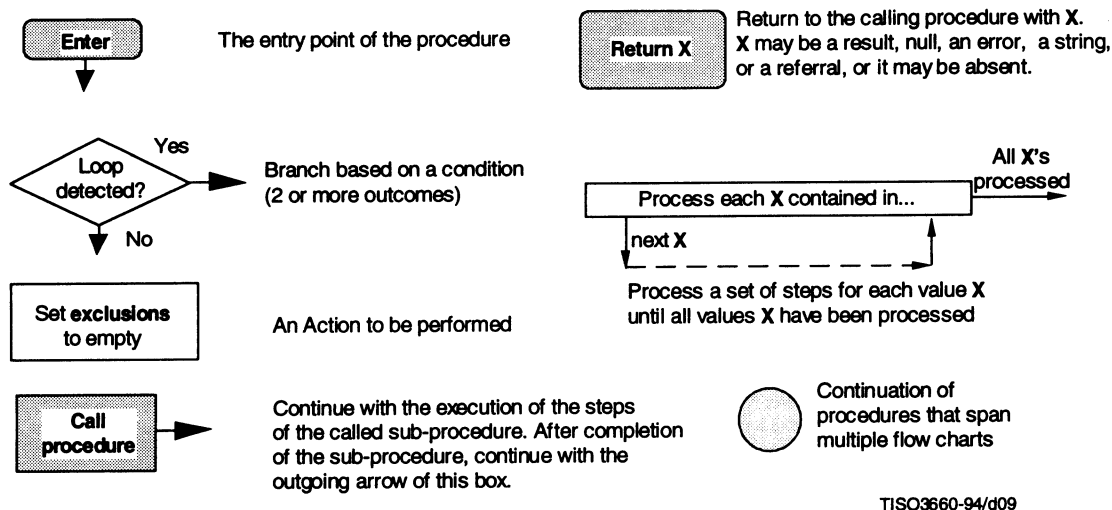


Figure 7 – Symbols Used in Flow Charts

16.1.2 Use of common data structures

All procedures make use of some data structures that are available during the processing of an operation within the operation dispatcher. These data structures serve to coordinate the data flow within the operation dispatcher. Most of these structures are directly associated with the argument of the operation and the result to be created for the operation. Components of the argument and result are referred to using their names within the associated ASN.1 definition (e.g. the **operationProgress** component of the chaining arguments). If any of these structures is a compound structure, a component of this structure may be referred to as **compound.component** (e.g. **operationProgress.nameResolutionPhase**).

The following data structures are defined within the operation dispatcher:

- **NRcontinuationList** – A list of continuation references created for use in the **Name Resolution Continuation Reference** procedure.
- **SRcontinuationList** – A list of continuation references created for use in the **List or Search Continuation Reference** procedure.
- **admPoints** – A list of references to DSEs of type administrative point that is collected during Name Resolution.

Further, a procedure may use a set of locally defined variables.

16.1.3 Errors

At each stage of the processing an error may be detected during the execution of any sub-procedure. The error identified within this sub-procedure is normally returned to the requester as a corresponding protocol error. In this case, the operation dispatcher is terminated immediately. In the case that multiple errors are received, local procedures may select one of them to be returned.

Alternatively, a procedure may choose to process errors (e.g. if **ServiceError busy** is returned to a chained search sub-request) at certain points of operation processing. In this case the procedure continues with its execution and no error is returned to the requester.

16.1.4 Asynchronous events

During the processing of an operation request within the Operation Dispatcher several asynchronous events may occur. The following paragraphs specify how to handle an exceeded time limit or size limit or administrative limit, a loss of association and an Abandon request for an operation that is being processed. The handling of all other asynchronous events, e.g. local policy decisions etc., is outside the scope of this specification.