Stable Implementation Agreements for Open Systems Interconnection Protocols: Part 11 - Directory Services Protocols

Output from the December 1990 NIST Workshop for Implementors of OSI

SIG Chair: **You Bong Weon-Yoon**
SIG Editor: **Michael Ransom**

Foreword

This part of the Stable Implementation Agreements was prepared by the Directory Services Special Interest Group (DSSIG)of the National Institute of Standards and Technology (NIST) Workshop for Implementors of Open systems Interconnection (OSI). See Procedures Manual for Workshop charter.

Text in this part has been approved by the Plenary of the above mentioned Workshop. This part replaces the previously existing chapter on Directory Services Protocol. There is no significant technical change from this text as previously given.

Future changes and additions to this version of these Implementor Agreements will be published as change pages. Deleted and replaced text will be shown as strikeout. New and replacement text will be shown as shaded.

Table of Contents

Part 11 - Directory Services Protocols 1

0 Introduction 1

1 Scope 2

2 Normative references 2

3 Status 3

4 Use of the Directory 3

5 Directory ASEs and Application Contexts 4

6 Schema 4
6.1 Support of Structures and Naming Rules 4
6.2 Support of Object Classes and Subclasses 5
6.3 Support of Attribute Types 5
6.4 Support of Attribute Syntaxes 5
6.5 Naming Contexts 5
6.6 Common Profiles 6
6.6.1 OIW Directory Common Application Directory Profile 6
6.6.1.1 Standard Application Specific Attributes and Attribute Sets 6
6.6.1.2 Standard Application Specific Object Classes 6
6.6.2 OIW Directory Strong Authentication Directory Profile 7
6.6.2.1 Other Profiles Supported 7
6.6.2.2 Standard Application Specific Object Classes 7
6.7 Restrictions on Object Class Definitions 7

7 7 Pragmatic Constraints 8
7.1 General Constraints 8
7.1.1 Character Sets 8
7.1.2 APDU Size Considerations 8
7.1.3 Service Control (SC) Considerations 8
7.1.4 Priority Service Control 8
7.2 Constraints on Operations 9
7.2.1 Filters 9
7.2.2 Errors 9
7.2.3 Error Reporting – Detection of Search Loop 9
7.3 Constraints Relevant to Specific Attribute Types 10

8 8 Conformance 10
8.1 DUA Conformance 10
8.2 DSA Conformance 11
8.3 DSA Conformance Classes 11

8.3.1 Conformance Class 0 – Centralized DSA 12
8.3.2 Conformance Class 1 – Distributed DSA 12
8.4 Authentication Conformance 12
8.5 Directory Service Conformance 13
8.5.1 Service Conformance 13
8.5.1.1 r: required 13
8.5.1.2 n: not required 14
8.5.2 Protocol Conformance 14
8.5.2.1 M: mandatory 14
8.5.2.2 G: generate 14
8.5.2.3 S: support 14
8.5.2.4 O: optional 14
8.6 The Directory Access Profile 15
8.7 The Directory System Profile 15
8.8 Digital Signature Protocol Conformance Profile 16
8.9 Strong Authentication Protocol Conformance Profile 16

9 9 Distributed Operations 16
9.1 Referrals and Chaining 16
9.2 TraceInformation 16

10 Underlying Services 17
10.1 ROSE 17
10.2 Session 17
10.3 ACSE 17

11 11 Access Control 17

12 12 Test Considerations 17
12.1 Major Elements of Architecture 17
12.2 Search Operation 18

13 13 Errors 19
13.1 Permanent vs. Temporary Service Errors 19
13.2 Guidelines for Error Handling 19
13.2.1 Introduction 19
13.2.2 Symptoms 20
13.2.3 Situations 20
13.2.4 Error Actions 20
13.2.5 Reporting 20

14 Specific Authentication Schemes 21
14.1 Specific Strong Authentication Schemes 21
14.1.1 ElGamal 21
14.1.1.1 Background 21
14.1.1.2 Digital Signature 22
14.1.1.3 Verification 23
14.1.1.4 Known Constraints on Parameters 23
14.1.1.5 Note on subjectPublicKey 24

14.1.2 One–Way Hash Functions 24
14.1.2.1 SQUARE–MOD–N Algorithm 24
14.1.2.2 MD2 Algorithm 25
14.1.2.3 Study of Other One–Way Hash Functions 25
14.1.2.4 Use of One–Way Hash Functions in Forming Signatures 25
14.1.3 ASN.1 for Strong Authentication Algorithms 25
14.1.4 Note on the ENCRYPTED MACRO 27
14.2 Protected Simple Authentication 27
14.3 Simple Authentication 27

Annex A (normative)

Maintenance of Attribute Syntaxes 28
A.1 Introduction 28
A.2 General Rules 28
A.3 Checking Algorithms 28
A.3.1 distinguishedNameSyntax 28
A.3.2 integerSyntax 29
A.3.3 telephoneNumberSyntax 29
A.3.4 countryName 29
A.3.5 preferredDeliveryMethod 29
A.3.6 presentationAddress 29
A.4 Matching Algorithms 29
A.4.1 UTCTimeSyntax 30
A.4.2 distinguishedNameSyntax 30
A.4.3 caseIgnoreListSyntax 30

Annex B (informative)

Glossary 31

Annex C (informative)

Requirements for Distributed Operations 34
C.1 General Requirements 34
C.2 Protocol Support 34
C.2.1 Usage of ChainingArguments 34
C.2.2 Usage of ChainingResults 35

Annex D (informative)

Guidelines for Applications Using the Directory 36
D.1 Tutorial 36
D.1.1 Overview 36
D.1.2 Use of the Directory Schema 36
D.1.2.1 Use of Existing Object Classes 36
D.1.2.2 Kinds of Object Classes 36
D.1.2.3 Use of Unregistered Object Classes 37
D.1.2.4 Side Effects of Creating Unregistered Object Classes 38

D.2 Creation of New Object Classes 39
D.2.1 Creation of New Subclasses 39
D.2.2 Creation of New Attributes 39
D.3 DIT Structure Rules 39

Annex E (informative)

Template for an Application Specific Profile for Use of the Directory 40

Annex F (informative)

Bibliography 42

List of Figures

Figure 1 - Structure of this Implementation Agreement. 1
Figure 2 - Centralized Directory Model. 2
Figure 3 - Distributed Directory Model. 2
Figure 4 - APDU Exchange. 8
Figure 5 - Logical DSA Application Environment. 9
Figure 6 - Three Ways of Creating Two Object Classes. 38

List of Tables

Table 1 - Pragmatic Constraints for Selected Attributes 43
Table 2 - Directory Access Service Support. 46
Table 3 - DAP Protocol Support 48
Table 4 - Directory System Service Support. 59
Table 5 - DSP Protocol Support 60
Table 6 - DAP Support for Digital Signature Protocol Conformance Profile. 68
Table 7 - DSP Support for Digital Signature Protocol Conformance Profile. 69
Table 8 - DAP Support for Strong Authentication Protocol Conformance Profile. 70
Table 9 - DSP Support for Strong Authentication Protocol Conformance Profile. 71
Table 10 - Error Symptoms 72
Table 11 - Error Situations. 76
Table 12 - Notation Used to Describe Error Actions. 77
Table 13 - Error Actions 79
Table 14 - Simple Credential Fields and Protected Simple Authentication 86

Part 11 - Directory Services Protocols

0 Introduction

This is an Implementation Agreement developed by the Implementor's Workshop sponsored by the National Institute of Standards and Technology to promote the useful exchange of data between devices manufactured by different vendors. This agreement is based on and employs protocols developed in accord with the OSI Reference Model. While this agreement introduces no new protocols, it eliminates ambiguities in interpretations.

This is an Implementation Agreement for the OSI Directory based on the ISO and CCITT documents cited in clause 2 of this part(hereafter referenced as Directory Documents). This agreement is aligned with the UNOFFICIAL 'FINAL' version of the X.500 Series of Recommendations, December 1988. Where technical differences between the ISO and CCITT versions of these documents exist (e.g., Transport Requirements) the ISO versions are given precedence. Figure 1 displays the structure of this Implementation Agreement. References to corresponding CCITT documents are included for information.

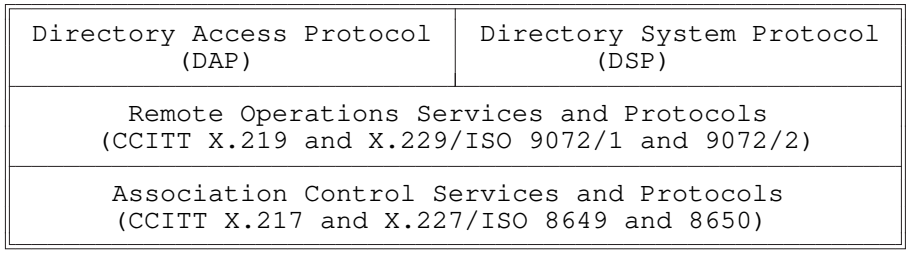

Figure 1 - Structure of this Implementation Agreement.

The Directory User Agents (DUAs) and Directory System Agents (DSAs) provide access to The Directory on behalf of humans and applications such as Message Handling and File Transfer, Access, and Management. See clause 1 for more information on the model used in the Directory.

This document covers both the Directory Access Protocol (DAP) and the Directory System Protocol(DSP) defined in the Directory Documents. A good working knowledge of the Directory Documents is assumed by this chapter. All terminology and abbreviations used but not defined in this text may be found in those documents.

1 Scope

Centralized and distributed directories can both be accommodated in this Agreement by the appropriate choice of protocols and pragmatic constraints from those specified. Figure 2 illustrates a centralized directory and figure 3 illustrates a distributed directory.

Figure 2 - Centralized Directory Model.

Figure 3 - Distributed Directory Model.

This agreement does not cover interaction between co-located entities, such as a co-resident DUA and DSA. It also does not specify the interface between a user (person or application) and a DUA.Bilateral agreements between a DUA and DSA or DSA and DSA may be implemented in addition to the requirements stated in this document. Conformance to this agreement requires the ability to interact without the use of bilateral agreements other than those required in the Directory Documents.

The logical structure of the Directory Information Base (DIB) is described in the Directory Documents. The manner in which a local portion of the DIB is organized and accessed by its DSA is not in the scope of this agreement.

2 Normative references

ISO/IEC 9594-1:1990(E),Information Technology - Open Systems Interconnection - The Directory - Part 1: Overview of Concepts, Models, and Services.

ISO/IEC 9594-2:1990(E),Information Technology - Open Systems Interconnection - The Directory - Part 2: Models.

ISO/IEC 9594-3:1990(E),Information Technology - Open Systems Interconnection - The Directory - Part 3: Abstract Service Definition.

ISO/IEC 9594-4:1990(E),Information Technology - Open Systems Interconnection - The Directory - Part 4: Procedures for Distributed Operation.

ISO/IEC 9594-5:1990(E),Information Technology - Open Systems Interconnection - The Directory - Part 5: Protocol Specifications.

ISO/IEC 9594-6:1990(E),Information Technology - Open Systems Interconnection - The Directory - Part 6: Selected Attribute Types.

ISO/IEC 9594-7:1990(E),Information Technology - Open Systems Interconnection - The Directory - Part 7: Selected Object Classes.

ISO/IEC 9594-8:1990(E),Information Technology - Open Systems Interconnection - The Directory - Part 8: Authentication Framework.

CCITT Recommendation X.500:1988,The Directory - Overview of concepts, Models and Services.

CCITT Recommendation X.501:1988,The Directory - Models.

CCITT Recommendation X.509:1988,The Directory - Authentication Framework.

CCITT Recommendation X.511:1988,The Directory - Abstract Service Definition.

CCITT Recommendation X.518:1988,The Directory - Procedures for Distributed Operations.

CCITT Recommendation X.519:1988,The Directory - Protocol Specifications.

CCITT Recommendation X.520:1988,The Directory - Selected Attribute Types.

CCITT Recommendation X.521:1988,The Directory - Selected Object Classes.

3 Status

This version was completed in December 1990.

4 Use of the Directory

Given the rapid multiplication and expansion of OSI applications, telecommunication systems and services, there is growing need for users of OSI applications, as well as the applications themselves, to communicate with each other. In order to facilitate their communications, a Directory protocol, as referenced in these agreements, has been tailored to meet their respective needs.

In one instance, The Directory will be used as a service to provide humans, in an on-line fashion, rapid and easy retrieval of information useful for determining what telecommunications services are available, and/or how to access, and address their correspondents. Further, service providers offering such a Public Directory may also use this service internally with other various telecommunications services (e.g., MHS) for the proper addressing of calls or messages. Likewise, this does not preclude the usage of these agreements to similarly generate a privately operated Directory that supports both human and application information exchanges.

In another instance, The Directory, will be used as a service by computer applications without direct human involvement. One important service is to provide Presentation Address resolution for named objects, on behalf of OSI applications. The Directory may be used by applications to search for objects (i.e., Application Entities), without direct human involvement, by the use of the ‘‘search’’ or ‘‘list’’ operations.

To support the many possible usages, The Directory is a general purpose system. It is capable of storing data of many different forms as attributes within entries, and is also capable of supporting simple or complex hierarchical structures, with variations in structure possibly occurring between one part of The Directory and another.

Compliant DSA implementations should safeguard this generality, where possible, by placing the minimum of restrictions in ‘‘hard-wired’’ form.

5 Directory ASEs and Application Contexts

This clause highlights the ASEs (Application Service Elements) and Application Contexts defined in the Directory Documents and of concern in these Agreements. The functionality of the Directory AEs (DUAs and DSAs) is defined by a set of ASEs, each Directory ASE specifying a set of Directory operations.

The interaction between these AEs is described in terms of their use of ASEs. This specific combination of a set of ASEs and the rules for their usage defines an application context.

The following ASEs are described in the Directory Documents:

a) Read ASE

b) Chained Read ASE

c) Search ASE

d) Chained Search ASE

e) Modify ASE

f) Chained Modify ASE

ROSE and ACSE also form part of the Directory Application Contexts.

The following Application Contexts are described in the Directory Document:

a) Directory Access Application Context

b) Directory System Application Context

6 Schema

There are seven (7) major topics that relate to schema:

6.1 Support of Structures and Naming Rules

DSAs shall be capable of supporting (subject to refinements laid down in these Agreements) the structure and naming rules defined in the Directory Documents, Part 7, Annex B.

Part 7, Annex B of the Directory Documents provides a framework for the basic use of the Directory in terms of the objects defined in Part 7. It does not, however, form part of the standard and,in any case, permits structures and practices which may be undesirable.The guidelines below provide tighter control within the Annex B framework.

It is recommended that only an entry subordinate to Root or Country may use a StateOrProvinceName AVA as an RDN.

6.2 Support of Object Classes and Subclasses

The DSAs shall be able to support all superclasses of the supported object classes (e.g., Top, Person).

Use of an object class in this profile or the standard (or a subclass derived from one or more of these object classes)is recommended wherever the semantics are appropriate for the application.The derivation of a new object class as an immediate subclass of Top should be avoided. For example, to represent printers in the Directory, one can derive a subclass of Device.

An entry of a particular object class may contain any optional attribute listed for it in the Directory Documents; a conformant DSA shall be able to support all these optional attributes.

In addition, a DSA may permit any locally registered attribute, or a subset of these, by providing the local extension facilities permitted by unregistered object classes (viz. Directory Documents, Part 2, clause 9.4.1 (a) and Note).

6.3 Support of Attribute Types

DSAs shall be able to support the storage and use of attribute type information, as defined in the Directory Documents, Part 6, including their use in naming and access to entries; they shall also support the definition of new attribute types, making use of pre-existing attribute syntaxes.

DSAs shall support the encoding, decoding, and matching of all the attributes in the Naming Prefixes of every naming context they hold (ref Directory Documents, Part 4, clause 9). These attributes may include attributes that are not permitted to appear in entries in those naming contexts.

6.4 Support of Attribute Syntaxes

Suggested methods for the interpretation of selected Attribute Syntaxes are defined in annex A.

6.5 Naming Contexts

The root of a naming context shall not be an alias entry.

6.6 Common Profiles

This subclause identifies profiles that are commonly useful for various applications while an application-specific profile(s) is identified by the application.

6.6.1 OIW Directory Common Application Directory Profile

6.6.1.1 Standard Application Specific Attributes and Attribute Sets

The attributes and attribute sets in the Directory Document, Part 6, associated with the object classes listed below are required.

6.6.1.2 Standard Application Specific Object Classes

DSAs shall be able to support storage and use of the object classes below, as defined in the Directory Documents, Part 7, and these object classes are expected to be useful for a range of applications.

The following object classes are mandated by the standard:

a) Top

b) DSA

c) Alias

The following object classes are expected to be generally useful in the creation of the upper portion of the DIT:

a) Country

b) Locality

c) Application Process

d) Organization

e) OrganizationalUnit

The following object classes are expected to be generally useful in the creation of DIT leaf entries:

a) Alias

b) ApplicationProcess

c) ApplicationEntity

d) DSA

e) Device

f) Group of Names

g) OrganizationalPerson

h) OrganizationalRole

i) ResidentialPerson

6.6.2 OIW Directory Strong Authentication Directory Profile

6.6.2.1 Other Profiles Supported

This profile is used in conjunction with the OIW Directory Common Application Directory Profile.

6.6.2.2 Standard Application Specific Object Classes

The following object classes are expected to be generally useful for applications to support strong authentication:

a) Strong Authentication User

b) Certification Authority

6.7 Restrictions on Object Class Definitions

An object class may not be defined as a subclass of itself, as the chain of superclasses of such an object class would be a closed loop, isolated from all other object classes, specifically Top. Such isolation is clearly illegal.

7 7 Pragmatic Constraints

This clause describes pragmatic constraints to which a conformant implementation shall adhere in addition to those specified in the Directory Documents. The pragmatic constraints can be divided into two major areas. The first includes those aspects of pragmatic constraints which apply to scope of service (see 7.1 and 7.2). The second includes those aspects of pragmatic constraints which are specific to particular attribute types (see 7.3).

7.1 General Constraints

7.1.1 Character Sets

It is a requirement to support all character sets and other name forms defined in the Directory Documents, Part 6. Those character sets include:

a) T.61

b) PrintableString

c) NumericString

7.1.2 APDU Size Considerations

In the process of chaining requests it is possible that a chaining DSA may receive, invoke or return APDUs that exceed its capacity. It is a minimum requirement that invoke APDUs and return result APDUs shall be accepted unless they exceed 32767 octets in size;in this case they may be discarded as illustrated in the right side of figure 4 (page ?), and an ‘‘unwillingToPerform’’ error reporting service shall be used.

Figure 4 - APDU Exchange.

7.1.3 Service Control (SC) Considerations

This agreement recognizes that DUAs may automatically supply defaults for any SC parameter. The choice of default values selected (if any) is seen to be a matter of local policy and consumer needs.

7.1.4 Priority Service Control

Priority is specified as a service control argument in the Directory Documents. The following statements represent a clarification of the semantics that may be used by a DSA in interpreting and operating on this parameter.

The logical model in figure 5 may be considered as an example by DSAs that implement this Service

Control. In figure 5, note that:

a) the DSA maintains three logical queues corresponding to the three priority levels;

b) the DSA Scheduler is separate and distinct from any scheduling function provided by the underlying operating system or control program services;

c) the DSA Scheduler presents jobs to the Underlying Operating Services for execution and always presents jobs of a higher priority before those of a lower priority;

d) the DSA Scheduler will not preempt a request once it has been passed to the underlying operating system service.

Figure 5 - Logical DSA Application Environment.

7.2 Constraints on Operations

There are no overall constraints upon service arguments or results except those implied in 7.1.2 of this document.

7.2.1 Filters

It is required that DSAs, at a minimum, support 8 nested ‘‘Filter’’ parameters, and a total limit of 32 Filter Items.If these limits are exceeded, the recipient of that Search Argument may return the Service Problem ‘‘unwillingToPerform’’.

7.2.2 Errors

There are no constraints upon any Error service except the APDU size limit as defined in 7.1.2.

7.2.3 Error Reporting - Detection of Search Loop

A search operation may encounter a looping situation when the search encompasses ‘‘whole-subtree’’, and an alias is encountered which is a superior to some other subtree that has been encountered during the search.

DSAs should be able to detect this situation. One possible method is by:

a) Maintaining a list of the base objects of searches initiated as a consequence of Step 5 of Part 4, clause 18.7.2.2.1 of the Directory Documents (this may require an analysis of the TraceInformation field).

b) Determining whether a new base object is superior to any base object on this list.

A new base object which would cause a loop in this way should be discarded (i.e., should not cause a new search), but no error should be reported by an error-reporting service. The circumstances should be logged so that it may be reported to an appropriate Administrative Authority for rectification.

7.3 Constraints Relevant to Specific Attribute Types

Table 1 gives pragmatic constraints associated with selected attribute types specified in the Directory Documents; many of these constraints also appear and are the same in the CCITT version of the Directory Documents. Each constraint in table 1 is given in terms of a length constraint. The length constraint for a given attribute value is the number of units which a sending entity shall not exceed and which a receiving entity shall accept and process. A sending entity need not be capable of sending attribute values as large as the length constraints.

Note that in table 1 the length constraint for strings is expressed as the number of allowable characters.

In addition to the constraints given in table 1, the following constraints apply to alphabets and integer values.

a) Alphabets: T.61 Strings used as attribute values shall only encode graphic characters and spaces. They shall not contain formatting characters (such as subscript) or other control characters.

b) Integer Values: DSAs shall be required to ‘‘pass through’’ encoded integer attribute values of arbitrary length (e.g., when chaining a Directory operation). No Directory component (i.e., DUA or DSA) shall be deemed non-conformant if it encodes integer attribute values of arbitrary length.

Components of the Directory are required to support (for storage and processing), as a minimum, integer attribute values encoded in 4 octets.

8 8 Conformance

The following subclauses will describe various aspects of Directory conformance. It should be noted that conformance to the various ASEs and conformance to the Authentication Framework are viewed as separate issues and are presented in that context.

8.1 DUA Conformance

Conformance requirements for DUAs are adequately specified in the Directory Documents, Part 5, clause 9.1 and the Directory Access Profile (see 8.6). It should be noted that the DUA conformance is based on DAP Protocol and not the User Interface. Not all options available in the standard need to be made available to the user of the DUA.

It is recognized that DUAs will be widely differing in nature:

a) Some are intended to support human users, some application users

b) Particular DUAs may not support particular operations because the application that they support has no requirement; others will be general purpose, and will support all operations.

c) Some DUAs will have a fixed view of the Directory content and structure, reflecting the usage of The Directory by a particular application; others will have a more flexible view which can be adapted to new usages.

d) Some DUAs will provide automatic referral services with automatic establishment and release of associations; others will place the burden on the user.

e) Some DUAs will provide a variety of authentication means; others will support no authentication

f) Some DUAs will handle operations synchronously; others will have the capability of maintaining several identifiable dialogues with The Directory at one time.

In the next subclause, different types of DSAs are discussed. The DUA is independent of the type of DSA it is communicating with and does not need to know what type of DSA it is communicating with.

8.2 DSA Conformance

Basic conformance requirements for a DSA are defined in the Directory Documents, Part 5, clause 9.2. Some of the terms used to describe DSA conformance are summarized below.

a) *Centralized*: A centralized DSA is defined as one that contains its entire relevant DIT; it follows that it will not make use of the DSP or generate referral responses. Since this model only contains a single DSA it is not subject to DSA interworking issues and will always provide a consistent level of service and results. A centralized DSA shall be fully‘‘protocol’’ conformant to the DAP.

b) *Cooperating*: In a distributed directory, responsibility for various portions of the DIT may be ‘‘distributed’’ among multiple DSAs. On a per operation basis we define a DSA to be holding when it is responsible for the fragment of the DIB in which a given entry will appear if it exists; we define a DSA to be propagating when it is unable to complete the name resolution process.

All DSAs shall be capable of acting as a holder and a propagator.

8.3 DSA Conformance Classes

A DSA implementation shall satisfy the conformance requirements as defined in the Directory Documents, Part 5, subclause 9.2,and shall support the ‘‘Versions’’ argument of ‘‘Bind’’.

Per the conformance clause of the Directory Documents, a DSA shall conform to the abstract syntax of the attribute types for which conformance is claimed. These attribute types shall include those required by 6.3 of this Implementor’s Agreement.

Additionally, an implementation conformant to these agreements shall state which of the following conformance classes it implements:

8.3.1 Conformance Class 0 - Centralized DSA

A DSA conformant to this class only supports the DirectoryAccessAC.

As the performance of Search and List operations can consume significant resources, the policies of some centralized DSAs may be such that these operations will not be performed. For these cases, the reply to requests for such operations would be a Service Error with the ''unwillingToPerform'' Service Problem.

8.3.2 Conformance Class 1 - Distributed DSA

A DSA implementation conformant to this class shall implement all the operations in the ASEs that are part of the Application context for which it claims conformance. It shall support the DirectoryAccessAC and it may optionally support the DirectorySystemAC.

DSAs conformant to these Agreements shall support the OIW Directory Common Application Directory Profile. In addition, DSAs may optionally conform to the OIW Directory Strong Authentication Directory Profile. Future versions of these Agreements may allow additional possibilities for minimal profile conformance.

8.4 Authentication Conformance

A Directory System may choose to implement various levels of authentication (Directory Documents, Part 8). We define the following levels of authentication in the DS:

a) No authentication at all; (**None**)

b) **Simple Uncorroborated**: identification without verification

c) **Simple Uncorroborated** authentication with verification: verified identification without a password.

d) **Simple Corroborated** authentication: verified identification with a password; intended to make masquerading difficult.

e) **Strong** authentication: identification with verification using cryptographic techniques intended to make masquerading, in practical terms, nearly impossible.

The ''Authentication Framework'' document describes the specific goal of each authentication level; listed below are several practical uses of the various levels.[1]

Simple Uncorroborated authentication may be desired to maintain access statistics or in a private network where the initiator is implicitly trusted and there is no need to incur the additional overhead of more sophisticated authentication methods.

---

[1]It is the case that some DSAs containing public information may not require authentication.

Simple Corroborated authentication may be necessary in situations where strong authentication is not practical, (i.e., international connection, no knowledge of algorithms in use, etc).

Strong authentication will be required for secure environments.

A DSA that implements Simple Corroborated authentication will check the user password by means of a compare operation on the user's entry. If no user password is supplied (Simple Uncorroborated authentication) the DSA will validate the presence of the entry for the user, by a read operation or otherwise. The authentication will fail if the password is incorrect or if the user's entry does not exist.

A DSA that implements Simple Uncorroborated authentication without verification will accept simple credentials without validating them.

Implementations claiming conformance shall, as a minimum, implement None and Simple Uncorroborated authentication without verification.

8.5 Directory Service Conformance

The following subclauses will describe various aspects of Directory conformance. Conformance to the Authentication Framework is viewed as a separate issue from conformance to the rest of the Directory document and is presented in that context.

Directory Profiles are broken into two subclauses. Service support specifies the level of support for operations and errors. Protocol support specifies the protocol elements required for implementations which claim conformance to specified operations.

8.5.1 Service Conformance

To specify the support for operations and errors, two classifications are used as follows.

8.5.1.1 r: required

The operation shall be implemented and the respective error shall be handled for conformance to these agreements.

For DUAs, *required* means:

a) or ARGUMENT parameters, create the DAP protocol elements to convey the service request to the DSA.

b) or RESULT and ERROR parameters, accept the DAP protocol elements.

For DSAs, *required* means:

a) or ARGUMENT parameters, accept the protocol elements when received and create the protocol elements when acting as a requesting DSA.

b) or RESULT and ERROR parameters, be able to convey all possible results when responding in either the DAP or DSP protocols and when receiving results, perform additional processing as defined for cooperating DSAs.

8.5.1.2 n: not required

It is left to implementations as to whether the operation or error is implemented or not.

8.5.2 Protocol Conformance

To specify the support for protocol elements, four classifications are used as follows.

8.5.2.1 M: mandatory

Generation of element is a mandatory static conformance requirement (i.e., a conformant implementation shall be capable of generating the element).

Generation of element is a mandatory dynamic conformance requirement (i.e., the element shall be present in all instances of communication which use the element).

The terms *static conformance* and *dynamic conformance* are defined in ISO 9646-1, ''OSI Conformance Testing Methodology and Framework, Part 1: General Concepts.''

8.5.2.2 G: generate

Generation of element is a mandatory static conformance requirement.

Generation of element is a conditional dynamic conformance requirement; the condition is:

Where a DSA is a propagating DSA, it shall be capable of generating the protocol element as received in related APDUs received from other DSAs. Where the DSA is a holding DSA, it shall be capable of creating all possible values of a protocol element unless otherwise noted in the ''Comments'' line.

8.5.2.3 S: support

When receiving protocol elements, implementations of these agreements shall be capable of accepting these elements without error. Actions specified in the Directory documents and in these agreements shall be taken.

8.5.2.4 O: optional

When generating protocol elements:

a) Generation of element is an optional static conformance requirement. If the implementor claims support for the corresponding Directory capability, then the implementation shall be capable of generating the element.

b) Generation of element is an optional dynamic conformance requirement. If the implementor claims support for the corresponding Directory capability, then the element shall be present in instances of communication which use the element (except where defaults allow otherwise).

When receiving protocol elements, implementations of these agreements shall be capable of accepting these elements without error. However, actions specified in the base standard and in these agreements may be taken but are not required.

Where protocol elements are nested, the classification of the nested protocol elements is of relevance only when the immediately containing protocol element is generated. The classification of the protocol elements at the highest level is relative with respect to support of the operation.

Also note that in table 3, some rows contain two support classifications in the DSA column. In such cases, the support classification in parentheses applies to centralized DSA's only. When there is only one support classification given, it applies equally to centralized and non-centralized DSA's.

8.6 The Directory Access Profile

This agreement requires implementations of the DUA to provide access to the Directory Services as defined in the DUA column in table 2. For the services in table 2 which are supported, these agreements further require DUAs to support the protocol elements as defined in the DUA column in table 3 (parts 1 - 7).

These agreements require implementations of the DSA to support the Directory Services as defined in the DSA column in table 2 (page ?). These agreements further require DSAs to support the protocol elements as defined in the DSA column in table 3. Table 3 is listed in seven parts (page ? through page ?). Note that the requirements for a centralized DSA and a cooperating DSA are different.

8.7 The Directory System Profile

These agreements require implementations of distributed DSAs which provide DSP to support the responder role for services as defined in table 4 (page ?). Further, these agreements require DSAs to support the protocol elements as specified in table 5. Table 5 is listed in nine parts (page ? through page ?).

DSAs are required to support the requestor role for all the services as defined in table 4 if conforming to the chained mode of interaction.

8.8 Digital Signature Protocol Conformance Profile

Table 6 on page ? and table 7 on page ? provide information on the digital signature protocol conformance profile.

Note that elements in CommonArguments and CommonResults SecurityParameters that are not specified in table 6 and table 7 are covered in the Directory Service Protocol Support (table 5) and Directory Access Protocol Support (table 3).

8.9 Strong Authentication Protocol Conformance Profile

Table 8 and table 9 provide information on the strong authentication protocol conformance profile.

9 9 Distributed Operations

The following requirements apply to DSAs supporting distributed operations:

> DSAs supporting authentication (e.g., simple authentication by name and password) shall be able to invoke DSP operations to carry out authentication by reference to other DSAs. Thus all such DSAs shall support the DSP protocol. The requirement is implied by the Directory Documents.

9.1 Referrals and Chaining

It is recommended that a DSA which has chained a request act upon any referrals it receives rather than returning them to the requestor if the ‘‘preferChaining’’ service control is present.

9.2 TraceInformation

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.

Each DSA which is propagating an operation to another, adds a new item to the trace information. If the propagation of a Search operation involves the creation of a new Search (cf Directory Documents, Part 4, clause 18.7.2.2.2), the trace information shall not be re-set, but the full trace information for the overall Search operation to the point where the new Search was generated shall be included in the new Search.

10 Underlying Services

This section specifies requirements over and above those given in the Directory Documents.

10.1 ROSE

It should be noted that support of ‘‘abandon’’ implies support of operation class 2.

10.2 Session

All directory implementations are required to support Session Version 2.

10.3 ACSE

The A-ABORT service is required by association-accepting DSAs to escape unwanted associations, which, under the ROSE protocol, they cannot release. In all other cases (association-initiating DSAs and DUAs) it may be preferable (though not required) to escape associations using UNBIND rather than abort.

The aborting DUA or DSA may optionally use the user information field of the A-ABORT. Such information, however, is only meaningful for diagnostic purposes and its use is not covered by these Agreements.

11 11 Access Control

Guidelines relating to access control can be found in Annex F of the Directory Documents, Part 2.

12 12 Test Considerations

This clause outlines some items that implementors may wish to consider in terms of testing expectations; additionally, future conformance testers may wish to consider these items when developing tests.

12.1 Major Elements of Architecture

One important aspect of testing is to confirm the correct behavior of DSAs and DUAs with respect to major elements of the directory architecture.

Such major elements include:

a) Conformance Statement

b) Distinguished names (e.g., name resolution, equivalence of various forms)

c) Entries and Attributes (e.g., accessibility by operations, compliance with rules)

d) Handling of distributed operations (e.g., naming contexts and knowledge)

e) Schemas

1) Structure rules (e.g., storage and maintenance of structure and of naming rules)

2) Object classes and sub-classes (e.g., storage and extension of rules for object attributes)

3) Attribute types (e.g., storage and maintenance of syntax classes and rules for multi or single valued attributes)

4) Attribute syntax (e.g., maintenance and support for attribute value testing and matching, to specification for a defined set of attribute types)

f) Operations

1) all operations

2) correct function

3) correct result

4) correct responses

g) Aliases (e.g.,correct resolution, error responses)

h) Authentication and Access Control (e.g., limitation of modify access)

i) ROSE (e.g., correct handling of invokes, results, rejects, and invoke ids)

j) ACSE (e.g., association establishment / refusal for invalid application contexts,etc.)

12.2 Search Operation

Testing of support for filter items should be reasonable. It is not expected that DSAs will be able to handle worst case testing in this area.

13 13 Errors

This clause provides clarification of the semantics of various operation errors and implementation guidelines on their usage.

13.1 Permanent vs. Temporary Service Errors

This subclause provides some clarification regarding the usage of the Service Errors *busy*, *unavailable*, and *unwillingToPerform*.

The error *busy* is particularly transient. It is returned when one or more of The Directory's internal resources are being used to their capacity and, hence, the requested operation cannot, for the moment, be performed. The Directory should be able to recover from this type of resource depletion after a short while.

The error *unavailable* is also temporary but somewhat less transient. It indicates that The Directory (or some part of it)is currently unavailable and may continue to be unavailable for a reasonably long period of time. For example, this error is returned when a given DSA is functionally disabled, or when a specific part of the DIB is undergoing reconfiguration.

The error *unwillingToPerform* has a permanent connotation. It indicates that The Directory cannot perform the requested operation because it would require resources beyond its capacity. For example, this error may be returned by a DSA if satisfying a request would result in the generation of an APDU in excess of 32767 octets.

13.2 Guidelines for Error Handling

13.2.1 Introduction

This subclause provides a recommended mapping of error situations which may be encountered to ROSE Rejects or to the errors provided in the DAP and DSP protocols of the Directory Documents.

The Directory Documents are not adequately definitive about the handling of errors. In this document, more explicit guidelines are given.

Error situations are defined by:

a) Symptom (i.e.,the manner in which the error was detected).

b) Situation (i.e., the circumstance or phase during which the error was detected. For each possible situation, the error-handling procedure needs to be defined).

13.2.2 Symptoms

Table 10 (page ? to page ?) describes a set of symptoms; the set is not necessarily exhaustive. Each is identified by a title which is used later in describing error actions. The title used for each symptom is not intended to imply any particular usage in a particular implementation.

13.2.3 Situations

Table 11 (page ?) identifies recognized situations within which particular symptoms may give rise to distinct error actions.

13.2.4 Error Actions

Table 13 (page ? to page ?) summarizes specific error actions for each possible combination of symptom and situation. Symptoms are described in 13.2.2 and situations are described in 13.2.3.

Each entry in table 13 corresponds to the symptom in the left-most column and the situation given in the column header. Each entry may specify:

a) a specific error action. The error action is described using the notation shown in table 12.

b) a specific error action and a relevant note. The note will be indicated by a number enclosed in parentheses. The notes can be found on page ?.

c) only a relevant note.

d) a blank (which indicates the corresponding combination of symptom and situation is not meaningful in the context of these Agreements).

The entries in table 13 which specify a specific error action will do so using the notation shown in table 12 (page ?).

13.2.5 Reporting

In addition to the use of error-reporting services, DSAs should implement logging services to assist in management of the Directory. The list below describes classes of error which should be logged.Note that the list is not necessarily complete.

a) Errors indicating attempted breaches of security.

b) Errors indicating local software or hardware malfunction.

c) Errors indicating malfunction or other unacceptable behavior on the part of the invoker of an operation.

d) Errors indicating loss of chaining service by another DSA.

e) Error conditions that would be difficult to diagnose with the level of detail supplied over the protocol.

f) Aborts and other exceptional communications events.

The form and accessibility of any such logs is for further study.

14 Specific Authentication Schemes

This clause describes identified authentication algorithms. Use of algorithms in this clause is not mandatory. Use of algorithms other than those described in this clause or described in the Directory Documents is by bilateral agreement.

14.1 Specific Strong Authentication Schemes

This subclause provides information on one alternative to the RSA digital signature scheme. The alternative is identified as the ''ElGamal'' digital signature scheme. Future contributions may result in other alternatives being added to this subclause.

Implementors may choose to provide digital signature capability based on RSA, ElGamal, or some other scheme appropriate for use in the OSI Directory environment.

It should be noted that use of RSA is governed by U.S.A. patent law.

14.1.1 ElGamal

The information in this subclause includes a tutorial description of the ElGamal scheme for digital signature using the notation defined in the Directory Documents, Part 8. It is intended that much of the tutorial information provided in this subclause will be moved to the security agreements sometime in the future.

14.1.1.1 Background

The ElGamal digital signature scheme is based on earlier work done by Diffie and Hellman [DIFF76] in which it was suggested that a likely candidate for a one-way function is the *discrete exponential function*

$$f(x) \equiv \alpha^x (\mathrm{mod} p) \quad (1)$$

where x is an integer between 1 and p-1 inclusive, where p is a very large prime number, and where α is an integer such that $1 \leq \alpha \geq p$ and $\{\alpha \bmod p, \alpha^2 \bmod p, ..., \alpha^{p-1} \bmod p\}$ is equal to the set {1, 2, ..., p-1}. In algebraic terminology, such an α is called a *primitive element*. References on the topic of primitive roots and elements are [McCl79] and [PATT87].

Now, in the real number system, if $y = \alpha^x$, then by definition of the logarithm we can solve for x using $x = \log_\alpha(y)$. The same idea extends to solving eq (1) for x so that inverting $f(x)$ requires calculating *discrete*

logarithms. The reason Diffie and Hellman suspected eq (1) is one-way is that for suitable p, it is computationally difficult to invert $f(x)$. According to the current state of the art, computing discrete logs for suitable p has been found to require a number of operations roughly equivalent to

$$\exp(\sqrt{cb \ln b}) \tag{2}$$

where b is the number of bits in p, and c is estimated at $c = .69$ according to [ODLY]. This can be compared to only about $2 \log_2 p$ multiplications for discrete exponentiation. If in fact the best known algorithm for computing discrete logs is near optimal then Expression (2)is a good measure of the problem's complexity (for a properly chosen p) and the discrete exponential function has all the qualities of a one-way function as described by Diffie and Hellman.

14.1.1.2 Digital Signature

Private Key: X_s denotes the private key for user X. X_s is a randomly chosen integer which user X keeps secret.

Public Key: X_p denotes the public key for user X and is calculated using the corresponding private key such that

$$X_p \equiv \alpha^{X_s} \pmod{p} \tag{3}$$

where

a) p is a prime satisfying the requirements listed in 14.1.1.4.

b) α is a primitive element mod p.

c) Note that p and α could be used globally, but because they should be easily changeable (see 14.1.1.4 for information about why these two parameters should be easily changeable) it would probably be preferable for each user to choose his/her own p and α. If users choose their own, then p and α must be made available to the recipient for use in the signature verification process.

Signing Procedure: Suppose user A wants to sign a message intended for recipient B. The basic idea is to compute a two part signature (r, s) for the message m such that

$$\alpha^{h(m)} \equiv (A_p)^r r^s \pmod{p} \tag{4}$$

where h is a one-way hash function.

Compute the signature (r, s) as follows.

a) Choose a random number k, uniformly between 0 and p-1 such that k and p-1 have no common divisor except 1 (i.e., $\gcd(k, p-1)=1$).

b) Compute r such that

$$r \equiv \alpha^k \pmod{p} \tag{5}$$

c) Use *r* to solve for the corresponding *s* as follows.

1) rewrite eq (4) using eq (5) and the definition of the public key to get

$$\alpha^{h(m)} \equiv \alpha^{(A_s)r} \alpha^{ks} (\mathrm{mod} p) \tag{6}$$

Combining exponents, get

$$\alpha^{h(m)} \equiv \alpha^{(A_s)r+ks} (\mathrm{mod} p) \tag{7}$$

eq (7) implies that

$$h(m) \equiv (A_s)r + ks (\mathrm{mod} p-1) \tag{8}$$

Note that eq (8) has a single solution for s because k was chosen such that gcd(*k*,*p*-1)=1. See [SIER88] for supporting theorem.

2) now solve for s and get

$$s \equiv I(h(m) - (A_s)r) (\mathrm{mod} p-1) \tag{9}$$

where *I* is computed such that $k * I \equiv 1 \pmod{p-1}$.

The ElGamal signature is comparable in size to the corresponding RSA signature.

14.1.1.3 Verification

The recipient receives *Ap*, *m*, *r*, *s*, α, and *p* and computes both sides of eq (4) and then compares the results.

14.1.1.4 Known Constraints on Parameters

The following list of constraints is the result of a search of current literature and may not be complete.

a) *p* must be prime

b) *p* must be large.

Note that Expression (2) can be used to speculate on the level of security afforded by crypto systems based on the discrete log problem. Breaking the ElGamal scheme has not been proven to be equivalent to finding discrete logs, but if we assume equivalence then we can estimate how large p should be for a desired level of security.

For instance,suppose we wanted to use Expression (2) to decide how large *p* should be so that we can be reasonably sure the system cannot be broken (using the best *known* algorithm) in a practical amount of time. To be on the conservative side, we decide we want to protect against a special purpose machine that can perform 10^{15} operations per second. Specifically, we want to

know how large *p* should be so that such a machine would take at least one year to break the system.

In one year, the hypothetical machine can perform 3 x 10^{22} operations. To find the size of the desired *p*, solve the following equation for *b*.

$$\exp(\sqrt{cb}\ln b) = 3 \times 10^{22} \tag{10}$$

We get $b \approx 606$. This is the number of bits in the desired *p*. So, the magnitude of the desired *p* is about 2^{606} which is roughly 266 x 10^{180}.

Hence, to be reasonably sure of attaining the desired level of security, we find a prime number greater than 266 x 10^{180} which satisfies all the other criteria listed in this subclause. Our confidence, however, is strictly based on the assumption that breaking ElGamal is as difficult as finding discrete logs and the assumption that the best known algorithm for finding discrete logs is near optimal.

c) *p* should occasionally be changed. This requirement is discussed in [ODLY84] and is related to the discovery of new algorithms for computing discrete logarithms in *GF*(*p*).

d) *p*-1 must have at least one large prime factor. This requirement is discussed in [ODLY84] and is imposed by the Silverman-Pohlig-Hellman algorithm p which_computes discrete logarithms in *GF*(*p*) using on the order $\sqrt{r}$ operations and a comparable amount of storage, where *r* is the largest prime factor in *p*-1.

e) *p* should not be the square of any prime. A subexponential-time algorithm for computing discrete logarithms in *GF*(p^2) has been found. See [ELGA85b]for details.

14.1.1.5 Note on subjectPublicKey

The ASN.1 data element subjectPublicKey, defined as BIT STRING in Annex (G) of Directory Documents, Part 8, should be interpreted in the case of ElGamal as being of type:

SEQUENCE { INTEGER, INTEGER }

where the first integer is the Arithmetic Modulus and the second is the primitive element for the finite field. The sequence is represented by the ASN.1 Basic Encoding Rules.

Implementors should take note that the size of the integers used for these parameters is expected to exceed the pragmatic constraints specified for integers by the upper layers SIG.

14.1.2 One-Way Hash Functions

14.1.2.1 SQUARE-MOD-N Algorithm

Recent research regarding the square-mod-n one-way hash function described in Annex D of the Directory Documents, Part 8, has revealed that the function is not secure. Its use, therefore, is discouraged.

14.1.2.2 MD2 Algorithm

MD2 is a one-way hash function and is described in [RFC1115]. Implementors should note that the use of MD2 may be subject to license agreements.

14.1.2.3 Study of Other One-Way Hash Functions

The Directory SIG is studying the applicability of alternative one-way hash functions. One recent development in this area was the announcement by Ralph Merkle that 2-pass SNEFRU is broken; its use is therefore discouraged. Refer to the Working Agreements for further status on the study of one-way hash functions.

14.1.2.4 Use of One-Way Hash Functions in Forming Signatures

MD2 may be used to form digital signatures in conjunction with RSA or ElGamal.

14.1.3 ASN.1 for Strong Authentication Algorithms

This subclause defines object identifiers assigned to authentication algorithms. The definitions take the form of the ASN.1 module, ‘‘OIWAlgorithmObjectIdentifiers’’.

```
OIWAlgorithmObjectIdentifiers {iso(1) identified-organization(3)
  oiw(14) dssig(7) oIWAlgorithmObjectIdentifiers(1)}
DEFINITIONS ::=
BEGIN

EXPORTS
  md2, md2WithRSA, elGamal, md2WithElGamal;

IMPORTS
  authenticationFramework
    FROM UsefulDefinitions {joint-iso-ccitt ds(5) modules(1)
                            usefulDefinitions(0)}
  ALGORITHM
    FROM AuthenticationFramework authenticationFramework;

-- categories of object identifiers

algorithm OBJECT IDENTIFIER ::= {iso(1) identified-organization(3)
                                 oiw(14) dssig(7) algorithm(2)}

encriptionAlgorithm OBJECT IDENTIFIER ::= {algorithm 1}

hashAlgorthm OBJECT IDENTIFIER        ::= {algorithm 2}

signatureAlgorithm OBJECT IDENTIFIER  ::= {algorithm 3}

-- algorithms

md2 ALGORITHM
    PARAMETER NULL
    ::= {hashAlgorithm 1}

md2WithRsa ALGORITHM
    PARAMETER NULL
    ::= {signatureAlgorithm 1}

elGamal ALGORITHM
    PARAMETER NULL
    ::= {encryptionAlgorithm 1}

Editor's Note: Refer to the June 1990 Working Agreements for information
regarding why PARAMETER NULL is specified above for the elGamal
encryption algorithm.

md2WithElGamal ALGORITHM
    PARAMETER NULL
    ::= {signatureAlgorithm 2}

END -- of Algorithm Object Identifier Definitions
```

14.1.4 Note on the ENCRYPTED MACRO

The value associated with the ENCRYPTED MACRO, as defined in Directory Documents, part 8, clause 8.4 shall be interpreted in the case of ElGamal as being type:

```
SEQUENCE { INTEGER, INTEGER }
```

The first integer in the sequence is *r* (see eq(5), 14.1.1.2). The second integer is *s* (see eq (9), 14.1.1.2).

14.2 Protected Simple Authentication

Protecting the user's distinguished name and password provides greater degrees of security than where passwords are not protected.

The procedure for achieving this protection, referred to as protected simple authentication, is outlined in the Directory Documents, Part 8, clause 5.3. The approach by which protected identifying information may be generated is outlined in the Directory Documents,Part 8, clause 5.4. For the purpose of these agreements, *f*1 and *f*2 as specified in the Directory Documents, Part 8, clause 5.4 are identical MD2 one-way functions. The algorithms for implementation of the MD2 one-way function are described in [RFC1115] (see D.3). Note that the use of MD2 maybe subject to licensing agreement. Use of other algorithms for other one-way functions is by bilateral agreement.

User *A* generates Protected2 as specified in the Directory Documents, Part 8, clause 5.4. Authenticator2 is then conveyed to *B* in the form of Simple Credentials. Table 14 on page ? shows the relationship between SimpleCredentialfields and the elements of protected simple authentication as shown in figure 2 of the Directory Documents, Part 8.

14.3 Simple Authentication

There are two major classes of authentication supported by the Directory (i.e., simple and strong authentication). Simple authentication is based on a password being passed between the two associated entities (e.g., between a Directory User and a DUA, or between two DSAs). In the case of interaction between a Directory User and a DUA, the password is compared in some way with the password attribute in the user's entry in the Directory. In the case of interaction between two DSAs, this cannot be done since the DSA object class, as defined in the Directory Documents (Part 7, clause 6.14) does not contain a password attribute.

To facilitate simple authentication between DSAs,it is recommended that a DSA have local access to a list of one or more known DSAs, with a copy of each known DSA's password. Maintenance of that information is done through the use of bilateral agreements between DSA administrators.

Annex A (normative)

Maintenance of Attribute Syntaxes

A.1 Introduction

The attribute types defined in the Directory Documents, Part 6, and listed in table 1 (page ?) have requirements, in DSAs which support them, for underlying algorithms that:

a) check attribute values for syntactical correctness and compliance with pragmatic constraints;

b) match attribute values (comparing for equality, for matching substrings, and for relative ordering).

A.2 General Rules

A DSA may receive a legitimately encoded attribute or AVA that is unsupported by the DSA. If the DSA is not required to act on it, or to store it within an entry, it may handle it by passing it on without error. Such attributes may also be used in search filter-item definitions: in this case, no error is reported, but the filter-item shall be deemed to be undefined for all entries in the DSA. This rule applies to occurrences of attributes in both operation arguments and results.

Conversely, a DSA must return a suitable error if an operation requires it to act on or store an attribute or AVA of type unsupported by the DSA. This constraint applies even for AVAs that are contained in attributes that take names as values, since the DSA will be unable correctly to match the attribute values without this attribute information.

A.3 Checking Algorithms

The subclauses below give additional checks (beyond those directly implied by the Directory Documents) which shall be applied to attributes before they are stored in the DSA.

A.3.1 distinguishedNameSyntax

Each component AVA must be checked, unregistered attribute types comprising an error; check also that no two AVAs in the same RDN have the same attribute type.

A.3.2 integerSyntax

Local implementations may apply local limitations.

A.3.3 telephoneNumberSyntax

The value of policing further rules is for further study (this applies also to telexNumber, teletexTerminalIdentifier, facsimileTelephoneNumber, G3FacsimileNonBasicParameters, x121Address, and iSDNAddress).

A.3.4 countryName

The value must be checked for compliance with ISO 3166: 1981 (E/F). (Note that from time to time further codes may be allocated.)

A.3.5 preferredDeliveryMethod

The values of the integer elements should not be restricted.

A.3.6 presentationAddress

No further checks should be applied.

A.4 Matching Algorithms

Matching algorithms are conveniently defined in terms of a two-step process:

a) Take the checked reference value, and the value to be matched, and, if necessary, reduce them to a canonical (i.e., standard) form (normalization) appropriate to each attribute syntax.

b) Carry out the comparison in the specified way (e.g., equality, substrings or ordering) using the appropriate rules for the value - character string, integer, boolean, etc.

Note that the lexical ordering of character strings (when supported) may be subject to local rules.

IMPORTANT NOTE: The combination of normalization and comparison may be replaced, in a particular implementation, by equivalent procedures. Additional notes on normalization are given below.

A.4.1 UTCTimeSyntax

If the ‘‘seconds’’ field is absent, it shall be inserted, and set to ‘‘00’’, and the form converted to the ‘‘Z’’ form.Note. The normalization strategy does not match times where the stored form omits the seconds field, and the compared form contains it, e.g.,

8804261919Z

880426191926Z

(It might have been expected that these two forms,which coincide in time to within a few seconds, would be considered identical.)

A.4.2 distinguishedNameSyntax

For each attribute value, carry out normalization in accordance with the normalization rules defined for the type (if registered); values corresponding to unregistered attribute types are left unchanged at this stage.

A.4.3 caseIgnoreListSyntax

To facilitate matching, particularly for substrings, normalization may be considered in terms of a representation which replaces the separate ASN.1 elements by a single string with a delimiter.

Annex B (informative)

Glossary

The following abbreviations may be useful; not all are used within these agreements.

ACL Access Control List

ACSE Association Control Service Element

ADDMD Administration Directory Management Domain

AETitle Application Entity Title

APDU Application Protocol Data Unit

ASE Application Service Element

ASN.1 Abstract Syntax Notation - 1

AVA Attribute Value Assertion

BRM Basic Reference Model

CA Certification Authority

CCITT The International Telegraph and Telephone Consultative Committee

CEN Committee for European Normalization

CENELEC Committee for European Normalization Electronique

CEPT Committee of European Posts and Telephones

COS Corporation for Open Systems

DAP Directory Access Protocol

DIB Directory Information Base

DIT Directory Information Tree

DMD Directory Management Domains

DSA Directory System Agent

DSP Directory System Protocol

DUA	Directory User Agent
EWOS	European Workshop for Open Systems
FTAM	File Transfer, Access & Management
INTAP	Interoperability Technical Association for Information Processing, Japan
ISDN	Integrated Services Digital Network
ISO/IEC	International Organization for Standardization
KT	Knowledge Tree
LL	Lower layers of OSI model (layers 1-4)
MAP	Manufacturing Automation Protocol
MHS	Message Handling Systems
NIST	National Institute of Standards and Technology
NSAP	Network Services Access Point
OSI	Open Systems Interconnection
PKCS	Public Key Crypto System
POSI	Promotion for Open System Interconnection
PRDMD	Private Directory Management Domain
PSAP	Presentation Service Access Point
RDN	Relative Distinguished Name
ROSE	Remote Operations Service Element
SSAP	Session Service Access Point
SIG	Special Interest Group
SPAG	Standards Promotion & Application Group
TOP	Technical and Office Protocols
TSAP	Transport Service Access Point

UL	Upper layers of OSI model (layers 5-7)
UPU	Universal Postal Union

Annex C (informative)

Requirements for Distributed Operations

The following material is included for tutorial purposes, and does not represent material additional to the Directory Documents. It is also not intended as a complete statement of requirements (the Distributed Operations part of the Directory Documents should be referred to for a complete treatment).

C.1 General Requirements

DSAs supporting distributed operations and claiming support of chaining must fully support DSP, as defined by the Directory Documents. DSAs supporting distributed operations must always be able to accept incoming DSP associations and invocations. DSAs claiming support of chaining must support:

a) Loop detection

b) Loop avoidance

In passing on operations (when chaining or multi-casting), the original DAP-supplied invocation must be passed on without change of content. In particular, there must be no alteration in anyway of any primitive content.

The support of a facility for returning cross-references (Directory Documents, Part 4, clause 10.4.1) is optional.

To ensure that traceInformation can be analyzed properly, DSAs shall only possess names that are compliant with the recommendations of the Directory Documents, Part 7 (including Annex B).

C.2 Protocol Support

C.2.1 Usage of ChainingArguments

When using ChainingArguments[2]:

a) *originator* need not be used if requestor in CommonArguments is used;

b) *targetObject* shall not be used unless the target object differs from object/base object (if it is present, object/base object are ignored for purposes of name resolution);

c) *operationProgress*, *traceInformation*, *aliasDereferenced*, *aliasedRDNs*, *referenceType*, and *timeLimit* shall be generated, accepted, and used in accordance with the Directory Documents;

[2] In this subclause, the names of protocol elements (within ChainingArguments) are italicized.

d) *returnCrossReferences* and info may optionally be generated, and shall always be accepted.

C.2.2 Usage of ChainingResults

When using ChainingResults[3]: *crossReferences* and *info* may optionally be generated, and shall always be accepted.

[3] In this subclause, the names of protocol elements (within ChainingResults) are italicized.

Annex D (informative)

Guidelines for Applications Using the Directory

D.1 Tutorial

D.1.1 Overview

Applications may have a requirement for Directory functionality. This tutorial provides assistance to those groups intending to specify Directory usage for a specific application (e.g., Message Handling Systems).

D.1.2 Use of the Directory Schema

D.1.2.1 Use of Existing Object Classes

Applications wishing to use the Directory should have determined within a standard, Implementor's Agreements, or on a propriety basis, the relevant Directory schema for their objects. Consider the following two examples:

a) Network management applications may with to define a SMAE object class.

b) File transfer applications may with to define a File Store object class.

Groups should examine relevant standards to determine if application- specific object classes or attributes have been already defined before considering any additional definition. These object classes and attributes may be found in a variety of places including a specific application standard (e.g., [Recommendation CCITT '88 X.402 | ISO 10021-2] and the Directory Documents.). Standardized object classes and attributes should be strongly considered before additional schema elements are created.

D.1.2.2 Kinds of Object Classes

There are effectively two kinds of object classes permitted within the Directory Documents: structural and auxiliary. The terms structural and auxiliary are used here for convenience when referring to particular kinds of object classes. The terms themselves are not defined in the Directory Documents.

Structural object classes have associated DIT structure rules (which control naming). Entries of this object class type are intended to be instantiated in Directory entries. A structural object class provides information on the base mandatory and optional content of a DIT entry.

An auxiliary object class provides information to enhance the mandatory and optional contents of entries. It is always used in conjunction with a structural object class.

The object class hierarchy is formed as a result of the definition of structural object classes, and the

addition of auxiliary object classes.

For example, all object classes in the Directory Documents, Part 7, are structural except for strong Authentication User and certification Authority. These two object classes should be considered auxiliary and used in conjunction with other, structural object classes.

D.1.2.3 Use of Unregistered Object Classes

The Directory Documents, Part 2, clause 9.4.1 provides a ‘‘special’’ form of object class called ‘‘unregistered.’’ An unregistered object class is not assigned an object identifier. One of the uses for unregistered object classes is to provide a means of creating a single Directory entry which logically represents a variety of object classes.Uses for unregistered object classes include:

a) Locally adding attributes to a predefined superclass;

b) Locally making optional attribute types in a predefined superclass mandatory;

c) Creating an object class derived from multiple superclasses, without needless proliferation of registered object classes.

For example, it may be advantageous to provide an entry which represents a person who is both a MHS and a FTAM user.

Unregistered object classes may best be illustrated by example. Consider an entry which represents a collection of company entries for Fizzy Company whose users have MHS O/R addresses. Using the guidelines above, the Fizzy Company defines an unregistered object class using the structural object class organizationalPerson from the Directory Documents, Part 7, and the auxiliary object class mhs-user from the MHS standards [Recommendation X.402 j ISO 10021-2] as follows:

```
fizzyCompanyPerson ::= OBJECT-CLASS
                       SUBCLASS OF organizationalPerson, mhs-user
                       MUST CONTAIN {}
                       MAY CONTAIN  {}
```

Note that no object identifier is assigned.

Also note that since there are not MUST or MAY CONTAIN’s in the fizzyCompanyPerson Object Class, the last two lines of the object class assignment (i.e., ‘‘MUST CONTAIN MAY CONTAIN’’) are optional. As with the registered form of object classes, an unregistered object class always inherits all the attributes in any of its superclasses. There is no mechanism defined whereby a subclass may selectively inherit attributes from its superclasses.

An unregistered object class always appears as a leaf in the Object Class tree. (i.e., An unregistered object class may not be a superclass of some other object class).

Using unregistered object classes in conjunction with multiple inheritance is useful as shown by figure 6 in which three ways of creating the same two object classes are shown. Either three, four, or five registered object classes are used.

Examples (a) and (c) in figure 6 are both better ways of defining the object classes than that in example (b), even though example (c) needs to use one more registered object class than example(b). This is because the multiple inheritance technique, used in examples (a) and (c), enables a Directory User searching the Directory to easily create a filter to find all entries that contain mhs-user attributes, based on a value in the object class attribute (Each Directory entry contains a list of the object identifiers of the object classes it has inherited from, so the filter would just have to find all entries that held the object identifier value of mhs-user).

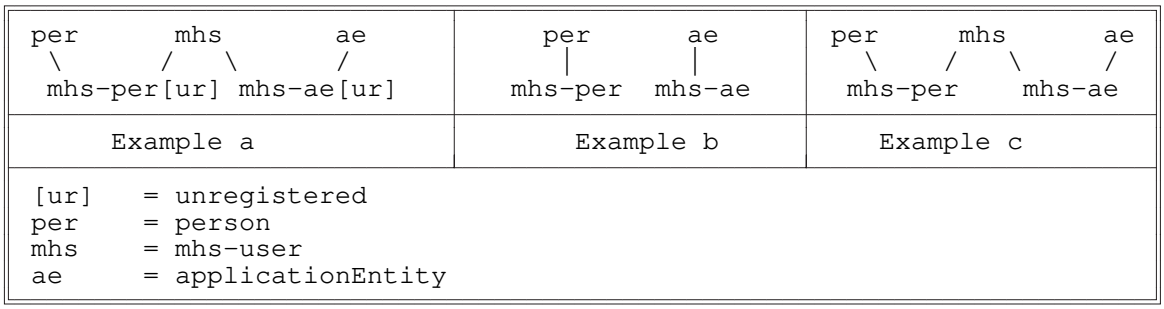

Figure 6 - Three Ways of Creating Two Object Classes.

Example (a), which uses three registered object classes, is better than example (c), which uses five, because registering the extra two object classes does not provide any advantage over not registering them, and the first method avoids needless proliferation of registered object classes.

D.1.2.4 Side Effects of Creating Unregistered Object Classes

This subclause discusses two side effects of creating unregistered object classes.

a) When an unregistered object class is defined from a single superclass, there is no means available to distinguish between the two. Within the local scope for which the unregistered class is defined, all relevant entries are considered to belong to the unregistered class.

The following is an example of this problem:

An object class of oC1(reg) has attribute type at1 mandatory and at2 optional. An unregistered form of this, oC1(unreg)is created, which makes at2 mandatory. When an Add Entry operation is received with both attributes present, the entry could belong to either form of oC1; it is indeterminate. After the entry is added a Modify Entry operation is received which requests the removal of attribute type at2. It is not clear if this operation should succeed, or whether an object class violation should be reported. If the attribute may be removed, then the entry belonged to the oC1(reg) object class and the unregistered form never existed, otherwise if the attribute may not be removed, then the entry belonged to oC1(unreg) and the registered form no longer exists.

b) More than one unregistered object class cannot be defined from the same superclass(es) for use within the same local scope, as there is no means available to distinguish the classes from one another.

D.2 Creation of New Object Classes

If no appropriate object class is available, a new object class may be defined. This should only be done if no standardized object classes and attributes can fulfill the requirements.

D.2.1 Creation of New Subclasses

Generally, an application-specific object class is defined as a subclass of a pre-existing Directory object class. These object classes are specified in the Directory Documents, Part 7. The subclass may be structural or auxiliary. Optional attributes of the superclass may be made mandatory. New attributes may also be added.

For example, MHS has used the Directory structural object class applicationEntity to derive the object class for their MHS-specific application entity MTAs.

If absolutely no relevant object class is available, an object class may be defined as a subclass of the basic object class called ‘‘Top’’.

If no appropriate object class is available, a new object class may be defined. This should only be undertaken if no standardized object class can fulfill the requirements. When defining new object classes the object-class macro, as defined in the Directory Documents, Part 2, clause 9.4.6, should be used.

If new subclasses are defined, suggested or required name forms may also be specified in text.

D.2.2 Creation of New Attributes

If no appropriate attributes are available, a new attribute type may be defined. This should only be undertaken if no standardized attributes can fulfill the requirements. When defining new attributes the attribute macro, as defined in the Directory Documents, Part 2, clause 9.5.3, should be used.

D.3 DIT Structure Rules

Applications may desire to provide guidance on DIT structure rules and naming. As with object classes, standardized or suggested structure (including naming) rules from the Directory Documents part 7, Annex B and application-specific standards should be consulted before providing new structure rules. Annex B in the Directory Documents, Part 7, provides guidelines on how to specify this information. Structure rules associated with superclasses should be adopted wherever suitable.

Annex E (informative)

Template for an Application Specific Profile for Use of the Directory

The template defined below should be used by OIW SIGs intending to specify Directory usage. Such application specific profiles shall be contained in application specific chapters of the OIW agreements. The information under each heading should be filled in (the text under each heading provides guidance on the meaning of the heading and should not be included in the profile).

a) PROFILE TITLE

Application specific profiles are named in the following way:

OIW <SIG-NAME> <DESCRIPTOR> DIRECTORY PROFILE

(e.g., OIW DIRECTORY STRONG AUTHENTICATION DIRECTORY PROFILE)

b) OTHER PROFILES SUPPORTED

Other OIW Directory profiles which are to be used by this specific application are listed here. Attributes, attribute sets, object classes and structure rules that are referenced in these profiles need not be enumerated below.

c) STANDARD APPLICATION SPECIFIC ATTRIBUTES AND ATTRIBUTE SETS

Any attributes supported from the relevant standards. For example, the MHS SIG might include mhs-or-address here.

d) STANDARD APPLICATION SPECIFIC OBJECT CLASSES

Any object classes supported from the relevant standards. For example, the MHS SIG might include mhs-user here.

e) OIW APPLICATION SPECIFIC ATTRIBUTES AND ATTRIBUTE SETS

This, optional, component of this profile allows for the specification of OIW application specific attributes and attribute sets. This section of this template should be used rarely and with consideration that no standard profile or attribute/attribute set exists which can be used.

f) OIW APPLICATION SPECIFIC OBJECT CLASSES

This, optional, component of this profile allows for the specification of OIW application specific object classes. This section of this template should be used rarely and with consideration that no standard profile or object class exists which can be used.

g) STRUCTURE RULES

Guidance for DIT structural rules, provided only when structure rules associated with superclasses

are not adopted. The Directory Documents, Part 7, Annex B provide an example and guideline to use in specifying this information.

Annex F (informative)

Bibliography

[ELGA85] ElGamal T., ‘‘A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms,’’ *IEEE Transactions on Information Theory*, vol. IT-31, No. 4,July 1985.

[DIFF76] Diffie W., Hellman M., ‘‘New Directions in Cryptography,’’ *IEEE Transactions on Information Theory*, vol. IT-22, Nov. 1976

[COPP86] Coppersmith, D., Odlyzko, A., Schroeppel, R., ‘‘Discrete Logarithms inGF (p),’’ *Algorithmica*, vol. 1, 1986.

[McCl79] McClellan, J., Rader, C., *Number Theory in Digital Signal Processing*, Prentice-Hall, 1979.

[PATT87] Patterson, W., *Mathematical Cryptology for Computer Scientists and Mathematicians*, Rowman & Littlefield, 1987.

[ODLY] Odlyzko, A., ‘‘On the Complexity of Computing Discrete Logarithms and FactoringIntegers,’’ to appear in *Fundamental Problems in Communication and Computation*, B. Gopinath and T.Loven, Eds., New York, NY: Springer.

[ODLY84] Odlyzko, A., ‘‘Discrete Logarithms in Finite Fields and Their Cryptographic Significance,’’ in *Advances in Cryptology, Proceedings of EUROCRYPT 84*. New York, NY:Springer-Verlag, pp. 224-314.

[ELGA85b] ElGamal, T., ‘‘A Subexponential-time Algorithm for Computing Discrete Logarithms over GF (p2),’’ *IEEE Transactions on Information Theory*, vol. IT-31, July 1985.

[SIER88] Sierpinski, W., *Elementary Theory of Numbers*, North-Holland 1988.

[RFC1115] Linn, J., *Privacy Enhancement for Internet Electronic Mail: Part III - Algorithms, Modes, and Identifiers*, RFC-1115, August 1989, IAB Privacy Task Force.

Table 1 - Pragmatic Constraints for Selected Attributes

Attribute Type	Content	Constraints	Primary Source	Notes
Aliased Object Name	Distinguished Name			Note 3
Business Category	T.61 or Printable String	ub-business-category 128	CCITT X.520	
Common Name	T.61 or Printable String	ub-common- name 64	CCITT X.520	
Country Name	Printable String	2	ISO 3166	
Description	T.61 or Printable String	ub-description 1024	CCITT X.520	About 1 screen full
Destination Indicator	Printable String	ub-destination-indicator 128	CCITT X.520	
Facsimile Telephone Number	Facsimile Telephone Number	ub-telephone-numb er 32	CCITT X.520	Optionally includes G3 non-basic parameters (Upper bounds ffs)
International ISDN Number	Numeric String	ub-isdn-address 16	CCITT X.520	E.164 Internat'l ISDN Number
Knowledge Information	T.61 or Printable String	1024	OIW	About 1 screen full
Locality Name	T.61 or Printable String	ub-locality-name 128	CCITT X.520	
Member	Distinguished Name			Note 3
Object Class	Object Identifier	256 octets	OIW	
Organization Name	T.61 or Printable String	ub-organization-na me 64	CCITT X.520	
Organizational Unit Name	T.61 or Printable String	ub-organizational-unit- name 64	CCITT X.520	
Owner	Distinguished Name			Note 3
Physical Delivery OfficeName	T.61 or Printable String	ub-physical-office-n ame 128	CCITT X.520	

Table 1 - Pragmatic Constraints for Selected Attributes (continued)

Attribute Type	Content	Constraints	Primary Source	Notes
Post Office Box	T.61 or Printable String	ub-post-office-box 40	CCITT X.520	
Postal Address	Postal Address	ub-postal-line6 ub-postal-string30	CCITT X.520	UPU
Postal Code	T.61 or Printable String	ub-postal-code 40	CCITT X.520	
Presentation Address	Presentation Address	224 octets	NIST	Note 2(page ?), ISO 7498.3 & X.200
Registered Address	Postal Address	ub-postal-line6 ub-postal-string30	CCITT X.520	
Role Occupant	Distinguished Name			Note 3
Search_Guide	Guide	256	OIW	
See Also	Distinguished Name			Note 3 (page ?)
Serial Number	Printable String	ub-serial-number 64	CCITT X.520	
State or Province Name	T.61 or Printable String	ub-state-name 128	CCITT X.520	
Street Address	T.61 or Printable String	ub-street-address 128	CCITT X.520	
Supported Application Context	Object Identifier	256	OIW	
Surname	T.61 or Printable String	ub-surname 64	CCITT X.520	
Telephone Number	Printable String	ub-telephone-numb er 32	CCITT X.520	E.123

Table 1 - Pragmatic Constraints for Selected Attributes (concluded)

Attribute Type	Content	Constraints	Primary Source	Notes
Teletex Terminal Identifier	Teletex Terminal Identifier	ub-teletex-terminal-id 1024	CCITT X.520	Optionally includes Teletex non-basic parameters (upper bound ffs)
Telex Number	Telex Number	ub-telex-number14 ub-country-code4 ub-answerback 8	CCITT X.520	Contains sequence of telex number, country code, and answerback
Title	T.61 or Printable String	ub-title 64	CCITT X.520	
User Password	Octet String	ub-user-password 128	CCITT X.520	Allow long passwords generated by machine
X.121 Address	Numeric String	ub-x121-address 15	CCITT X.520	X.121

NOTES

1 The pragmatic constraints of these parameters are defined in other standards. We will accommodate these values in our pragmatic constraints.

2 Presentation address is composed of "X" NSAP addresses, and three selectors, (20X + 32 + 16 + 16), e.g., if X= 1, this would be 84. These numbers are based on the most recent implementors' agreements. With 8 NSAP addresses this value is 224.

3 Pragmatic constraints are only applied to the individual components of Distinguished Name as defined in the Directory Documents, Part 2. Not all components of a DN will necessarily be understood by an implementation.

4 Implementors should be aware that constraints on Postal Address may not be sufficient for some markets.

Table 2 - Directory Access Service Support.

Operations and Errors	Support Classification		Comments
	DUA	DSA	
-- BIND and UNBIND --			
DirectoryBind	r	r	
DirectoryUnbind	r	r	
-- OPERATIONS --			
-- READ OPERATIONS--			
Read	n	r	
Compare	n	r	
Abandon	n	r (note 2)	
-- SEARCH OPERATIONS --			
List	n	r (note 1)	
Search	n	r (note 1)	
-- MODIFY OPERATIONS --			
AddEntry	n	r	
RemoveEntry	n	r	
ModifyEntry	n	r	
ModifyRDN	n	r	
-- ERRORS --			
Abandoned	(note 4)r		
AbandonedFailed	(note 4)r		
AttributeError	(note 4)r		
NameError	(note 4)r		
Referral	(note 4)	r(note 3)	

Table 2 - Directory Access Service Support. (concluded)

Operations and Errors	Support Classification		Comments
	DUA	DSA	
SecurityError	(note 4)	r	
ServiceError	(note 4)	r	
UpdateError	(note 4)	4	

NOTES

1 As performance of Search and List operations can consume significant resources, the policies of some centralized DSAs may be that such operations will not be performed. For these cases, the reply to the requests for such operations would be ServiceError with the ''unwillingToPerform'' Service Problem.

2 Reference Directory Documents, Part 3, clause 9.3.6

3 Centralized DSAs would not generate referrals.

4 See EntryInformationSelection information under Common Data Types (table 3, Part 6)

Table 3 - DAP Protocol Support

Protocol Element	Support Classification		Comments
	DUA	DSA	
- BIND and UNBIND -			
DirectoryBind			
DirectoryBindArgument	M	S	
credentials	O	S	
simple	O	S	
name	G	S	
validity	O	O	
password	G	S	
strong	O	O	See Strong Authentication Protocol Conformance Profile for requirements when strong authentication is supported.
externalProcedure	O	O	
versions	O	S	Supported value: v1988
DirectoryBindResult	S	G	
credentials	O	G	Shall be the same CHOICE as in DirectoryBindArgument.
simple	O	G	
name	S	G	
validity	O	O	
password	O	O	
strong	O	O	See Strong Authentication Protocol Conformance Profile for requirements when strong authentication is supported.
externalProcedure	O	O	
versions	S	O	Supported value: v1988

Table 3 - DAP Protocol Support (continued)

Protocol Element		Support Classification DUA	Support Classification DSA	Comments
DirectoryBindError		S	G	
versions		S	O	Supported value: v1988
ServiceProblem		S	G	Supported value: unavailable
SecurityProblem		S	G	Supported values: inappropriateAuthentication, invalidCredentials
DirectoryUnbind				The DirectoryUnbind has no arguments.
- OPERATIONS, ARGUMENTS AND RESULTS -				
- READ OPERATIONS -				
Read				
ReadArgument	M		S	
object		M	S	
selection		O	S	See note 2 on page ?.
CommonArguments		O	S	
ReadResult		S	G	
entry		S	M	
CommonResults		S	G	
Compare				
CompareArgument		M	S	
object		M	S	
purported		M	S	
CommonArguments		O	S	
CompareResult		S	G	

Table 3 - DAP Protocol Support (continued)

Protocol Element	Support Classification		Comments
	DUA	DSA	
DistinguishedName	S	G	
matched	S	M	
fromEntry	S	G	
commonResults	S	G	
Abandon			
AbandonArgument	M	S	
invokeId	M	S	
AbandonResult	S	G	
- SEARCH OPERATIONS -			
List			
ListArgument	M	S	
object	M	S	
CommonArguments	O	S	
ListResult	S	G	
listInfo	S	G	
DistinguishedName	S	G	
subordinates	S	M	
Rel.DistinguishedName	S	M	For the case where subordinates is empty set, RDN is absent.
aliasEntry	S	G	
fromEntry	S	G	
partialOutcomeQualifier	S	G	
CommonResults	S	G	
UncorrelatedListInfo	S	G(O)	

Table 3 - DAP Protocol Support (continued)

Protocol Element	Support Classification		Comments
	DUA	DSA	
ListResult	S	G	See note 1 on page ? for additional information related to the DSA support classification.
Search			
SearchArgument	M	S	
baseObject	M	S	
subset	O	S	
filter	O	S	
searchAliases	O	S	
selection	O	S	
CommonArguments	O	S	
SearchResult	S	G	
searchinfo	S	G	
DistinguishedName	S	G	
entries	S	M	
partialOutcomeQualifier	S	G	
CommonResults	S	G	
uncorrelatedSearchinfo	S	G (O)	
SearchResult	S	G	
partialOutcomeQualifier	S	G	
limitProblem	S	G	
unexplored	S	G	
unavailableCriticalExt	S	O	
- MODIFY OPERATIONS -			
AddEntry			
AddEntryArgument	M	S	

Table 3 - DAP Protocol Support (continued)

Protocol Element	Support Classification		Comments
	DUA	DSA	
object	M	S	
entry	M	S	
CommonArgument	O	S	
AddEntryResult	S	G	
RemoveEntry			
RemoveEntryArgument	M	S	
object	M	S	
CommonArguments	O	S	
RemoveEntryResult	S	G	
ModifyEntry			
ModifyEntryArgument	M	S	
object	M	S	
changes	M	S	At least one entry modification must be supported.
addAttribute	O	S	
removeAttribute	O	S	
addValues	O	S	
removeValues	O	S	
CommonArguments	O	S	
ModifyEntryResult	S	G	
ModifyRDN			
ModifyRDNArgument	M	S	
object	M	S	
newRDN	M	S	
deleteOldRDN	O	S	
CommonArguments	O	G	

Table 3 - DAP Protocol Support (continued)

Protocol Element	Support Classification		Comments
	DUA	DSA	
ModifyRDNResult	S	G	
- ERRORS AND PARAMETERS -			
Abandoned			
AbandonFailed			
problem	S	M	
operation	S	M	
AttributeError			
object	S	M	
problems	S	M	Min. 1 error(See Directory Documents, Part 3, subclause 12.4.2.2)
type	S	M	
value	S	G	
NameError			
problem	S	M	
matched	S	M	
Referral			
candidate	S	G	
SecurityError			
problem	S	M	
ServiceError			
problem	S	M	
UpdateError			
problem	S	M	

Table 3 - DAP Protocol Support (continued)

Protocol Element	Support Classification		Comments
	DUA	DSA	
ModifyRDNResult	S	G	
- ERRORS AND PARAMETERS -			
Abandoned			
AbandonFailed			
problem	S	M	
operation	S	M	
AttributeError			
object	S	M	
problems	S	M	Min. 1 error(See Directory Documents, Part 3, subclause 12.4.2.2)
type	S	M	
value	S	G	
NameError			
problem	S	M	
matched	S	M	
Referral			
candidate	S	G	
SecurityError			
problem	S	M	
ServiceError			
problem	S	M	
UpdateError			
problem	S	M	

Table 3 - DAP Protocol Support (continued)

Protocol Element	Support Classification		Comments
	DUA	DSA	
- COMMON ARGUMENTS / RESULTS -			
CommonArguments			
ServiceControls	O	S	
SecurityParameters	O	S	See subclause 8.8.
certification-path	O	S	
name	O	S	
time	O	S	
random	O	S	
target	O	S	
requestor	O	S	
OperationProgress	O	S (O)	
nameResolutionPhase	M	S	
nextRDNToBeResolved	O	S	
aliasedRDNs	O	S (O)	
extensions	O	S	
identifier	M	S	
critical	O	S	
item	M	S	
CommonResults			
SecurityParameters	O	G (O)	See subclause 8.8.
certification-path	O	G	
name	O	G	
time	O	G	
random	O	G	
target	O	G	

Table 3 - DAP Protocol Support (continued)

Protocol Element	Support Classification		Comments
	DUA	DSA	
performer	O	G (O)	
aliasDereferenced	O	G	
- COMMON DATA TYPES -			
ServiceControls			
options	O	S	
priority	O	S	
timeLimit	O	S	
sizeLimit	O	S	
scopeOfReferral	O	S	
EntryInformationSelection			
attributeTypes	O	S	
allAttributes	O	S	Must support at least one of the CHOICE.
select	O	S	
infoTypes	O	S	
EntryInformation			
DistinguishedName	S	M	
fromEntry	S	G	
SET OF CHOICE	S	G	
AttributeType	S	G	
Attribute	S	G	
Filter			Must support at least one of the CHOICE.
item	O	S	
and	O	S	
or	O	S	

Table 3 - DAP Protocol Support (continued)

Protocol Element	Support Classification		Comments
	DUA	DSA	
not	O	S	
FilterItem			
equality	O	S	
substrings	O	S	
type	M	S	
strings	M	S	
initial	O	S	Must support at least one of the CHOICE.
any	O	S	
final	O	S	
greaterOrEqual	O	S	
lessOrEqual	O	S	
present	O	S	
approximateMatch	O	S	
SecurityParameters	O	O	See subclause 8.8.
certification-path	O	S	
name	O	S	
time	O	S	
random	O	S	
target	O	S	
ContinuationReference			
targetObject	O	M	
aliasedRDNs	O	G	
OperationProgress	O	M	
nameResolutionPhase	O	M	
nextRDNToBeResolved	O	G	

Table 3 - DAP Protocol Support (concluded)

Protocol Element	Support Classification		Comments
	DUA	DSA	
rdnsResolved	O	G	
AccessPoint	O	M	
AccessPoint			
Name	O	M	
PresentationAddress	O	M	
pSelector	O	G	
sSelector	O	G	
tSelector	O	G	
nAddress	O	M	

NOTES

1 As performance of Search and List operations can consume significant resources, the policies of some centralized DSAs may be that such operations will not be performed. For these cases, the reply to the requests for such operations would be ServiceError with the ‘‘unwillingToPerform’’ Service Problem.

2 See EntryInformationSelection information under Common Data Types (table 3, part 6)

Table 4 - Directory System Service Support.

Operations and Errors	Support Classification		Comments
	Request	Response	
- BIND and UNBIND -			
DSABind	n(notes 1,2)	r	
DSAUnbind	n(notes 1,2)	r	
- OPERATIONS -			
- CHAINED READ OPERATIONS -			
ChainedRead	n(notes 1,2)r		
ChainedCompare	n(notes 1,2)r		
chainedAbandon	n(note 1) r		
- CHAINED SEARCH OPERATIONS -			
ChainedList	n (note 1) r		
ChainedSearch	n (note 1) r		
- CHAINED MODIFY OPERATIONS -			
ChainedAddEntry	n (note 1) r		
ChainedRemoveEntry	n (note 1) r		
ChainedEntry	n (note 1) r		
ChainedModifyRDN	n (note 1) r		
- ERRORS -			
Abandoned	n(note 1)	r	
Abandonfailed	n(note 1) r		
AttributeError	n(note 1) r		
NameError	n(note 1)	r	
DSARefferal	n(note 1)	r	
SecurityError	n(note 1) r		
SeviceError	n(note 1)	r	
UpdateError	n(note 1)	r	

NOTES

1 Necessary when supporting the chained mode of interaction.

2 Some of these operations may be necessary to support distributed authentication. This requirement is distinct from support for chained mode of interaction.

Table 5 - DSP Protocol Support

Protocol Element	Support Classification		Comments
	Request	Response	
- BIND and UNBIND -			
DSABind			
DirectoryBindArgument	M	S	
credentials	G	S	
simple	G	S	
name	G	S	
validity	O	O	
password	G	S	
strong	O	O	See Strong Authentication Protocol Conformance Profile for requirements when strong authentication is supported.
externalProcedure	O	O	
versions	G	S	Supported value: v1988
DSABindResult	S	G	
credentials	S	G	Shall be the same CHOICE as in DirectoryBindArgument.
simple	S	G	
name	S	G	
validity	O	O	
password	S	G	
strong	O	O	See Strong Authentication Protocol Conformance Profile for requirements when strong authentication is supported.
externalProcedure	O	O	
versions S		G	Supported value: v1988
DirectoryBindError	S	G	
versions	S	G	Supported value: v1988
ServiceProblem	S	G	Supported values: busy and unavailable.
SecurityProblem	S	G	Supported values: inappropriate Authentication, invalidCredentials.
DSAUnbind			The DSAUnbind has no arguments.

Table 5 - DSP Protocol Support (continued)

Protocol Element	Support Classification		Comments
	Request	Response	
- OPERATIONS, ARGUMENTS AND RESULTS -			
- CHAINED READ OPERATIONS -			
ChainedRead			
ChainingArgument	M	S	
ReadArgument	M	S	
object	M	S	
selection	G	S	
CommonArguments	G	S	
ChainingResult	S	M	
ReadResult	S	M	
entry	S	M	
CommonResults	S	G	
ChainedCompare			
ChainingArgument	M	S	
CompareArgument	M	S	
object	M	S	
purported	M	S	
CommonArguments	G	S	
ChainingResult	S	M	
CompareResult	S	M	
DistinguishedName	S	G	
matched	S	M	
fromEntry	S	G	
CommonResults	S	G	
ChainedAbandon			
AbandonArgument	M	S	
invokeId	M	S	
AbandonResult	S	G	
- OPERATIONS, ARGUMENTS AND RESULTS -			
- CHAINED SEARCH OPERATIONS -			
ChainedList			
ChainingArguments	M	S	

Table 5 - DSP Protocol Support (continued)

Protocol Element	Support Classification		Comments
	Request	Response	
ListArgument	M	S	
object	M	D	
CommonArguments	G	S	
ChainingResults	S	M	
ListResult	S	M	
listInfo	S	G	
DistinguishedName	S	G	
subordinates	S	M	
Rel.DistinguishedName	S	M	
aliasEntry	S	G	
fromEntry	S	G	
partialOutcomeQualifier	S	G	
CommonResults	S	G	
uncorrelatedListInfo	S	G	
ListResult	S	G	
ChainedSearch			
SearchArgument	M	S	
baseObject	M	S	
sugset	G	S	
filter	G	S	
searchAliases	G	S	
selection	G	S	
CommonArguments	G	S	
ChainingResults	S	M	
SearchResult	S	M	
Searchinfo	S	M	
DistinguishedName	S	G	
entries	S	M	
partialOutcomeQualifier	S	G	
CommonResults	S	G	
uncorrelatedSearchinfo	S	G	
SearchResult	S	G	
partialOutcomeQualifier	S	G	
limitProblem	S	G	

Table 5 - DSP Protocol Support (continued)

Protocol Element	Support Classification		Comments
	Request	Response	
unexplored	S	G	
unavailableCriticalExt	S	G	
- CHAINED MODIFY OPERATIONS -			
ChainedAddEntry			
ChainingArguments	M	S	
AddEntryArgument	M	S	
object	M	S	
entry	M	S	
CommonArguments	G	S	
ChainingResults	S	M	
AddEntryResults	S	M	
ChainedRemoveEntry			
ChainingArguments	M	S	
RemoveEntryArgument	M	S	
object	M	S	
CommonArguments	G	S	
ChainingResults	S	M	
RemoveEntryResult	S	M	
ChainedModifyEntry			
ChainingArguments	M	S	
ModifyEntryArgument	M	S	
object	M	S	
changes	M	S	
addAttribute	G	S	
removeAttribute	G	S	
addValues	G	S	
removeValues	G	S	
CommonArguments	G	S	
ChainingResults	S	M	
ModifyEntryResult	S	M	
ChainedModifyRDN			
ChainingArguments	M	S	
ModifyRDNArgument	M	S	
object	M	S	

Table 5 - DSP Protocol Support (continued)

Protocol Element	Support Classification		Comments
	Request	Response	
newRDN	M	S	
deleteOldRDN	G	S	
CommonArguments	G	S	
ChainingResults	S	M	
ModifyRDNResult	S	M	
- ERRORS and PARAMETERS -			
Abandoned			
AbandonFailed			
problem	S	M	
operation	S	M	
AttributeError			Min.1 error (see Directory Documents, part 3, subclause 12.4.2 .2)
object	S	M	
problems	S	M	
problem	S	M	
type	S	M	
value	S	G	
NameError			
problem	S	M	
matched	S	M	
DSARefferal			
ContinuationReference	S	M	
contextPrefix	S	G	
SecurityError			
problem	S	M	
ServiceError	S	G	For Directory operations
problem	S	M	
UpdateError	S	G	
problem	S	M	
- COMMON ARGUMENTS / RESULTS -			
CommonArguments			
ServiceControls	G	S	
SecurityParameters	O	S	see subclause 8.8.

Table 5 - DSP Protocol Support (continued)

Protocol Element	Support Classification		Comments
	Request	Response	
requestor	G	S	
OperationProgress	G	S	
nameResolutionPhase	M	S	
nextRDNToBeResolved	G	S	
aliasedRDNs	G	S	
extensions	G	S	
identifier	M	S	
critical	G	S	
item	M	S	
CommonResults			
SecurityParameters	S	O	See subclause 8.8.
requestor	S	G	
aliasDereferenced	S	G	
- COMMON DATA TYPES -			
ServiceControls			
options	G	S	
priority	G	S	
timeLimit	G	S	
sizeLimit	G	S	
scopeOfReferral	G	S	
EntryInformationSelection			
attributeTypes	G	S	
allAttributes	G	S	
select	G	S	
infoTypes	G	S	
EntryInformation			
DistinguishedName	S	M	
fromEntry	S	G	
SET OF CHOICE	S	G	
AttributeType	S	G	
Attribute	S	G	
Filter			
item	G	S	
and	G	S	
or	G	S	
not	G	S	

Table 5 - DSP Protocol Support (continued)

Protocol Element	Support Classification		Comments
	Request	Response	
FilterItem			
equality	G	S	
substrings	G	S	
type	G	S	
strings	G	S	
initial	G	S	
any	G	S	
final	G	S	
greaterOrEqual	G	S	
lessOrEqual	G	S	
present	G	S	
approximateMatch	G	S	
- COMMON DATA TYPES FOR DISTRIBUTED OPERATION -			
ChainingArguments			
originator	G	S	
targetObject	G	S	
operationProgress	G	S	
nameResolutionPhase	M	S	
nextRDNToBeResolved	G	S	
traceInformation	M	S	
aliasDereferenced	G	S	
aliasedRDNs	G	S	
returnCrossRefs	G	S	See Directory Documents, Part 4, subclause 10.4.1
referenceType	G	S	
DomainInfo	O	O	
timeLimit	G	S	
SecurityParameters	O	S	See note 1 (page ?) regarding the support classification for Request. Also see subclause 8.8
ChainingResults			
Info	O	O	
crossReferences	S	G	

Table 5 - DSP Protocol Support (continued)

Protocol Element	Support Classification		Comments
	Request	Response	
SecurityParameters	S	O	See note 1 (page ?) regarding the support classification for Response. Also see subclause 8.8
CrossReference			
contextPrefix	S	M	See Directory Documents, Part 4, subclause 12.4.2.2
accessPoint	S	M	
TraceInformation			
TraceItem	M	S	
TraceItem			
dsa	M	S	
targetObject	G	S	
operationProgress	M	S	
nameResolutionPhase	M	S	
nextRDNToBeResolved	G	S	
ContinuationReference			
targetObject	S	M	
aliasedRDNs	S	G	
operationProgress	S	M	
nameResolutionPhase	S	M	
nextRDNToBeResolved	S	G	
rdnsResolved	S	G	
referenceType	S	G	
AccessPoint	S	M	
AccessPoint			
Name	S	M	
PresentationAddress	S	M	
pSelector	S	G	
sSelector	S	G	

Table 5 - DSP Protocol Support (concluded)

Protocol Element	Support Classification		Comments
	Request	Response	
tSelector	S	G	
nAddress	S	M	

NOTES

1 The support classification is G when supporting the chained mode of interaction.

2 Some of these operations may be necessary to support distributed authentication. This requirement is distinct from support for chained mode of interaction.

Table 6 - DAP Support for Digital Signature Protocol Conformance Profile.

Protocol Element	Support Classification		Comments
	DUA	DSA	
- COMMON ARGUMENTS / RESULTS -			
CommonArguments			
SecurityParameters			
certification-path	G	S	
name	G	S	
time	G	S	
random	G	S	
target	G	S	
requestor	G	S	
CommonResults			
SecurityParameters	S	G	
performer	S	G	

Table 7 - DSP Support for Digital Signature Protocol Conformance Profile.

Protocol Element	Support Classification		Comments
	DUA	DSA	
- COMMON ARGUMENTS / RESULTS -			
CommonArguments			
SecurityParameters			
certification-path	G	S	
name	G	S	
time	G	S	
random	G	S	
target	G	S	
requestor	G	S	
CommonResults			
SecurityParameters	G	S	
performer	O	G	

Table 8 - DAP Support for Strong Authentication Protocol Conformance Profile.

Protocol Element	Support Classification		Comments
	DUA	DSA	
DirectoryBindArgument	M	S	
credentials	G	S	
simple	G	S	
name	G	S	
validity	G	S	
password	G	S	
strong			
certification-path	G	S	
bind-token	G	S	
externalProcedure	O	O	
versions	O	S	
DirectoryBindResult	S	G	
credentials	S	G	
simple	S	G	
name	S	G	
validity	S	G	
password	S	G	
strong	S	G	
certification-path	S	G	
bind-token	S	G	
externalProcedure	O	O	
versions	S	O	

Table 9 - DSP Support for Strong Authentication Protocol Conformance Profile.

Protocol Element	Support Classification		Comments
	DUA	DSA	
DirectoryBindArgument	M	S	
credentials	G	S	
simple	G	S	
name	G	S	
validity	G	S	
password	G	S	
strong			
certification-path	G	S	
bind-token	G	S	
externalProcedure	O	O	
versions	O	S	
DirectoryBindResult	S	G	
credentials	S	G	
simple	S	G	
name	S	G	
validity	S	G	
password	S	G	
strong	S	G	
certification-path	S	G	
bind-token	S	G	
externalProcedure	O	O	
versions	S	O	

Table 10 - Error Symptoms

Symptom	Description
E_ACCESS	The initiator has insufficient access rights to carry out this operation.
E_ADMIN_LIMIT	The Directory has reached some limit set by an administrative authority, and no partial results are available to return to the user.
E_ALIAS_DEREF	One of three situations exists: **NOTES** 1 An alias has been encountered while a previous alias was being dereferenced, or 2 a name contained an alias plus one or more additional RDNs when the dontDereferenceAliases service control was being used, or 3 the name, supplied in an operation that precludes alias dereferencing, contained an alias plus one or more additional RDNs.
E_ALIAS_LOOP	During a whole-subtree search operation, an alias has been encountered which would lead to a loop (i.e., the alias points to an entry which is superior to entries which have already been evaluated in carrying out the search).
E_ALIAS_PROBLEM	An alias has been encountered, but the entry to which it points does not exist.
E_ARG_BOUNDS	The argument does not comply with pragmatic constraints (defined locally or by functional standards).

Table 10 - Error Symptoms (continued)

Symptom	Description
E_ARG_SYNTAX	An operation argument either has incorrect ASN.1 encoding or correct ASN.1 encoding, but does not comply to the syntax as defined in the Directory Documents. **NOTES** 1 Within BindArgument, additional elements are permitted, to allow future extensions, and do not create an error situation. 2 Errors within attribute values are not included in this codification (see E_ATT_SYNTAX).
E_ARG_VIOL	An operation argument has correct syntax, but it violates additional rules and constraints levied by the Directory Documents (e.g., use of a Priority integer value whose meaning is undefined). **NOTES** 1 Within a Relative Distinguished Name, having two AVAs of the same attribute type is an error which is covered by E_DN, and not by E_ARG_VIOL. 2 Errors within attribute values are not included in this codification (see E_ATT_SYNTAX).
E_ATT_BOUNDS	An attribute value does not comply with bounds specified either by the Directory Documents or by functional standards.
E_ATT_OR_VALUE_EXISTS	Within an entry, an attribute or attribute value already exists, causing an error situation.
E_ATT_SYNTAX	An attribute value either has incorrect ASN.1 encoding or it has correct ASN.1 encoding but does not comply with the ASN.1 encoding defined by the attribute type.
E_ATT_VALUE	An attribute value, although of correct ASN.1 encoding, and conformant with the syntax defined for the attribute type, is not compliant with other rules (e.g.,a non-ISO 3166 country name encoding).
E_ACCESS	The initiator has insufficient access rights to carry out this operation.
E_AUTHENTICATION	The authentication offered does not match that required by the object being authenticated.

Table 10 - Error Symptoms (continued)

Symptom	Description
E_BUSY	The DSA is unable to handle this operation at this time (but it may be able to do so after a short while).
E_CHAIN	The DSA needs to use chaining to carry out this operation, but is prohibited from doing so by Service Controls.
E_CREDENTIALS	The credentials offered do not match those of the object with which authentication is taking place.
E_DBE	An inconsistency has been detected in the DSA's data base, which may be localized to a particular entry or set of entries.
E_DIT_STRUCTURE	An attempt was made via an add operation to place an entry in the DIB whose object class would violate the DIT structure rules.
E_DN	A DN contains an RDN with two AVAs of the same attribute type.
E_DSA	A DSA to which chaining is taking place is unable to respond.
E_ENTRY_EXISTS An	entry of the given name already exists, causing an error.
E_EXTENSION	A DSA was unable to satisfy a request because one or more critical extensions were not available.
E_ILLEGAL_ROOT_OBJ	Root's DN has been supplied as the object of a Read, Compare, AddEntry, RemoveEntry, ModifyEntry, ModifyRDN, or as the Base Object of a single level search.
E_ILLEGAL_ROOT_VAL	Root's DN has been supplied illegally as an attribute value (eg., as an Aliased Object Name).
E_LOOP	A loop has been detected in the knowledge information within the system.
E_MATCH	The attribute specified does not support the required matching capability.
E_MISSING_AVA	When creating, or after modifying, an entry, an AVA in the entry's RDN is not represented within the entry's set of attributes.
E_MISSING_OBJECT_CLASS	When creating an entry, the entry does not possess an object class.
E_MULTI_DSA	The operation is an update operation which affects other DSAs.
E_NAMING_VIOLATION	The name of the new or modified entry is incompatible with its object class.
E_NON_LEAF_OPERATION	The operation being attempted is illegal except on a leaf.
E_NONNAMING_ATTRIBUTE	In either an add or ModifyRDN operation, an attribute is included in the last RDN that is nota valid naming attribute according to the DIT structure rules.

Table 10 - Error Symptoms (continued)

Symptom	Description
E_NOT_SINGLE_VALUED	An attribute, registered as single-valued, has been found with more than one value.
E_NO_SUCH_ATT	The specified attribute has not been found.
E_NO_SUCH_OBJECT	The specified entry has not been found.
E_NO_SUCH_VALUE	The specified attribute value has not been found.
E_OBJECT_CLASS_MOD	An (illegal) attempt has been made to alter or remove an object class attribute.
E_OBJECT_CLASS_VIOL	There is a schema violation (e.g., missing mandatory attribute, or non-allowed attribute present).
E_REFERENCE	An erroneous reference has been detected (e.g., DSA cannot handle name even as far as the number of RDNs that have already been resolved).
E_SCOPE	No referrals were available within the requested scope.
E_SYSTEM_PERM	A serious and permanent software or system error has been detected which prevents completion of the operation.
E_SYSTEM_TEMP	A serious but temporary software or system error has been detected which prevents completion of the operation.
E_TIMEOUT	The operation has not completed within the allotted time.
E_UNABLE_TO_COMPLETE	The DSA is unable to complete this operation, or others like it (this applies particularly to search).
E_UNABLE_TO_PROCEED	The DSA cannot satisfy the operation after receiving it on the basis of a valid non-specific subordinate reference.
E_UNDEFINED_ATT	An unregistered attribute has been encountered.
E_UNSUPPORTED_OC	The object class of the entry is not supported as a valid object class for entries within this DSA.
E_VERSION	An unexpected version has been found in Bind.
E_ZERO_VALUES	An attribute has been found (e.g., as a result of a modify-entry operation) with no values.

Table 11 - Error Situations.

Situation	Description
BIND-LOCAL	A bind is being attempted; either the entry named is (or should be) within a local naming context, or name resolution is being carried out on the part of the name that is known locally.
BIND-REMOTE	A bind is being attempted, and the entry named is not within a local naming context; remote validation of credentials is being carried out.
NAME-RESOLUTION	Name resolution is being carried out.
ADD-ENTRY-NAME-RESOLUTION	During an add entry operation, name resolution has been successfully accomplished on the superior object, and is not being carried out to determine whether the new entry already exists.
ADD-ENTRY	The entry is being generated.
MODIFY-ENTRY	The entry is being modified.
MODIFY-RDN	The RDN is being modified.
REMOVE-ENTRY	The entry is being removed.
READ	The entry is being read.
COMPARE	A Compare operation is being carried out on the entry.
LIST	A List operation is being carried out on the entry.
SEARCH-FILTER	A Search operation is being carried out; the filter is being evaluated or acted upon.
SEARCH-ENTRY	A Search operation is being carried out; the required entry information is being evaluated or acted upon.
ABANDON	An Abandon operation is being carried out.
TRACE-EVALUATION	The trace element is being evaluated for loops.

Table 12 - Notation Used to Describe Error Actions.

Error Action Notation	Meaning
Rej	A reject operation is generated, with problem mistyped-argument.
Ab(<qualifier>)	Abandon Failed Error is generated. The qualifier may take on values codified as follows: CA - Cannot abandon NSO - No such operation TL - Too late
A(<qualifier>)	Attribute Error is generated. The qualifier may take on values codified as follows: AVE - Attribute or value already exists CV - Constraint violation IAS - Invalid attribute syntax IM - Inappropriate matching NSA - No such attribute UAT - Undefined attribute type
N(<qualifier>)	NameError is generated. The qualifier may take on values codified as follows: ADP - Alias dereferencing problem AP - Alias problem IAS - Invalid attribute syntax NSO - No such object
SC(<qualifier>)	Security Error is generated. The qualifier may take on values codified as follows: IA - Inappropriate authentication IAR - Insufficient access rights IC - Invalid credentials IS - Invalid signature NI - No information PR - Protection required

Table 12 - Notation Used to Describe Error Actions. (concluded)

Error Action Notation	Meaning
S(<qualifier>)	Service Error is generated. The qualifier may take on values codified as follows: ALE - Administrative limit exceeded B - Busy CR - Chaining required DE - Dit Error IR - Invalid reference LD - Loop detected OOS - Out of Scope TLE - Time limit exceeded UA - Unavailable UAP - Unable to proceed UCE - Unavailable critical extension UWP - Unwilling to perform
U(<qualifier>)	Update Error is generated. The qualifier may take on values codified as follows: AMD - Affects multiple DSAEAE - Entry already exist NAN - Not allowed on non-leaf NAR - Not allowed on RDN NV - Naming violation OCV - Object class violation OMP - Object class modification prohibited

Table 13 - Error Actions

Symptom (See Table 10)	Situation (See Table 11)					
	Bind-Local	Bind-Remote-Resolution	Name-Resolution	Add-Entry-Name-Resolution	Add-Entry	Modify-Entry
E_ACCESS			SC(IAR)(14)	SC(IAR)(14)	SC(IAR)(14)	SC(IAR)(14)
E_ADMIN_LIMIT	S(UA)	S(UA)	S(ALE)	S(ALE)	S(ALE)	S(ALE)
E_ALIAS_DEREF	S(IC)	S(IC)	N(ADP)			
E_ALIAS_LOOP						
E_ALIAS_PROBLEM	S(IC)	S(IC)	N(AP)			
E_ARG_BOUNDS	(8)	(7)	S(UWP)(12)	S(UWP)(12)	S(UWP)(12)	S(UWP)(12)
E_ARG_SYNTAX	(1)	(1)	Rej	Rej	Rej	Rej
E_ARG_VIOL	(1)	(1)	Rej	Rej	Rej	Rej
E_ATT_BOUNDS	SC(IC)	(7)	N(IAS)	N(IAS)	A(CV)	A(CV)
E_ATT_OR_VALUE_EXISTS					A(AVE)	A(AVE)
E_ATT_SYNTAX	SC(IC)	(7)	N(IAS)	N(IAS)	A(IAS)	A(IAS)
E_ATT_VALUE	SC(IC)	(7)	N(IAS)	N(IAS)	A(IAS)	A(IAS)
E_AUTHENTICATION	SC(IA)	SC(IA)				
E_BUSY	S(UA)	S(UA)	S(B)	S(B)	S(B)	S(B)
E_CHAIN				S(CR)		
E_CREDENTIALS	SC(IC)	SC(IC)				
E_DBE	S(UA)	S(UA)	S(DE)	S(DE)	S(DE)	S(DE)
E_DIT_STRUCTURE					U(NV)	
E_DN	SC(IC)	SC(IC)	N(NSO)	C(NV)		
E_DSA		S(UA)	S(UA)	S(UA)		

Table 13 - Error Actions (continued)

Symptom (See Table 10)	Situation (See Table 11)					
	Bind-Local	Bind-Remote-Resolution	Name-Resolution	Add-Entry-Name-Resolution	Add-Entry	Modify-Entry
E_ENTRY_EXISTS				U(EAE)		
E_EXTENSION			S(UWP)	S(UCE)	S(UCE)	S(UCE)
E_ILLEGAL_ROOT_OBJ	SC(IC)	SC(IC)		N(NSO)	N(NSO)	N(NSO)
E_ILLEGAL_ROOT_VAL	SC(IC)	(7)	N(IAS)	N(IAS)	A(IAS)	A(IAS)
E_LOOP		S(UA)	S(LD)			
E_MATCH	SC(IC)	SC(IC)	A(IM)	A(IM)		A(IM)
E_MISSING_AVA					U(NAR)	U(NAR)
E_MISSING_OBJECT_CLASS					U(OCV)	U(OMP)
E_MULTI_DSA				S(AMD)		
E_NAMING_VIOLATION				U(NV)		
E_NON_LEAF_OPERATION						
E_NONNAMING_ATTRIBUTE					U(NV)	
E_NOT_SINGLE_VALUED					A(CV)	A(CV)
E_NO_SUCH_ATT						A(NSA)
E_NO_SUCH_OBJECT	SC(IC)	SC(IC)	N(NSO)			
E_NO_SUCH_VALUE						A(NSA)
E_OBJECT_CLASS_MOD						U(OMP)
E_OBJECT_CLASS_VIOL					U(OCV)	U(OCV)
E_REFERENCE		S(UA)	S(IR)			
E_SCOPE			S(OOS)			
E_SYSTEM_PERM	S(UA)		S(UWP)	S(UWP)	S(UWP)	S(UWP)
E_SYSTEM_TEMP	S(UA)		S(UA)	S(UA)	S(UA)	S(UA)
E_TIMEOUT	S(UA)	(9)	S(TLE)	S(TLE)	S(TLE)	S(TLE)
E_UNABLE_TO_COMPLETE						
E_UNABLE_TO_PROCEED		(2)	(2)			
E_UNDEFINED_ATT	SC(IC)		(3)	U(NV)	A(UAT)	A(UAT)
E_UNSUPPORTED_OC					U(OCV)	
E_VERSION	S(UA)					
E_ZERO_VALUES					A(CV)	A(CV)

Table 13 - Error Actions (continued)

Symptom (See Table 10)	Situation (See Table 11)				
	Modify-RDN	Remove-Entry	Read	Compare	Trace-Evaluation
E_ACCESS	SC(IAR)(14)	SC(IAR)(14)	SC(IAR)(14)	SC(IAR)(14)	
E_ADMIN_LIMIT	S(ALE)		S(ALE)	S(ALE)	
E_ALIAS_DEREF					
E_ALIAS_LOOP					
E_ALIAS_PROBLEM					
E_ARG_BOUNDS	S(UWP)(12)		S(UWP)(12)	S(UWP)(12)	
E_ARG_SYNTAX	Rej	Rej	Rej	Rej	Rej
E_ARG_VIOL	Rej	Rej	Rej	Rej	Rej
E_ATT_BOUNDS	N(IAS)			A(CV)	(7)
E_ATT_OR_VALUE_EXISTS					
E_ATT_SYNTAX	N(IAS)			A(IAS)	(7)
E_ATT_VALUE	N(IAS)			A(IAS)	(7)
E_AUTHENTICATION					
E_BUSY	S(B)	S(B)	S(B)	S(B)	
E_CHAIN					
E_CREDENTIALS					
E_DBE	S(DE)	S(DE)	S(DE)	S(DE)	
E_DIT_STRUCTURE					
E_DN	A(CV)			A(IAS)	
E_DSA					
E_ENTRY_EXISTS	U(EAE)				
E_EXTENSION	S(UCE)	S(UCE)	S(UCE)	S(UCE)	

Table 13 - Error Actions (continued)

Symptom (See Table 10)	Situation (See Table 11)				
	Modify-RDN	Remove-Entry	Read	Compare	Trace-Evaluation
E_ILLEGAL_ROOT_OBJ	N(NSO)	N(NSO)	N(NSO)	N(NSO)	
E_ILLEGAL_ROOT_VAL	N(IAS)			A(IAS)	(7)
E_LOOP					
E_MATCH	A(IM)			A(IM)	(7)
E_MISSING_AVA					
E_MISSING_OBJECT_CLASS					
E_MULTI_DSA	S(AMD)	S(AMD)			
E_NAMING_VIOLATION	U(NV)				
E_NON_LEAF_OPERATION	U(NAN)	U(NAN)			
E_NONNAMING_ATTRIBUTE					
E_NOT_SINGLE_VALUED	A(CV)				
E_NO_SUCH_ATT			A(NSA)(4)	A(NSA)(4)	
E_NO_SUCH_OBJECT					
E_NO_SUCH_VALUE					
E_OBJECT_CLASS_MOD					
E_OBJECT_CLASS_VIOL	U(OCV)				
E_REFERENCE					
E_SCOPE					
E_SYSTEM_PERM	S(UWP)	S(UWP)	S(UWP)	S(UWP)	S(UWP)
E_SYSTEM_TEMP	S(UA)	S(UA)	S(UA)	S(UA)	S(UA)
E_TIMEOUT	S(TLE)	S(TLE)	S(TLE)	S(TLE)	
E_UNABLE_TO_COMPLETE					
E_UNABLE_TO_PROCEED					
E_UNDEFINED_ATT	A(UAT)		A(NSA)(4)	A(NSA)	(7)
E_UNSUPPORTED_OC					
E_VERSION					
E_ZERO_VALUES					(11)

Table 13 - Error Actions (continued)

Symptom (See Table 10)	Situation (See Table 11)			
	List (Filter)	Search (Filter)	Search Entry	Abandon
E_ACCESS	SC(IAR)(14)	SC(IAR)(14)	SC(IAR)(14)	
E_ADMIN_LIMIT	S(ALE)(13)	S(ALE)(13)	S(ALE)(13)	
E_ALIAS_DEREF		(5)		
E_ALIAS_LOOP		(5)		
E_ALIAS_PROBLEM		(5)		
E_ARG_BOUNDS	S(UWP)(12)	S(UWP)(12)	S(UWP)(12)	
E_ARG_SYNTAX	Rej	Rej	Rej	Rej
E_ARG_VIOL	Rej	Rej	Rej	
E_ATT_BOUNDS		A(CV)		
E_ATT_OR_VALUE_EXISTS				
E_ATT_SYNTAX		A(IAS)		
E_ATT_VALUE		A(IAS)		
E_AUTHENTICATION				
E_BUSY	S(B)	S(B)	S(B)	
E_CHAIN				
E_CREDENTIALS				
E_DBE	S(DE)	S(DE)	S(DE)	
E_DIT_STRUCTURE				
E_DN		A(IAS)		
E_DSA		(5)		
E_ENTRY_EXISTS				
E_EXTENSION	S(UCE)(13)	S(UCE)(13)	S(UCE)(13)	

Table 13 - Error Actions (continued)

Symptom (See Table 10)	Situation (See Table 11)			
	List (Filter)	Search (Filter)	Search Entry	Abandon
E_ILLEGAL_ROOT_OBJ		(10)		
E_ILLEGAL_ROOT_VAL		A(IAS)		
E_LOOP		(5)		
E_MATCH		A(IM)		
E_MISSING_AVA				
E_MISSING_OBJECT_CLASS				
E_MULTI_DSA				
E_NAMING_VIOLATION				
E_NON_LEAF_OPERATION				
E_NONNAMING_ATTRIBUTE				
E_NOT_SINGLE_VALUED				
E_NO_SUCH_ATT				
E_NO_SUCH_OBJECT				
E_NO_SUCH_VALUE				
E_OBJECT_CLASS_MOD				
E_OBJECT_CLASS_VIOL				
E_REFERENCE				
E_SCOPE				
E_SYSTEM_PERM	S(UWP)	S(UWP)	S(UWP)	Ab(CA)
E_SYSTEM_TEMP	S(UA)	S(UA)	S(UA)	Ab(CA)
E_TIMEOUT	S(TLE)(13)	S(TLE)(13)	S(TLE)(13)	
E_UNABLE_TO_COMPLETES	(B)	S(B)	S(B)	Ab(CA)

Table 13 - Error Actions (continued)

Symptom (See Table 10)	Situation (See Table 11)			
	List (Filter)	Search (Filter)	Search Entry	Abandon
E_UNABLE_TO_PROCEED				
E_UNDEFINED_ATT		(6)	(6)	
E_UNSUPPORTED_OC				
E_VERSION				
E_ZERO_VALUES				

Table 13 - Notes (concluded)

NOTES

1 Use A-U-ABORT. Note, however, that extra elements are permitted here.

2 An ‘‘unable-to-proceed’’ error becomes SC(IC) for bind and N(NSO) for operations if no DSA contacted can located the object.

3 An undefined attributed encountered during name resolution is only an error- N(NSO) - if the entry is identified as local. See also Note 10 below.

4 The A(NSA) condition is reserved in the case of ‘‘read’’ for the situation when no attribute of the specific list provided can be returned (for reasons that include security errors).

5 Any failure to propagate a search causes abandonment of that part of the search.

6 Undefined attributes are regarded as not matched or found, but cause no errors in search.

7 This error, if detected, should be ignored; processing continues.

8 This error would occur as a result of a bind argument with a name containing too many RDNs for the DSA. Use either S(UA) or S(IC).

9 DSAs should use the time-limit service control with local timeout to limit the remote validation of credentials; if the operation fails as a result, S(UA) is used.

10 For a single-entry search, N(NSO) may be used.

11 Either the whole attribute should be removed, or the deleteOldRDNflag should be ignored.

12 Wherever S(UWP) appears in the above tables beside EARGBOUNDS, a ROSE ‘‘Rej’’ is also admissible.

13 The error is returned when there are no partial results, otherwise a partialOutcomeQualifier with the appropriate limitProblem is returned (cf Directory Documents, Part 3, item g of clause 12.8.2,and Part 3, clause 10.1.3.3.1).

14 In every case where a security error occurs, except in bind, SC(NI) may be used in place of the specified problem, to support a Security Policy which states that no information on the problem may be divulged. In the case of the bind, SC(NI) is not available.

Table 14 - Simple Credential Fields and Protected Simple Authentication

Simple Credential Field	Equivalent Notation in Directory Documents, Part 8, figure 2
name	A
time1	t_1^A
time2	t_2^A
random1	q_1^A
random2	q_2^A
password	protected2