

INTERNET-DRAFT
Intended Category: Standard Track
Expires 20 May 2002
Obsoletes: RFC [2252](#)

K. Dally, Editor
The MITRE Corp.
S. Legg
ADACEL
[20](#) November 2001

**Lightweight Directory Access Protocol (v3):
Attribute Syntax Definitions
<[draft-ietf-ldapbis-syntaxes-01](#)>**

[Editor's note:

This Internet-Draft (I-D) is a modified version of the text of [RFC 2252](#), in order to bring it up to date. This action is part of the maintenance activity that is needed in order to progress LDAPv3 to Draft Standard. The changes are described in Annex C of this document. Open items are listed in Annex B.

End of Editor's note]

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of RFC 2026](#).

This document is intended to be, after appropriate review and revision, submitted to the RFC Editor as a Standard Track document. Distribution of this memo is unlimited. Technical discussion of this document will take place on the IETF LDAP Revision Working Group (LDAPbis) mailing list <ietf-ldapbis@openldap.org>. Please send editorial comments directly to the author <kdally@mitre.org>.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts. Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>. The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

Copyright 2000, The Internet Society. All Rights Reserved.

Please see the Copyright section near the end of this document for

more information.

Dally, Legg

Expires 20 May 2002

[Page 1]

Abstract

The Lightweight Directory Access Protocol (LDAP) [[1](#)] requires that the contents of AttributeValue fields in protocol elements be octet strings. This document defines a set of syntaxes for LDAPv3, and the rules by which attribute values of these syntaxes are represented in the LDAP protocol. The syntaxes defined in this document are referenced by this and other documents that define attribute types. In addition to defining the set of attribute syntaxes which LDAP servers should support, this document defines other schema elements (mandatory and optional) that are common to all LDAP servers.

Table of Contents

Status of Memo.....	1
Abstract.....	2
1. Overview.....	6
2. General Issues.....	6
2.1 Notation.....	6
2.2 Syntaxes.....	9
2.2.1 Syntaxes Implementation Status.....	9
2.2.2 Syntax Object Identifiers.....	10
2.2.3 Syntax Description.....	10
2.2.4 Example.....	11
2.3 Matching Rules.....	11
2.3.1 Matching Rules Implementation Status.....	11
2.3.2 Matching Rule Description.....	11
2.3.3 Example.....	12
2.4 Attribute Types.....	12
2.4.1 Attribute Types Implementation Status.....	13
2.4.2 Attribute Types Description.....	13
2.4.3 Example.....	15
2.5 Object Classes.....	15
2.5.1 Object Classes Implementation Status.....	15
2.5.2 Object Class Description.....	16
2.5.3 Example.....	16
3. Syntaxes.....	17
3.1 Attribute Type Description.....	17
3.2 Binary.....	17
3.3 Bit String.....	18
3.4 Boolean.....	19
3.5 Certificate.....	19
3.6 Certificate List.....	19
3.7 Certificate Pair.....	20
3.8 Country String.....	20
3.9 Delivery Method.....	20
3.10 Directory String.....	21
3.11 DIT Content Rule.....	21
3.12 DIT Structure Rule Description.....	22
3.13 DN.....	23
3.14 Enhanced Guide.....	23
3.15 Facsimile Telephone Number.....	24
3.16 Fax.....	24
3.17 Generalized Time.....	25
3.18 Guide.....	25
3.19 IA5 String.....	26

3.20	Integer.....	26
3.21	JPEG.....	26

3.22	LDAP Syntax Description	26
3.23	Matching Rule Description	27
3.24	Matching Rule Use Description	27
3.25	MHS OR Address	28
3.26	Name and Optional UID	28
3.27	Name Form Description	28
3.28	Numeric String	29
3.29	Object Class Description	29
3.30	Octet String	30
3.31	OID	30
3.32	Other Mailbox	30
3.33	Postal Address	31
3.34	Presentation Address	31
3.35	Printable String	31
3.36	Substring Assertion Syntax	32
3.37	Supported Algorithm	32
3.38	Telephone Number	33
3.39	Teletex Terminal Identifier	33
3.40	Telex Number	34
3.41	UTC Time	34
4.	Matching Rules	35
4.1	bitStringMatch	35
4.2	caseExactIA5Match	35
4.3	caseIgnoreIA5Match	35
4.4	caseIgnoreListMatch	35
4.5	caseIgnoreMatch	36
4.6	caseIgnoreOrderingMatch	36
4.7	caseIgnoreSubstringsMatch	36
4.8	distinguishedNameMatch	36
4.9	generalizedTimeMatch	36
4.10	generalizedTimeOrderingMatch	37
4.11	integerFirstComponentMatch	37
4.12	integerMatch	37
4.13	numericStringMatch	37
4.14	numericStringSubstringsMatch	38
4.15	objectIdentifierFirstComponentMatch	38
4.16	objectIdentifierMatch	38
4.17	octetStringMatch	38
4.18	presentationAddressMatch	39
4.19	protocolInformationMatch	39
4.20	telephoneNumberMatch	39
4.21	telephoneNumberSubstringsMatch	39
4.22	uniqueMemberMatch	40
5.	Attribute Types	40
5.1	altServer	40
5.2	attributeTypes	40

5.3	createTimestamp.....	40
5.4	creatorsName.....	41

5.5	dITContentRules	41
5.6	dITStructureRules	41
5.7	ldapSyntaxes	41
5.8	matchingRules	41
5.9	matchingRuleUse	42
5.10	modifiersName	42
5.11	modifyTimestamp	42
5.12	nameForms	42
5.13	namingContexts	42
5.14	objectClasses	43
5.15	subschemaSubentry	43
5.16	supportedControl	43
5.17	supportedExtension	44
5.18	supportedLDAPVersion	44
5.19	supportedSASLMechanisms	44
6	Object Classes	45
6.1	Extensible Object Class	45
6.2	subschema	45
7	Security Considerations	46
7.1	Disclosure	46
7.2	Use of Attribute Values in Security Applications	46
7.3	Securing the Directory	46
8	Acknowledgements	46
9	Author's Address	47
10	References	47
11	Full Copyright Statement	48
Annex A	Object Identifiers for Syntaxes	49
Annex B	Topics to be Addressed in This Document	50
Annex C	Change Log	51

1. Overview

This document defines the framework for developing schemas for directories accessible via the Lightweight Directory Access Protocol.

Schema is the collection of attribute type definitions, object class definitions and other information which specify the entries and their contents that a server holds. A server uses schema to determine how to match a filter or attribute value assertion (in a compare operation) against the attributes of an entry, and whether to permit add and modify operations.

Therefore, [Section 2](#) states the general requirements and notations for definition of attribute types, object classes, syntaxes and matching rules.

[Section 3](#) lists syntaxes, [section 4](#) matching rules, [section 5](#) attribute types, and [section 6](#) object classes.

Additional documents define schemas for representing real-world objects as directory entries.

2. General Issues

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#)[2].

This document describes the syntaxes of data conveyed in an Internet protocol.

Attribute Type and Object Class definitions are written in a string representation of the AttributeTypeDescription and ObjectClassDescription data types defined in X.501(93)[3]. Implementors are strongly advised to first read the description of how schema is represented in X.500 before reading the rest of this document.

2.1 Notation

For the purposes of defining the rules for describing attribute syntaxes and other schema elements, the following Backus-Naur Form (BNF) definitions will be used. They are based on the BNF styles of [RFC 822](#) [4].

```
a = "a" / "b" / "c" / "d" / "e" / "f" / "g" / "h" / "i" / "j" /  
    "k" / "l" / "m" / "n" / "o" / "p" / "q" / "r" / "s" / "t" /  
    "u" / "v" / "w" / "x" / "y" / "z" / "A" / "B" / "C" / "D" /
```

"E" / "F" / "G" / "H" / "I" / "J" / "K" / "L" / "M" / "N" /

Dally, Legg

Expires 20 May 2002

[Page 6]

```

    "O" / "P" / "Q" / "R" / "S" / "T" / "U" / "V" / "W" / "X" /
    "Y" / "Z"

d = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9"

hex-digit = d / "a" / "b" / "c" / "d" / "e" / "f" /
           "A" / "B" / "C" / "D" / "E" / "F"

k = a / d / "-"

p = a / d / "'" / "(" / ")" / "+" / "," / "-" / "." / "=" /
   "/" / ":" / "?" / " "

numericstring = 1*d

anhstring = 1*k

keystring = a [ anhstring ]

printablestring = 1*p

space = 1*" "

whsp = [ space ]

utf8 = <any sequence of octets formed from the UTF-8[9]
      transformation of a character from ISO 10646[10]
      except ">

dstring = 1*( utf8 / "'" ) ; escaped utf8 string, each "'"
                        ; appearing in the value to be encoded is
                        ; escaped by a preceding "'"

qdstring = "'" dstring "'"

qdstringlist = [ qdstring *( space qdstring ) ]

qdstrings = qdstring / ( "(" whsp qdstringlist whsp ")" )

```

In the following BNF for the string representation of OBJECT IDENTIFIERS, 'descr' is the syntactic representation of an object descriptor, which consists of letters, digits, and hyphens starting

with a letter. An OBJECT IDENTIFIER in the numericoid format should not have leading zeroes (e.g. "0.9.3" is permitted but "0.09.3" should not be generated).

When 'oid' elements occur in a value, the 'descr' notation option SHOULD be used in preference to the 'numericoid'. An object descriptor is more readable than a numeric OBJECT IDENTIFIER, and a descriptor (where assigned and known by the implementation) SHOULD

be used in preference to numeric oids to the greatest extent possible. Examples of object descriptors in LDAP are attribute type, object class, and matching rule names.

```
oid = descr / numericoid

descr = keystring

numericoid = numericstring *( "." numericstring )

noidlen = numericoid [ "{" len "}" ]

len = numericstring

oids = oid / ( "(" space oidlist space ")" ) ; set of oids of
                                           ; either form

oidlist = oid *( space "$" space oid )

qdescriptors = qdescr / ( "(" whsp qdescriptorlist whsp ")" ) ; object
                                           ; descriptors used as schema element names

qdescriptorlist = [ qdescr *( whsp qdescr ) ]

qdescr = "'" descr "'"

xstring = "X-" 1*( a / "-" / "_" )

extensions = *( space xstring space qdstrings )
```

Note that while lines have been folded for readability in the definitions of schema elements, (e.g., `objectClassDescription` attribute), the values transferred in protocol would not contain newlines.

In cases where an arbitrary string, not a Distinguished Name or part of one, is used in a value of an attribute, a backslash quoting mechanism is used to escape the following separator symbol character, (such as `"`, `$` or `#`) if it should occur in that string. The backslash is followed by a pair of hexadecimal digits representing the next character. A backslash itself in the string which forms part of a larger syntax is always represented as `'\5C'` or `'\5c'`. An example is given in [section 3.33](#), `postalAddress` syntax.

Servers are not required to provide the same or any text in the description part of the subschema values they maintain.

[2.2](#) Syntaxes

This section defines general requirements for LDAPv3 attribute value syntaxes. All documents defining attribute syntaxes for use with LDAPv3 are expected to conform to these requirements.

Syntaxes are also defined for matching rules whose assertion value syntax is different from the attribute value syntax.

In an LDAP schema, an Object Identifier (OID) is assigned to a syntax definition when the syntax is named.

Syntaxes that are currently in use in this I-D and the user schema I-D[18] are specified in this document in [Section 3](#). The object identifiers for these syntaxes are listed in Annex A, also. The object identifiers for syntaxes not specified in this document are listed in the IANA _____. [Editor's note: For the time being, the undocumented syntaxes are listed at the end of Annex A. End editor's note.]

In X.501[3] and X.520[9], the definition of the syntax is part of the attribute specification and a distinct OID for the syntax is not assigned. As a result, X.501 does not define an attribute for publishing syntaxes explicitly in a subschema entry.

In [\[1\]](#) the encoding of the LDAPv3 protocol is specified. The protocol encapsulates values of attributes in many places. In this I-D, the encoding of the values is specified, as part of each syntax definition. These value encoding rules are termed "native LDAP encoding". The native LDAP encoding of a value is what is transmitted in the protocol, unless a transfer option has been invoked for the value. The transfer option mechanism and the Binary transfer option are defined in [\[1\]](#).

The native LDAP encoding defined for a given attribute syntax must produce octet-aligned values. To the greatest extent possible, the native LDAP encoding of a value should be usable for display purposes. In particular, encoding rules for attribute syntaxes defining non-binary values should produce strings that can be displayed with little or no translation by clients implementing LDAP. There are a few cases (e.g. audio) however, when it is not sensible to produce a human-readable representation.

[2.2.1](#) Syntaxes Implementation Status

Clients and servers need not implement all the syntaxes listed, and MAY implement other syntaxes.

Clients MUST NOT assume that the native LDAP encoding of a value of

an unrecognized syntax is a human-readable character string.

[2.2.2](#) Syntax Object Identifiers

Syntaxes for use with LDAPv3 are named by OBJECT IDENTIFIERS, which are dotted-decimal strings. These are not intended to be displayed to users.

The table in Annex A lists the syntaxes that have been defined for LDAPv3, thus far.

Other documents may define additional syntaxes. However, the definition of additional arbitrary syntaxes is strongly deprecated since it will hinder interoperability. Today's client and server implementations generally do not have the ability to dynamically recognize new syntaxes. In most cases, attributes will be defined with the syntax for directory strings.

A suggested minimum upper bound on the number of characters in a value with a string-based syntax, or the number of bytes in a value for all other syntaxes, may be indicated by appending this bound count inside of curly braces following the syntax name's OBJECT IDENTIFIER in an attribute type definition. See the "numericoid" production in paragraph 2.1. Such a bound is not part of the syntax name itself. For instance, "1.3.6.4.1.1466.0{64}" suggests that server implementations should allow a string to be 64 characters long, although they may allow longer strings. Note that a single character of the Directory String syntax may be encoded in more than one byte since UTF-8 is a variable-length encoding.

[2.2.3](#) Syntax Description

The following BNF is used in this document to associate a short description (e.g., a name) with a syntax OBJECT IDENTIFIER. The productions for whsp, numericoid, qdescrs and qdstring are given in paragraph 2.1. Implementors should note that future versions of this document may expand this definition to include additional terms. Terms whose identifier begins with "X-" are reserved for private experiments, and MUST be followed by a <space> and a <qdstrings> tokens.

```
SyntaxDescription = "(" whsp
                    numericoid
                    [ space "DESC" space qdstring ]
                    extensions
                    whsp ")"
```

Note that the SyntaxDescription BNF is also the BNF that defines the native LDAP encoding of values of the LDAP Syntax Description syntax.

[2.2.4](#) Example

For example, the syntax description of the INTEGER syntax for whole number values is:

```
( 1.3.6.1.4.1.1466.115.121.1.27 DESC 'INTEGER' )
```

[2.3](#) Matching Rules

The matching rules specified in this document are defined in [section 4](#).

Matching rules are used by servers to compare attribute values against assertion values when performing Search and Compare operations. They are also used to identify the value to be added or deleted when modifying entries, and are used when comparing a purported distinguished name with the name of an entry.

Most of the attributes given in this document have an equality matching rule defined.

...An OID is assigned to a matching rule when it is defined. A matching rule definition should not be changed without having a new OID assigned to it.

[2.3.1](#) Matching Rules Implementation Status

Servers which support matching rules and the extensibleMatch SHOULD implement all the matching rules in [section 4](#).

Servers MUST publish in the matchingRules attribute, the definitions of matching rules referenced by values of the attributeTypes and matchingRuleUse attributes in the same subschema entry. Other unreferenced matching rules MAY be published in the matchingRules attribute.

If the server supports the extensibleMatch, then the server MAY use the matchingRuleUse attribute to indicate the applicability of selected matching rules to designated attribute types in an extensibleMatch.

[2.3.2](#) Matching Rule Description

Matching rule descriptions are written according to the following BNF. The productions for numericoid, qdescrs, qdstring, oid, and whsp are given in paragraph 2.1. Implementors should note that future versions of this document may expand this BNF to include additional terms. Terms whose identifier begins with "X-" are reserved for private experiments, and MUST be followed by a <space>

and a <qdstrings> tokens.

```
MatchingRuleDescription = "(" whsp
    numericoid           ; MatchingRule identifier
    [ space "NAME" space qdescrs ]
    [ space "DESC" space qdstring ]
    [ space "OBSOLETE" ]
    space "SYNTAX" space numericoid
                        ; oid corrected to numericoid
    extensions
    whsp ")"
```

Note that the MatchingRuleDescription BNF is also the BNF that defines the native LDAP encoding of values of the Matching Rule Description syntax.

[2.3.3](#) Example

For example, in specifying a server which implements a privately-defined matching rule for performing sound-alike matches on Directory String-valued attributes, the matching rule could be written as (1.2.3.4.5 is an example, the OID of an actual matching rule would be different):

```
matchingRule: ( 1.2.3.4.5 NAME 'soundAlikeMatch'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

This description could be the one included in the subschema entry in the server. If this matching rule could be used with the attributes 2.5.4.41 and 2.5.4.15, the following could be the use description present in the subschema entry:

```
matchingRuleUse: ( 1.2.3.4.5 APPLIES ( givenName $ surname ) )
```

A client could then make use of this matching rule by sending a search operation in which the filter is of the extensibleMatch choice, the matchingRule field is "soundAlikeMatch", and the type field is "givenName" or "surName".

[2.4](#) Attribute Types

Attributes represent the characteristics of the real-world object which an entry represents. The attributes defined in this document are given in [section 5](#).

An OID is assigned to an attribute type when it is defined. An attribute type definition should not be changed without having a new OID assigned to it.

Dally, Legg

Expires 20 May 2002

[Page 12]

[2.4.1](#) Attribute Types Implementation Status

Servers MUST publish in the `attributeTypes` attribute of the same subschema entry, the definitions of attribute types referenced by values of the `objectClasses`, `nameForms`, `matchingRuleUse` and `dITContentRules` attributes, and attribute types referenced by the `SUP` field in values of the `attributeTypes` attribute itself. Other unreferenced attribute types MAY be published in the `attributeTypes` attribute.

Schema developers MUST NOT create attribute type definitions whose names conflict with attribute types defined for use with LDAP in existing standards-track RFCs.

All LDAP server implementations MUST recognize the attribute types defined in [section 5](#).

Servers MUST maintain values of these attributes in accordance with the definitions in X.501(93): `createTimestamp`, `modifyTimestamp`, `creatorsName`, `modifiersName`, `subschemaSubentry`, `attributeTypes`, `objectClasses`, `matchingRules`, and `matchingRuleUse`.

The `createTimestamp` and `creatorsName` attributes SHOULD appear in entries which were created using the Add operation.

The `modifyTimestamp` and `modifiersName` attributes SHOULD appear in entries which have been modified using LDAP update operations.

The `subschemaSubentry` attribute SHOULD appear in all entries.

Servers MUST recognize these attribute type names, but it is not required that a server provide values for these attributes, when the attribute corresponds to a feature which the server does not implement: `namingContexts`, `altServer`, `supportedExtension`, `supportedControl`, `supportedSASLMechanisms`, `supportedLDAPVersion`,

Servers MAY use the `ldapSyntaxes` attribute to list the syntaxes which are implemented.

All servers SHOULD recognize these attribute type names, although typically only X.500 servers will implement their functionality: `dITStructureRules`, `nameForms`, and `ditContentRules`.

For the status of user schema attribute types see section 3 of [\[12\]](#).

[2.4.2](#) Attribute Type Description

Attribute types are expressed according to the following BNF. The productions for `whsp`, `numericoid`, `qdescrs`, `qdstring`, `space`,

oid, and noidlen are given in paragraph 2.1. Implementors should

Dally, Legg

Expires 20 May 2002

[Page 13]

note that future versions of this document may expand this BNF to include additional terms. Terms which begin with the characters "X-" are reserved for private experiments, and MUST be followed by a <space> and a <qdstrings> tokens.

```
AttributeTypeDescription = "(" whsp
    numericoid                ; AttributeType identifier
    [ space "NAME" qdescrs ]   ; name used in AttributeType
    [ space "DESC" qdstring ]  ; description
    [ space "OBSOLETE" ]
    [ space "SUP" space oid ]   ; derived from this other
                                ; AttributeType
    [ space "EQUALITY" space oid ] ; Matching Rule name
    [ space "ORDERING" space oid ] ; Matching Rule name
    [ space "SUBSTR" space oid ]   ; Matching Rule name
    [ space "SYNTAX" space noidlen ] ; see section 2.3
    [ space "SINGLE-VALUE" ]       ; default multi-valued
    [ space "COLLECTIVE" ]        ; default not collective
    [ space "NO-USER-MODIFICATION" ] ; default user modifiable
    [ space "USAGE" space AttributeUsage ]
                                ; default userApplications
    extensions
    whsp ")"
```

Servers SHOULD provide at least one of the "SUP" and "SYNTAX" fields for each AttributeTypeDescription.

An AttributeDescription (i.e., the means of referring to an attribute in the protocol [\[1\]](#)) can be used as the value in a NAME part of an AttributeTypeDescription. Note that these are case insensitive. [Editor's Note: The preceding paragraph seems to be circular in nature, especially when looking at the AttributeType explanation in [\[1\]](#). What is the fix? End of Editor's Note]

Note that the AttributeTypeDescription does not list the matching rules which can be used with that attribute type in an extensibleMatch search filter. This is done using the matchingRuleUseDescription described in paragraph 3.24.

This document refines the schema description of X.501 [\[3\]](#) by requiring that the syntax field in an AttributeTypeDescription be a string representation of an OBJECT IDENTIFIER for the LDAP string syntax definition, and an optional indication of the maximum length of a value of this attribute (defined in [section 2.2.2](#)).

Note that the AttributeTypeDescription BNF is also the BNF that defines the Attribute Type Description syntax.

2.4.3 Example

For example, it would be useful for the directory to know when an entry was put into the directory. The following definition is an Attribute Type Description that could be used to specify such an attribute.

```
( 2.5.18.1 NAME 'createTimestamp'  
  EQUALITY generalizedTimeMatch  
  ORDERING generalizedTimeOrderingMatch  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 ; Generalized Time  
  SINGLE-VALUE  
  NO-USER-MODIFICATION  
  USAGE directoryOperation )
```

2.5 Object Classes

Object classes are used to categorize the kinds of entries stored in the directory and to determine what attributes are contained in those entries.

In general, every entry is defined in terms of an abstract class ("top"), at least one structural object class, and zero or more auxiliary object classes.

Whether an object class is abstract, structural, or auxiliary is defined when the object class OID is assigned. An object class definition should not be changed without having a new identifier assigned to it.

2.5.1 Object Classes Implementation Status

Servers SHOULD implement the subschema object class.

Implementing the extensibleObject object class is optional.

Servers MUST publish in the objectClasses attribute of the same subschema entry, the definitions of object classes referenced by values of the nameForms and dITContentRules attributes, and object classes referenced by the SUP field in values of the objectClasses attribute itself. Other unreferenced object classes MAY be published in the objectClasses attribute.

Schema developers MUST NOT create object class definitions whose names conflict with object classes defined for use with LDAP in existing standards-track RFCs.

2.5.2 Object Class Description

Object class descriptions are written according to the following BNF. The productions for whsp, numericoid, qdscrs, qdstring, space, and oids are given in paragraph 2.1. Implementors should note that future versions of this document may expand this definition to include additional terms. Terms whose identifier begins with "X-" are reserved for private experiments, and MUST be followed by a <space> and a <qdstrings> tokens.

```
ObjectClassDescription = "(" whsp
    numericoid           ; ObjectClass identifier
    [ space "NAME" space qdscrs ]
    [ space "DESC" space qdstring ]
    [ space "OBSOLETE" ]
    [ space "SUP" space oids ]           ; Superior ObjectClasses
    [ space ( "ABSTRACT" / "STRUCTURAL" / "AUXILIARY" ) ]
    ; default structural
    [ space "MUST" space oids ]         ; AttributeTypes
    [ space "MAY" space oids ]         ; AttributeTypes
    extensions
    whsp ")"
```

2.5.3 Example

For example, information about an employee with respect to their job is useful in an application which queries the directory. The same pieces of information are needed in several kinds of entries, such as manager, part-time, and exempt employees. An auxiliary object class could be developed to be included in the structural object classes that represent the different kinds of employees. The pieces of information could be: name of the last training course attended, how many courses has the employee taken, category of training program. The types of information could be named the lastCourse, coursesCount, program attributes, respectively. The following could be the description of an auxiliary object class that provides for inclusion of the training information in different kinds of entries. (The OID is artificial.)

```
( 1.3.170.2.65 NAME 'trainingInfo'
    AUXILIARY
    MUST program
    MAY ( lastCourse $ coursesCount ) )
```


3. Syntaxes

3.1 Attribute Type Description

A value in this syntax is a definition of an attribute type according to the BNF given in paragraph 2.4.2. The native LDAP encoding is the character codes in UTF-8 which correspond to the characters in the definition.

This syntax is the form in which schema attribute types are published in the directory. The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.3 DESC 'Attribute Type Description' )
```

For example, this is the definition from [\[18\]](#) of the businessCategory attribute type:

```
( 2.5.4.15 NAME 'businessCategory'  
  EQUALITY caseIgnoreMatch  
  SUBSTR caseIgnoreSubstringsMatch  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{128} ) ; DirectoryString
```

This definition is a value of the Attribute Type Description syntax. The native LDAP encoding of this value is the definition itself.

3.2 Binary

[Editor's note: The binary syntax is not used in the core LDAPv3 I-Ds and could be removed. However, the syntax needs to be documented because documents external to the core are already using it (e.g., [RFC 2798](#)). What should be done?? End editor's note.]

A value in this syntax is a value of any ASN.1 type. The native LDAP encoding of a value of an attribute represented in this syntax is the BER encoding of a chosen ASN.1 type. The ASN.1 type is typically named in the "DESC" field of the attribute type definition.

The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.5 DESC 'Binary' )
```


For example, the ASN.1 type "NotarySeal" could be:

```
NotarySeal ::= SEQUENCE {
    notarizingAuthority DirectoryString{256},
    notaryName IA5String,
    seal OCTET STRING } -- The digital signature
                        -- of the notary.
```

An attribute type (with artificial OID) defined to hold a NotarySeal value could be:

```
( 1.2.3.0 NAME 'officialSeal'
  DESC 'the NotarySeal of the witnessing official'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 ) ; Binary
```

The encoding of an officialSeal value, where the value of NotarySeal is:

```
headTeller NotarySeal ::= { "Chief Judge",
                             "Kathleen Dally",
                             'AB 01 4F 09 09 05 FC AF AF CD EA'H }
```

would be:

```
'30 2A 13 0B 43 68 69 65 66 20 4A 75 64 67 65 16
 0E 4B 61 74 68 6C 65 65 66 20 44 61 6C 6C 79 04
 0B AB 01 4F 09 09 05 FC AF AF CD EA'H
```

In the protocol, this value would be a member of a SET which is the 'vals' part of the type/vals pair in an attribute list element of the result of an operation.

[3.3](#) Bit String

A value in this syntax is a value of the BIT STRING data type from ASN.1 [5]. The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.6 DESC 'Bit String' )
```

The native LDAP encoding of a value is the following BNF:

```
bitstring = "" *binary-digit "B"
```

```
binary-digit = "0" / "1"
```

Example: '0101111101'B

[3.4](#) Boolean

A value in this syntax is a value of the BOOLEAN data type from ASN.1 [5]. That is, there are exactly two values: one value representing logically true, and the other representing logically false. The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.7 DESC 'Boolean' )
```

The native LDAP encoding of a value is the following BNF:

```
boolean = "TRUE" / "FALSE"
```

[3.5](#) Certificate

A value in this syntax is the binary string that results from BER/DER-encoding an X.509 [6] public key certificate. The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.8 DESC 'Certificate' )
```

Due to the changes from X.509(1988) to X.509(1993) and subsequent changes to the ASN.1 definition to support certificate extensions, the native LDAP encoding has not been completed for this syntax. Values in this syntax MUST only be transferred using the Binary transfer option (see [1]). The BNF for this syntax is being developed in the PKIX WG.

The BNF notation in [RFC 1778](#) [7] for "User Certificate" is not recommended to be used.

[3.6](#) Certificate List

A value in this syntax is the binary string that results from BER/DER-encoding an X.509 [6] certificate revocation list. The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.9 DESC 'Certificate List' )
```

Due to incompatibility of the X.509(1988) and X.509(1993)[6] ASN.1, the native LDAP encoding has not been completed for this syntax. Values in this syntax MUST only be transferred using the Binary transfer option (see [1]). The BNF for this syntax is being developed in the PKIX WG.

The BNF notation in [RFC 1778](#) [7] for "Authority Revocation List" is not recommended to be used.

[3.7](#) Certificate Pair

A value in this syntax is the binary string that results from BER/DER-encoding an X.509[6] public key certificate pair. The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.10 DESC 'Certificate Pair' )
```

The BNF notation in [RFC 1778](#) [7] for "Certificate Pair" is not recommended to be used.

[3.8](#) Country String

A value in this syntax is two ASN.1 printable string characters representing a country. The permitted values are as listed in ISO 3166[8]. The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.11 DESC 'Country String' )
```

The native LDAP encoding of a value is the following BNF:

```
CountryString = p p
```

The production for p is given in paragraph 2.1.

Example: US

[3.9](#) Delivery Method

A value in this syntax is a set of the ASN.1 enumerated INTEGER values that indicates, in preference order, the service(s) by which the user, represented by the entry, is willing and/or capable of receiving messages.

The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.14 DESC 'Delivery Method' )
```

The native LDAP encoding of a value is the following BNF:

```
delivery-value = pdm / ( whsp pdm space "$" space delivery-value )
```

```
pdm = "any" / "mhs" / "physical" / "telex" / "teletex" /  
      "g3fax" / "g4fax" / "ia5" / "videotex" / "telephone"
```

The production for space is given in paragraph 2.1.

Example: telephone \$ videotex

3.10 Directory String

A value in this syntax is a value of one of the TeletexString, PrintableString or UniversalString data types from ASN.1[5]. The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.15 DESC 'Directory String' )
```

The native LDAP encoding of a value is the character string itself.

Note: The form of DirectoryString is not indicated in protocol unless the binary option is used. Servers which convert to DAP MUST choose an appropriate form. Servers MUST NOT reject values merely because they contain legal Unicode characters outside of the range of printable ASCII.

Servers and clients MUST be prepared to receive arbitrary Unicode characters, including characters not presently assigned to any character set.

Example:

This is a string of DirectoryString containing #!%#@.

For characters in the PrintableString form, the value is the native LDAP encoding is the value itself.

If it is in the TeletexString form, then the characters are transliterated to their equivalents in UniversalString, and encoded in UTF-8[11].

If it is in the UniversalString or BMPString forms [10], UTF-8 is the native LDAP encoding.

3.11 DIT Content Rule Description

A value in this syntax is a definition of a DIT content rule according to the following BNF:

```
DITContentRuleDescription = "(" whsp
    numericoid                ; Structural ObjectClass identifier
    [ space "NAME" space qdescrs ]
    [ space "DESC" space qdstring ]
    [ space "OBSOLETE" ]
    [ space "AUX" space oids ]      ; Auxiliary ObjectClasses
    [ space "MUST" space oids ]     ; AttributeType identifiers
    [ space "MAY" space oids ]      ; AttributeType identifiers
    [ space "NOT" space oids ]      ; AttributeType identifiers
    extensions
    whsp ")"
```


The productions for whsp, numericoid, qdescrs, qdstring, space and oids are given in paragraph 2.1. Implementors should note that future versions of this document may expand this BNF to include additional terms. Terms which begin with the characters "X-" are reserved for private experiments, and MUST be followed by a <space> and a <qdstrings> tokens.

This syntax is the form in which schema content rules are published in the directory. The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.16 DESC 'DIT Content Rule
  Description' )
```

3.12 DIT Structure Rule Description

A value in the DIT Structure Rule Description syntax is a definition of a schema Structure Rule according to the following BNF:

```
DITStructureRuleDescription = "(" whsp
    ruleidentifier                ; DITStructureRule identifier
    [ space "NAME" space qdescrs ]
    [ space "DESC" space qdstring ]
    [ space "OBSOLETE" ]
    space "FORM" space oid        ; NameForm
    [ space "SUP" ruleidentifiers ] ; superior DITStructureRules
    extensions
    whsp ")"
```

```
ruleidentifier = numericstring
```

```
ruleidentifiers = ruleidentifier | "(" whsp ruleidentifierlist
    whsp ")"
```

```
ruleidentifierlist = [ ruleidentifier *( space ruleidentifier ) ]
```

[Editor's note: In all the manipulation, using \$ as the separator in ruleidentifierlist has disappeared. I think that a required space is the separator now. Is this true about space and is there a lost \$ problem? End editor's note]

The productions for whsp, numericstring, qdescrs, qdstring, space, and oid are given in paragraph 2.1.

The native LDAP encoding is the character codes in UTF-8 which correspond to the characters in the structure rule definition.

This syntax is the form in which schema structure rules are published in the directory.

The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.17 DESC 'DIT Structure Rule
  Description' )
```

[3.13](#) DN

A value in the Distinguished Name syntax is a structured set of the ASN.1 data types that are included in the DirectoryString syntax. The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.12 DESC 'DN' )
```

The native LDAP encoding of a value is defined in [\[12\]](#). Note that the native LDAP encoding is not reversible to the original BER encoding used in X.500 for Distinguished Names, as the CHOICE of any DirectoryString element in an RDN is not evident in the native LDAP encoding.. See the note in [section 3.10](#).

Examples (from [\[12\]](#)):

```
CN=Steve Kille,O=Isode Limited,C=GB
```

```
OU=Sales+CN=J. Smith,O=Widget Inc.,C=US
```

```
CN=L. Eagle,O=Sue\, Grabbit and Runn,C=GB
```

```
CN=Before\0DAfter,O=Test,C=GB
```

```
1.3.6.1.4.1.1466.0=#04024869,O=Test,C=GB
```

```
SN=Lu\C4\8Di\C4\87
```

[3.14](#) Enhanced Guide

A value in the Enhanced Guide syntax is the matching criteria and scope of operation in an Enhanced Filter.

The native LDAP encoding of a value is the following BNF:

```
EnhancedGuide = space oid whsp "#" whsp criteria whsp "#" whsp subset
```

```
subset = "baseobject" / "oneLevel" / "wholeSubtree"
```

```
criteria = or-term / "(" or-term ")"
```

```
or-term = and-term *( "|" and-term )
```



```
and-term = not-term *( "&" not-term )

not-term = "!" not-term /
           attributetype "$" match-type /
           "(" or-term ")" /
           "?true" / ; an empty "and" in the Criteria ASN.1 type
           "?false" ; an empty "or" in the Criteria ASN.1 type

match-type = "EQ" / "SUBSTR" / "GE" / "LE" / "APPROX"
```

The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.21 DESC 'Enhanced Guide' )
```

Example:

```
person#(sn)#oneLevel
```

[3.15](#) Facsimile Telephone Number

A value in the Facsimile Telephone Number syntax is a subscriber number on the (public) telephone network of a facsimile device. The native LDAP encoding of a value is the following BNF:

```
fax-number = printablestring [ "$" faxparameters ] ; telephone
              $ number, optionally
              $ followed by facsimile
              $parameters

faxparameters = faxparm / ( faxparm "$" faxparameters )

faxparm = "twoDimensional" / "fineResolution" / "unlimitedLength"
          / "b4Length" / "a3Width" / "b4Width" / "uncompressed"
```

The production for printablestring is given in paragraph 2.1.

The telephone number is based on E.123 [\[13\]](#).

A printablestring is the PrintableString data type from ASN.1 [\[5\]](#).

The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.22 DESC 'Facsimile Telephone Number' )
```

[3.16](#) Fax

A value in the Fax syntax is an image which is produced using the Group 3 facsimile process [\[14\]](#) to duplicate an object, such as a memo.

The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.23 DESC 'Fax' )
```

Values in this syntax are expressed as octet strings containing Group 3 Fax images as defined in [14].

[3.17](#) Generalized Time

A value in the Generalized Time syntax is a date and time. The year is given as a four-digit number.

The native LDAP encoding is a value of the GeneralizeTime data type from ASN.1[5]. Note that the time zone must be specified. It is strongly recommended that GMT time be used.

[NEW Editor's Note: Neither X.208 or X.680 require the time zone. I propose that the sentence be deleted. If necessary, individual attribute types can restrict time values to ones that indicate the time zone, i.e., the GMT time zone or the differential to it from the local time zone. To date, none of the X.500 and LDAPv3 standard attributes has made this restriction. End of Editor's Note]

The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.24 DESC 'Generalized Time' )
```

Example:

199412161032Z means 10:32 a.m. Dec. 16, 1994 in the Greenwich Mean Time time zone.

[3.18](#) Guide

A value in the Guide syntax is the matching criteria in a Filter.

The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.25 DESC 'Guide' )
```

The Guide syntax should not be used for defining new attributes. It is important for backwards compatibility with LDAPv2 systems.

The native LDAP encoding of a value is the following BNF:

```
guide-value = [ object-class "#" ] criteria
```

object-class = space oid

Dally, Legg

Expires 20 May 2002

[Page 25]

The criteria production is defined in the Enhanced Guide syntax in paragraph 3.14. The productions for oid and space are in paragraph 2.1.

[3.19](#) IA5 String

A value in the IA5 String syntax is a value of the IA5String data type from ASN.1[5]. International Alphabet 5 [[15](#)] (IA5) is the international version of the ASCII character set.

The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.26 DESC 'IA5 String' )
```

The native LDAP encoding of a value in this syntax is the character string value itself.

[3.20](#) INTEGER

A value in the INTEGER syntax is a whole number as specified in the INTEGER data type from ASN.1 [[5](#)].

The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.27 DESC 'INTEGER' )
```

The native LDAP encoding of a value is the decimal representation of the value, with each decimal digit represented by the its character equivalent. So, the number 1321 is represented by the character string "1321".

[3.21](#) JPEG

A value in the JPEG syntax is an image produced according to specific rules for light values. The native LDAP encoding of a value is strings containing JPEG images in the JPEG File Interchange Format (JFIF), as described in [[16](#)].

The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.28 DESC 'JPEG' )
```

[3.22](#) LDAP Syntax Description

A value in the LDAP Syntax Description syntax is a definition of a

LDAP syntax description according to the BNF given in [section 2.2.3](#).

Dally, Legg

Expires 20 May 2002

[Page 26]

The native LDAP encoding is the character codes in UTF-8 which correspond to the characters in the definition.

This syntax is the form in which schema syntax descriptions are published in the directory. The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.54 DESC 'LDAP Syntax Description' )
```

Note that, in X.520 [9], syntaxes are not labeled distinctly with respect to attributes.

[3.23](#) Matching Rule Description

A value in the Matching Rule Description syntax is a definition of a matching rule according to the BNF given in [section 2.3.2](#). The native LDAP encoding is the character codes in UTF-8 which correspond to the characters in the definition of a Matching Rule. This syntax is the form in which schema matching rules are published in the directory. The following syntax definition gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.30 DESC 'Matching Rule Description' )
```

[3.24](#) Matching Rule Use Description

A value in the Matching Rule Use Description syntax is a definition of a matching Rule and the attribute types with which the rule may be used in an extensibleMatch search filter according to the following BNF:

```
MatchingRuleUseDescription = "(" whsp
    numericoid                ; MatchingRule identifier
    [ space "NAME" space qdescribers ]
    [ space "DESC" space qdstring ]
    [ space "OBSOLETE" ]
    space "APPLIES" space oids    ; AttributeType identifiers
    extensions
    whsp ")"
```

The productions for whsp, numericoid, qdescribers, qdstring, space, and oids are given in paragraph 2.1. Implementors should note that future versions of this document may expand this BNF to include additional terms. Terms whose identifier begins with "X-" are reserved for private experiments, and MUST be followed by a <space> and a <qdstrings> tokens.

The native LDAP encoding is the character codes in UTF-8 which correspond to the characters in the definition.

The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.31 DESC 'Matching Rule Use
  Description' )
```

This syntax is the form in which schema matching rule usage permissions are published in the directory.

[3.25](#) MHS OR Address

A value in the MHS OR Address syntax is the addressing information of a user of an X.400 messaging service. The native LDAP encoding is defined in [RFC 1327](#) [17].

The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.33 DESC 'MHS OR Address' )
```

[3.26](#) Name and Optional UID

A value of the Name and Optional UID (Unique Identifier) syntax is a Distinguished Name as defined in paragraph 3.13 plus a bit string that differentiates the value from otherwise identical names.

The native LDAP encoding of a value is the following BNF:

```
NameAndOptionalUID = DistinguishedName [ "#" bitstring ]
```

The bitstring production is defined in [section 3.3](#).

Although the '#' character may occur in a string representation of a distinguished name, no additional special quoting is done.

Example:

```
1.3.6.1.4.1.1466.0=#04024869,0=Test,C=GB#'0101'B
```

The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.34 DESC 'Name And Optional UID' )
```

[3.27](#) Name Form Description

A value in the Name Form Description syntax is a definition of a name form according to the following BNF:

```
NameFormDescription = "(" whsp
```

numericoid ; NameForm identifier

Dally, Legg

Expires 20 May 2002

[Page 28]


```
[ space "NAME" space qdescrs ]
[ space "DESC" space qdstring ]
[ space "OBSOLETE" ]
space "OC" space oid           ; Structural ObjectClass
space "MUST" space oids        ; AttributeTypes
[ space "MAY" space oids ]     ; AttributeTypes
extensions
whsp ")"
```

The productions for whsp, numericoid, qdescrs, qdstring, oid, and oids are given in paragraph 2.1. Implementors should note that future versions of this document may have expanded this BNF to include additional terms.

A value indicates the one or more attributes in an entry type (e.g., person, device) that are used as the Relative Distinguished Name of the entries.

This syntax is the form in which schema name forms are published in the directory. The native LDAP encoding of a value is the character codes in UTF-8 which correspond to the characters in the definition.

The following syntax description gives the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.35 DESC 'Name Form Description' )
```

[3.28](#) Numeric String

A value in the Numeric String syntax is a series of numerals and spaces as specified in the NumericString data type from ASN.1 [\[5\]](#). The following string states the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.36 DESC 'Numeric String' )
```

The representation of a string in this syntax is the string value itself.

Example: 1997

[3.29](#) Object Class Description

A value in this syntax is a character string which expresses the definition of an object class according to the BNF given in paragraph 2.5.2. This syntax is the form in which schema object classes are published in the directory. The following string states the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.37 DESC 'Object Class Description' )
```


For example, the character string below specifies the country object class, which requires the c (country name) attribute and allows the searchGuide and description attributes. All of these schema elements are specified in RFC ____ [18].

```
( 2.5.6.2 NAME 'country' SUP top STRUCTURAL MUST c
  MAY ( searchGuide $ description ) )
```

[3.30](#) Octet String

A value in the Octet String syntax is a value of the OCTET STRING data type from ASN.1 [5]. The following string states the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.40 DESC 'Octet String' )
```

Values in this syntax are written as a series of 8-bit values, according to the octet string value notation specified in [5]. In the case of character strings, the characters themselves may be written.

Example:

```
secret
```

[3.31](#) OID

A value in the Object Identifier syntax is a series of integers, ordered as specified in the OBJECT IDENTIFIER data type from ASN.1 [5]. The following string states the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.38 DESC 'OID' )
```

Values in this syntax are expressed according to the BNF in paragraph 2.1 for "oid".

Examples: 1.2.3.4
cn

[3.32](#) Other Mailbox

A value in the Other Mailbox syntax gives a mail system name with the name of a mailbox in the system. The following string states the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.39 DESC 'Other Mailbox' )
```

Values in this syntax are written according to the following BNF:

```
otherMailbox = mailbox-type "$" mailbox
```

Dally, Legg

Expires 20 May 2002

[Page 30]

```
mailbox-type = printablestring
```

```
mailbox = <an encoded IA5 String>
```

The printablestring production is defined in paragraph 2.1.

In the above, mailbox-type represents the type of mail system in which the mailbox resides, for example "MCIMail"; and mailbox is the actual mailbox in the mail system defined by mailbox-type.

[3.33](#) Postal Address

A value in the Postal Address syntax is a series of strings which form an address in a physical mail system. The following string states the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.41 DESC 'Postal Address' )
```

Values in this syntax are written according to the following BNF:

```
postal-address = dstring *( "$" dstring )
```

In the above, each dstring component of a postal address value is written as a value of type Directory String syntax. Backslashes and dollar characters, if they occur in the component, are quoted as described in paragraph 2.1. Many servers limit the postal address to six lines of up to thirty characters.

The production for dstring is defined in paragraph 2.1.

Example:

```
1234 Main St.$Anytown, CA 12345$USA  
\241,000,000 Sweepstakes$PO Box 1000000$Anytown, CA 12345$USA
```

[3.34](#) Presentation Address

A value in the Presentation Address syntax is an OSI Application Layer address of a remote application. Values in this syntax are written as described in [RFC 1278](#) [19].

[Editor's note: Is this reference allowed, because [RFC 1278](#) is Informational as opposed to Standard? End of Editor's note]

The following string states the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.43 DESC 'Presentation Address' )
```

[3.35](#) Printable String

A value in the Printable String syntax is a series of alphabetic, numeric, and (limited) punctuation characters as specified in the

PrintableString data type from ASN.1 [\[5\]](#) and in production p of

Dally, Legg

Expires 20 May 2002

[Page 31]

paragraph 2.1. Values in this syntax are expressed as the string itself. The following string states the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.44 DESC 'Printable String' )
```

Example: This is a PrintableString.

[3.36](#) Substring Assertion Syntax

The Substring Assertion syntax is used in rules which may be used in substrings and extensible matching rules. When using a substrings assertion, substrings components are provided in a SubstringFilter sequence. The following string states the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.58 DESC 'Substring Assertion' )
```

When using a matching rule assertion, substring components are encoded according to the following BNF and provided as the matchValue of the MatchingRuleAssertion:

```
substring = [initial] any [final]
```

```
initial = value
```

```
any = "*" *(value "*")
```

```
final = value
```

The <value> production is UTF-8 string. Should the backslash or asterix characters be present in a production of <value>, they are quoted as described in [section 2.1](#).

[3.37](#) Supported Algorithm

A value in the Supported Algorithm syntax is the identifier of a cryptologic method with its intended usage and policies under which the algorithm is permitted. The following string states the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.49 DESC 'Supported Algorithm' )
```

No printable representation of values of the supportedAlgorithms attribute (see [\[18\]](#)) is defined in this document. Clients which wish to store and retrieve this attribute MUST use "supportedAlgorithms;binary", in which the value is transferred as a binary encoding.

[3.38](#) Telephone Number

A value in the telephone number syntax is the series of characters that express a number (address) assigned to a telephone system subscriber. The following string states the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.50 DESC 'Telephone Number' )
```

Values in this syntax are written as if they were Printable String types. Telephone numbers are defined in X.520 [\[9\]](#) to comply with the internationally agreed format for expressing international telephone numbers, Recommendation E.123 [\[15\]](#).

Example: +1 512 305 0280

[3.39](#) Teletex Terminal Identifier

A value in this syntax is a string of characters that express the identifier value assigned to a teletex service subscriber. The following string states the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.51 DESC 'Teletex Terminal  
Identifier' )
```

Values in this syntax are written according to the following BNF:

```
teletex-id = ttx-term 0*("$" ttx-param)

ttx-term   = printablestring

ttx-param  = ttx-key ":" ttx-value

ttx-key    = "graphic" / "control" / "misc" / "page" / "private"

ttx-value  = octetstring
```

In the above, the first printablestring is the encoding of the first portion of the teletex terminal identifier to be encoded, and the subsequent 0 or more octetstrings are subsequent portions of the teletex terminal identifier.

The production for printablestring is defined in paragraph 2.1.

[Editor's note: There is no production for octetstring in paragraph 2.1. How should it be defined? End of Editor's note]

3.40 Telex Number

A value in the Telex Number syntax is the number assigned to a telex system subscriber with the country and answerback values indicated.

The following string states the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.52 DESC 'Telex Number' )
```

Values in this syntax are written according to the following BNF:

```
telex-number = actual-number "$" country "$" answerback
```

```
actual-number = printablestring
```

```
country       = printablestring
```

```
answerback    = printablestring
```

In the above, actual-number is the syntactic representation of the number portion of the TELEX number being written, country is the TELEX country code, and answerback is the answerback code of a TELEX terminal.

The production for printablestring is defined in paragraph 2.1.

3.41 UTC Time

A value in the UTC Time syntax is a date and time indicating accuracy to minute or second. The year is given as a two-digit number. The following string states the OID assigned to this syntax:

```
( 1.3.6.1.4.1.1466.115.121.1.53 DESC 'UTC Time' )
```

Values in this syntax are written as if they were printable strings, formulated as specified for the UTCTime data type in ASN.1 [5].

Note that the time zone must be specified. It is strongly recommended that GMT time be used.

Note: This syntax is deprecated in favor of the Generalized Time syntax.

[Editor's note: The convention for interpretation of 2-digit year values should be here (at least by reference), but where is the LDAP convention specified? Is LDAP referring to X.500 for this? If so, where? End of Editor's note]

4. Matching Rules

When performing the `caseExactMatch`, `caseIgnoreMatch`, `caseIgnoreListMatch`, `telephoneNumberMatch`, `caseExactIA5Match` and `caseIgnoreIA5Match`, multiple adjoining whitespace characters are treated the same as an individual space, and leading and trailing whitespace is ignored.

4.1 bitStringMatch

The following BNF associates the `bitStringMatch` rule with the Bit String syntax:

```
( 2.5.13.16 NAME 'bitStringMatch'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.6 ) ; Bit String
```

This matching rule is used to test equality.

4.2 caseExactIA5Match

The following BNF associates the `caseExactIA5Match` rule with the IA5 String syntax:

```
( 1.3.6.1.4.1.1466.109.114.1 NAME 'caseExactIA5Match'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 ) ; IA5 String
```

This matching rule is used to test equality.

4.3 caseIgnoreIA5Match

The following BNF associates the `caseIgnoreIA5Match` rule with the IA5 String syntax:

```
( 1.3.6.1.4.1.1466.109.114.2 NAME 'caseIgnoreIA5Match'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 ) ; IA5 String
```

This matching rule is used to test equality.

4.4 caseIgnoreListMatch

The BNF below associates the `caseIgnoreListMatch` rule with the Postal Address syntax. The X.520 [] syntax for this matching rule is a SEQUENCE OF DirectoryString. Since the Postal Address syntax is such a sequence, it is used in defining the matching rule for LDAPv3, although the matching rule can be used with any SEQUENCE OF DirectoryString syntax/assertion.

```
( 2.5.13.11 NAME 'caseIgnoreListMatch'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.41 ) ; Postal Address
```


This matching rule is used to test equality.

[4.5](#) **caseIgnoreMatch**

The following BNF associates the caseIgnoreMatch rule with the Directory String syntax:

```
( 2.5.13.2 NAME 'caseIgnoreMatch'  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 ) ; Directory String
```

This matching rule is used to test equality.

[4.6](#) **caseIgnoreOrderingMatch**

The following BNF associates the caseIgnoreOrderingMatch rule with the Directory String syntax:

```
( 2.5.13.3 NAME 'caseIgnoreOrderingMatch'  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 ) ; Directory String
```

This matching rule is used to test inequality, i.e., greaterOrEqual or lessOrEqual.

The sort ordering for a caseIgnoreOrderingMatch is implementation-dependent.

[4.7](#) **caseIgnoreSubstringsMatch**

The following BNF associates the caseIgnoreSubstringsMatch rule with the Substring Assertion:

```
( 2.5.13.4 NAME 'caseIgnoreSubstringsMatch'  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.58 ) ; Substring Assertion
```

This matching rule is used to test substrings equality.

[4.8](#) **distinguishedNameMatch**

The following BNF associates the distinguishedNameMatch rule with the DN syntax:

```
( 2.5.13.1 NAME 'distinguishedNameMatch'  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 ) ; DN
```

This matching rule is used to test equality.

[4.9](#) **generalizedTimeMatch**

The following BNF associates the generalizedTimeMatch rule with the Generalized Time syntax:


```
( 2.5.13.27 NAME 'generalizedTimeMatch'  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 ) ; Generalized Time
```

This matching rule is used to test equality.

[4.10](#) **generalizedTimeOrderingMatch**

```
( 2.5.13.28 NAME 'generalizedTimeOrderingMatch'  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 ) ; Generalized Time
```

This matching rule is used to test inequality, i.e., greaterOrEqual or lessOrEqual.

[4.11](#) **integerFirstComponentMatch**

Implementors should note that the assertion syntax of this matching rule, an INTEGER, is different from the value syntax of attributes for which this is the equality matching rule.

```
( 2.5.13.29 NAME 'integerFirstComponentMatch'  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 ) ; INTEGER
```

This matching rule is used to test equality with the first component in a compound syntax.

Implementors should note that the assertion syntax of this matching rule, an INTEGER, is different from the value syntax of attributes for which this is the equality matching rule.

[4.12](#) **integerMatch**

The following BNF associates the integerMatch rule with the INTEGER syntax:

```
( 2.5.13.14 NAME 'integerMatch'  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 ) ; INTEGER
```

This matching rule is used to test equality.

[4.13](#) **numericStringMatch**

The following BNF associates the numericStringMatch rule with the Numeric String syntax:

```
( 2.5.13.8 NAME 'numericStringMatch'  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.36 ) ; Numeric String
```

This matching rule is used to test equality.

[4.14](#) **numericStringSubstringsMatch**

```
( 2.5.13.10 NAME 'numericStringSubstringsMatch'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.58 ) ; Substring Assertion
```

This matching rule is used to test substrings equality.

[4.15](#) **objectIdentifierFirstComponentMatch**

Implementors should note that the assertion syntax of this matching rule, an OID, is different from the value syntax of attributes for which this is the equality matching rule.

```
( 2.5.13.30 NAME 'objectIdentifierFirstComponentMatch'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 ) ; OID
```

This matching rule is used to test equality with the first component in a compound syntax.

If the client supplies an extensible filter using an `objectIdentifierFirstComponentMatch` whose `matchValue` is in the "descr" form, and the OID is not recognized by the server, then the filter is Undefined.

[4.16](#) **objectIdentifierMatch**

The following BNF associates the `objectIdentifierMatch` rule with the OID syntax:

```
( 2.5.13.0 NAME 'objectIdentifierMatch'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 ) ; OID
```

This matching rule is used to test equality.

Implementors should note that the assertion syntax of this matching rule, an OID, is different from the value syntax of attributes for which this is the equality matching rule.

If the client supplies a filter using an `objectIdentifierMatch` whose `matchValue oid` is in the "descr" form, and the oid is not recognized by the server, then the filter is Undefined.

[4.17](#) **octetStringMatch**

Servers which implement the `extensibleMatch` filter SHOULD allow the matching rule listed in this section to be used in the `extensibleMatch`. In general these servers SHOULD allow matching rules to be used with all attribute types known to the server, when the assertion syntax of the matching rule is the same as the value

syntax of the attribute.

Dally, Legg

Expires 20 May 2002

[Page 38]

The Octet String Match rule compares for equality an asserted octet string with an attribute value of type OCTET STRING.

The strings match if they are the same length and corresponding octets are identical.

```
( 2.5.13.17 NAME 'octetStringMatch'  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 )
```

[4.18](#) **presentationAddressMatch**

The following BNF associates the presentationAddressMatch rule with the Presentation Address syntax:

```
( 2.5.13.22 NAME 'presentationAddressMatch'  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.43 ) ; Presentation Address
```

This matching rule is used to test equality.

[4.19](#) **protocolInformationMatch**

The following BNF associates the protocolInformationMatch rule with the Protocol Information syntax:

```
( 2.5.13.24 NAME 'protocolInformationMatch'  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.42 ) ; Protocol Information
```

This matching rule is used to test equality.

[4.20](#) **telephoneNumberMatch**

The following BNF associates the telephoneNumberMatch rule with the Telephone Number syntax:

```
( 2.5.13.20 NAME 'telephoneNumberMatch'  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.50 ) ; Telephone Number
```

This matching rule is used to test equality.

[4.21](#) **telephoneNumberSubstringsMatch**

The following BNF associates the telephoneNumberSubstringsMatch rule with the Substring Assertion syntax:

```
( 2.5.13.21 NAME 'telephoneNumberSubstringsMatch'  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.58 ) ; Substring Assertion
```

This matching rule is used to test substrings equality.

[4.22](#) uniqueMemberMatch

The following BNF associates the uniqueMemberMatch rule with the Name and Optional UID syntax:

```
( 2.5.13.23 NAME 'uniqueMemberMatch'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.34 ) ; Name And Optional UID
```

This matching rule is used to test equality.

[5.](#) Attribute Types

[5.1](#) altServer

The values of this attribute are URLs of other servers which may be contacted when this server becomes unavailable. If the server does not know of any other servers which could be used this attribute will be absent. Clients may cache this information in case their preferred LDAP server later becomes unavailable.

```
( 1.3.6.1.4.1.1466.101.120.6 NAME 'altServer'
  ; [Editor's Note: DELETE EQUALITY caseIgnoreIA5Match
  ; OR SHOULD THIS BE caseExactIA5Match?? End Editor's Note]
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 ; IA5 String
  USAGE dSAOperation )
```

This attribute is only present in the root DSE (see [\[1\]](#) and [\[3\]](#)).

[5.2](#) attributeTypes

This attribute is typically located in the subschema entry.

```
( 2.5.21.5 NAME 'attributeTypes'
  EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.3 ; Attribute Type
  ; Description
  USAGE directoryOperation )
```

[5.3](#) createTimestamp

```
( 2.5.18.1 NAME 'createTimestamp'
  EQUALITY generalizedTimeMatch
  ORDERING generalizedTimeOrderingMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 ; Generalized Time
  SINGLE-VALUE
  NO-USER-MODIFICATION
  USAGE directoryOperation )
```


[5.4](#) creatorsName

```
( 2.5.18.3 NAME 'creatorsName'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 ; DN
  SINGLE-VALUE
  NO-USER-MODIFICATION
  USAGE directoryOperation )
```

[5.5](#) ditContentRules

```
( 2.5.21.2 NAME 'dITContentRules'
  EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.16 ; DIT Content Rule
                                           ; Description
  USAGE directoryOperation )
```

This attribute is located in the subschema entry.

[5.6](#) dITStructureRules

```
( 2.5.21.1 NAME 'dITStructureRules'
  EQUALITY integerFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.17 ; DIT Structure Rule
                                           ; Description
  USAGE directoryOperation )
```

This attribute is located in the subschema entry.

[5.7](#) ldapSyntaxes

This attribute is typically located in the subschema entry.

This attribute identifies the syntaxes implemented, with each value corresponding to one syntax.

```
( 1.3.6.1.4.1.1466.101.120.16 NAME 'ldapSyntaxes'
  EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.54 ; LDAP Syntax
                                           ; Description
  USAGE directoryOperation )
```

[5.8](#) matchingRules

This attribute is typically located in the subschema entry.

```
( 2.5.21.4 NAME 'matchingRules'
  EQUALITY objectIdentifierFirstComponentMatch
```



```
SYNTAX 1.3.6.1.4.1.1466.115.121.1.30 ; Matching Rule
                                         ; DESCRIPTION
USAGE directoryOperation )
```

[5.9](#) matchingRuleUse

This attribute is typically located in the subschema entry.

```
( 2.5.21.8 NAME 'matchingRuleUse'
  EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.31 ; Matching Rule Use
                                         ; Description
  USAGE directoryOperation )
```

[5.10](#) modifiersName

```
( 2.5.18.4 NAME 'modifiersName'
  EQUALITY distinguishedNameMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 ; DN
  SINGLE-VALUE
  NO-USER-MODIFICATION
  USAGE directoryOperation )
```

[5.11](#) modifyTimestamp

```
( 2.5.18.2 NAME 'modifyTimestamp'
  EQUALITY generalizedTimeMatch
  ORDERING generalizedTimeOrderingMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 ; Generalized Time
  SINGLE-VALUE
  NO-USER-MODIFICATION
  USAGE directoryOperation )
```

[5.12](#) nameForms

```
( 2.5.21.7 NAME 'nameForms'
  EQUALITY objectIdentifierFirstComponentMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.35 ; Name Form Description
  USAGE directoryOperation )
```

This attribute is located in the subschema entry.

[5.13](#) namingContexts

The values of this attribute correspond to naming contexts which this server masters or shadows. If the server does not master any information (e.g. it is an LDAP gateway to a public X.500 directory) this attribute will be absent. If the server believes it contains the entire directory, the attribute will have a single value, and

that value will be the empty string (indicating the null DN of the

root). This attribute will allow a client to choose suitable base objects for searching when it has contacted a server.

```
( 1.3.6.1.4.1.1466.101.120.5 NAME 'namingContexts'  
  ; [Editor's Note: DELETE EQUALITY distinguishedNameMatch  
  ; End of Editor's Note]  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 ; DN  
  USAGE dSAOperation )
```

This attribute is only present in the root DSE (see [\[1\]](#) and [\[3\]](#)).

[5.14](#) objectClasses

This attribute is typically located in the subschema entry.

```
( 2.5.21.6 NAME 'objectClasses'  
  EQUALITY objectIdentifierFirstComponentMatch  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.37 ; Object Class  
                                     ; Description  
  USAGE directoryOperation )
```

[5.15](#) subschemaSubentry

The value of this attribute is the name of a subschema entry (or subentry) where the server makes available attributes specifying the schema controlling the subject entry.

```
( 2.5.18.10 NAME 'subschemaSubentry'  
  EQUALITY distinguishedNameMatch  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 ; DN  
  NO-USER-MODIFICATION  
  SINGLE-VALUE  
  USAGE directoryOperation )
```

[5.16](#) supportedControl

The values of this attribute are the OBJECT IDENTIFIERS identifying controls which the server supports. If the server does not support any controls, this attribute will be absent.

```
( 1.3.6.1.4.1.1466.101.120.13 NAME 'supportedControl'  
  ; [Editor's Note: DELETE EQUALITY objectIdentifierMatch End  
  ; of Editor's Note]  
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 ; OID  
  USAGE dSAOperation )
```

This attribute is only present in the root DSE (see [\[1\]](#) and [\[3\]](#)).

[5.17](#) supportedExtension

The values of this attribute are OBJECT IDENTIFIERS identifying the supported extended operations which the server supports.

If the server does not support any extensions this attribute will be absent.

```
( 1.3.6.1.4.1.1466.101.120.7 NAME 'supportedExtension'
  ; [Editor's Note: DELETE EQUALITY objectIdentifierMatch End
  ; of Editor's Note]
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 ; OID
  USAGE dSAOperation )
```

This attribute is only present in the root DSE (see [\[1\]](#) and [\[3\]](#)).

[5.18](#) supportedLDAPVersion

The values of this attribute are the versions of the LDAP protocol which the server implements.

```
( 1.3.6.1.4.1.1466.101.120.15 NAME 'supportedLDAPVersion'
  ; Editor's Note: DELETE EQUALITY integerMatch
  ; ORDERING integerOrderingMatch End of Editor's Note]
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 ; INTEGER
  USAGE dSAOperation )
```

This attribute is only present in the root DSE (see [\[1\]](#) and [\[3\]](#)).

[5.19](#) supportedSASLMechanisms

The values of this attribute are the names of supported SASL mechanisms which the server supports. If the server does not support any mechanisms this attribute will be absent.

```
( 1.3.6.1.4.1.1466.101.120.14 NAME 'supportedSASLMechanisms'
  ; [Editor's Note: DELETE EQUALITY caseIgnoreMatch ; OR
  ; SHOULD THIS BE caseExactMatch?? End of Editor's Note]
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 ; Directory String
  USAGE dSAOperation )
```

This attribute is only present in the root DSE (see [\[1\]](#) and [\[3\]](#)).

6. Object Classes

6.1 Extensible Object Class

The extensibleObject object class, if present in an entry, permits that entry to optionally hold any attribute. The MAY attribute list of this class is implicitly the set of all attributes.

```
( 1.3.6.1.4.1.1466.101.120.111 NAME 'extensibleObject'
  SUP top
  AUXILIARY
  ; MAY all attributes is implied
)
```

The mandatory attributes of the other object classes of this entry are still required to be present.

Note that not all servers will implement this object class, and those which do not will reject requests to add entries which contain this object class, or modify an entry to add this object class.

Note that, if the server implements the extensibleObject class but an attribute is not recognized, this is the same case as for any other object class.

6.2 subschema

This object class contains a description of the schema that is applied in the server and is used in the subschema entry.

```
( 2.5.20.1 NAME 'subschema'
  AUXILIARY
  MAY ( dITStructureRules $
        nameForms $
        ditContentRules $
        objectClasses $
        attributeTypes $
        matchingRules $
        matchingRuleUse ) )
```

The ldapSyntaxes operational attribute may also be present in subschema entries. [Editor's Proposal: add "A Content Rule could be used to enable this." End of Editor's Proposal]

[7. Security Considerations](#)

[7.1 Disclosure](#)

Attributes of directory entries are used to provide descriptive information about the real-world objects they represent, which can be people, organizations or devices. Most countries have privacy laws regarding the publication of information about people.

[7.2 Use of Attribute Values in Security Applications](#)

The transformations of an AttributeValue value from its X.501 form to an LDAP string representation are not always reversible back to the same BER or DER form. An example of a situation which requires the DER form of a distinguished name is the verification of an X.509 certificate.

For example, a distinguished name consisting of one RDN with one AVA, in which the type is `commonName` and the value is of the `TeletexString` choice with the letters 'Sam' would be represented in LDAP as the string `CN=Sam`. Another distinguished name in which the value is still 'Sam' but of the `PrintableString` choice would have the same representation `CN=Sam`.

Applications which require the reconstruction of the DER form of the value SHOULD NOT use the string representation of attribute syntaxes when converting a value to LDAP format. Instead it SHOULD use the Binary syntax.

[7.3 Securing the Directory](#)

In order to protect the directory and its contents, strong authentication MUST have been used to identify the Client when an update operation is requested.

[Editor's Note: This paragraph has been provided at Kurt Zeilenga's suggestion. There is probably more to be said. Input please! End of Editor's Note]

[8. Acknowledgements](#)

This document is an update of [RFC 2252](#) by M. Wahl, A. Coulbeck, T. Howes, and S. Kille. [RFC 2252](#) was a product of the IETF ASID Working Group.

This document is based upon input of the IETF LDAPBIS working group. The authors wish to thank ____, J. Sermersheim, and K. Zeilenga for their significant contribution to this update.

9. Author's Address

Kathy Dally
The MITRE Corp.
7515 Colshire Dr., ms-W650
McLean VA 22102
USA

Phone: +1 703 883 6058
Email: kdally@mitre.org

10. References

- [1] [draft-ietf-ldapbis-protocol-xx](#), replacement for Wahl, M., Howes, T., and S. Kille, "Lightweight Directory Access Protocol (v3)", [RFC 2251](#), December 1997.
- [2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [RFC 2119](#), March 1997.
- [3] The Directory: Models. ITU-T Recommendation X.501, 1993.
- [4] Crocker, D., "Standard of the Format of ARPA-Internet Text Messages", STD 11, [RFC 822](#), August 1982.
- [5] Information Technology - Abstract Syntax Notation One (ASN.1): Specification of Basic Notation, ITU-T Recommendation X.680, 1994
- ...[6] The Directory: Authentication Framework. ITU-T Recommendation X.509 1993.
- [7] Howes, T., Kille, S., Yeong, W., Robbins, C., "The String Representation of Standard Attribute Syntaxes", [RFC 1778](#), March 1995.
- [8] ISO 3166, "Codes for the representation of names of countries".
- [9] The Directory: Selected Attribute Types. ITU-T Recommendation X.520, 1993.
- [10] Universal Multiple-Octet Coded Character Set (UCS) - Architecture and Basic Multilingual Plane, ISO/IEC 10646-1: 1993 (with amendments).
- [11] Yergeau, F., "UTF-8, a transformation format of Unicode and ISO 10646", [RFC 2044](#), October 1996.

- [12] [draft-ietf-ldapbis-dn-xx](#), replacement for Wahl, M., Kille, S., and T. Howes, "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names", [RFC 2253](#), December 1997.
- [13] Notation for national and international telephone numbers. ITU-T Recommendation E.123, 1988.
- [14] Standardization of Group 3 facsimile apparatus for document transmission - Terminal Equipment and Protocols for Telematic Services. ITU-T Recommendation T.4, 1993
- [15] International Reference Alphabet (IRA) (Formerly International Alphabet No. 5 or IA5) Information Technology - 7-Bit Coded Character Set for Information Interchange, ITU-T Recommendation T.50, 1992
- [16] JPEG File Interchange Format (Version 1.02). Eric Hamilton, C-Cube Microsystems, Milpitas, CA, September 1, 1992.
- [17] Hardcastle-Kille, S., "Mapping between X.400(1988)/ISO 10021 and [RFC 822](#)", [RFC 1327](#), May 1992.
- [18] [draft-ietf-ldapbis-user-schema-xx](#), replacement for Wahl, M., "A Summary of the X.500(96) User Schema for use with LDAPv3", [RFC 2256](#), December 1997.
- [19] Kille, S., "A String Representation for Presentation Addresses", [RFC 1278](#), November 1991.

11. Full Copyright Statement

Copyright (C) The Internet Society (2001). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Annex A Object Identifiers of Syntaxes

This list contains the object identifiers for the syntaxes used in this I-D and in the user schema I-D. The complete list of syntax object identifiers is maintained by IANA. See ____ for more information about IANA services.

Syntax of Value Represented	OBJECT IDENTIFIER
=====	=====
Attribute Type Description	1.3.6.1.4.1.1466.115.121.1.3
Bit String	1.3.6.1.4.1.1466.115.121.1.6
Certificate	1.3.6.1.4.1.1466.115.121.1.8
Certificate List	1.3.6.1.4.1.1466.115.121.1.9
Certificate Pair	1.3.6.1.4.1.1466.115.121.1.10
DN	1.3.6.1.4.1.1466.115.121.1.12
Delivery Method	1.3.6.1.4.1.1466.115.121.1.14
Directory String	1.3.6.1.4.1.1466.115.121.1.15
DIT Content Rule Description	1.3.6.1.4.1.1466.115.121.1.16
DIT Structure Rule Description	1.3.6.1.4.1.1466.115.121.1.17
Enhanced Guide	1.3.6.1.4.1.1466.115.121.1.21
Facsimile Telephone Number	1.3.6.1.4.1.1466.115.121.1.22
Generalized Time	1.3.6.1.4.1.1466.115.121.1.24
Guide	1.3.6.1.4.1.1466.115.121.1.25
IA5 String	1.3.6.1.4.1.1466.115.121.1.26
INTEGER	1.3.6.1.4.1.1466.115.121.1.27
LDAP Syntax Description	1.3.6.1.4.1.1466.115.121.1.54
LDAP Schema Definition	1.3.6.1.4.1.1466.115.121.1.56
LDAP Schema Description	1.3.6.1.4.1.1466.115.121.1.57
Master And Shadow Access Points	1.3.6.1.4.1.1466.115.121.1.29
Matching Rule Description	1.3.6.1.4.1.1466.115.121.1.30
Matching Rule Use Description	1.3.6.1.4.1.1466.115.121.1.31
Mail Preference	1.3.6.1.4.1.1466.115.121.1.32
MHS OR Address	1.3.6.1.4.1.1466.115.121.1.33
Modify Rights	1.3.6.1.4.1.1466.115.121.1.55
Name And Optional UID	1.3.6.1.4.1.1466.115.121.1.34
Name Form Description	1.3.6.1.4.1.1466.115.121.1.35

Numeric String	1.3.6.1.4.1.1466.115.121.1.36
Object Class Description	1.3.6.1.4.1.1466.115.121.1.37

Octet String	1.3.6.1.4.1.1466.115.121.1.40
OID	1.3.6.1.4.1.1466.115.121.1.38
Other Mailbox	1.3.6.1.4.1.1466.115.121.1.39
Postal Address	1.3.6.1.4.1.1466.115.121.1.41
Protocol Information	1.3.6.1.4.1.1466.115.121.1.42
Presentation Address	1.3.6.1.4.1.1466.115.121.1.43
Printable String	1.3.6.1.4.1.1466.115.121.1.44
Substring Assertion	1.3.6.1.4.1.1466.115.121.1.58
Subtree Specification	1.3.6.1.4.1.1466.115.121.1.45
Supplier Information	1.3.6.1.4.1.1466.115.121.1.46
Supplier Or Consumer	1.3.6.1.4.1.1466.115.121.1.47
Supplier And Consumer	1.3.6.1.4.1.1466.115.121.1.48
Supported Algorithm	1.3.6.1.4.1.1466.115.121.1.49
Telephone Number	1.3.6.1.4.1.1466.115.121.1.50
Teletex Terminal Identifier	1.3.6.1.4.1.1466.115.121.1.51
Telex Number	1.3.6.1.4.1.1466.115.121.1.52
UTC Time	1.3.6.1.4.1.1466.115.121.1.53

Annex B Topics Yet To Be Addressed In This Document

This appendix is provided for informational purposes only, it is not a normative part of this specification.

APPEARED: -00

Paragraph 2.2.3 - Should any syntaxes listed in the table be removed? Should any new syntaxes be added?

RESOLUTION: Cannot add syntaxes. Moving the table to an annex keeps a record of the OIDS that have been assigned. APPLIED:

APPEARED: -00

Paragraph 2.2.4 - Should attribute syntaxes be allowed to be referenced by a common name, and if so, where should the name come from?

RESOLUTION: Rejected because of adding functionality. APPLIED: -01

APPEARED: -00

How does the data model draft <[draft-wahl-ladpv3-defns-01.txt](#)> affect this draft?

RESOLUTION: It does not. The draft was preliminary to the revised Schema and Protocol I-Ds. APPLIED: -01

APPEARED: -00

[Section 3](#) - Should all listed syntaxes from paragraph 2.2.3 be detailed in this section? Nearly half the listed syntaxes are not referenced in this section.

RESOLUTION: No, because many are not being used, currently.

APPLIED: -01

APPEARED: -01

[Section 4](#) - Should all of the X.520(1993) matching rules be included? In particular, how about caseExactMatch? Also, should octetStringMatch be moved from updated [RFC 2256](#)?

RESOLUTION: APPLIED:

APPEARED: -00

[Section 6](#) - Recognized list of Object classes needs to be reconciled with updated [RFC 2256](#) and the data model draft.

RESOLUTION: Not necessary. APPLIED: -01

APPEARED: -00

[Section 7](#) - Proper security statement needs to be formulated.

RESOLUTION: Text has been expanded since [RFC 2252](#), but needs more work.

Annex C Change Log

This annex lists the changes that have been made from [RFC 2252](#) to this I-D.

This annex is provided for informational purposes only. It is not a normative part of this specification.

1. Removed the IESG Note.
2. Changed "types" to "syntaxes" in the last sentence of the Abstract. Also, added to the last sentence in order to indicate that syntaxes are not the only schema elements defined in this document.
3. Reorganized the sections so that:
 - * the schema element categories are specified in the order in which they build on one another: syntaxes, matching rules, attributes, object classes
 - * within each category the elements are specified in alphabetical order
4. Added an "Implementation Status" paragraph for each element, gathering the conformance statements.
5. Clarified schema description in the Overview.

6. Changed the "Common Encoding Aspects" section title to "Notation" and made corresponding changes throughout the document. The purpose being to relegate all encoding issues to the Protocol specification [1].
7. Added a MUST statement regarding the syntaxes required of servers.
8. Expanded the discussion of each of the syntaxes in [section 3](#).
9. Added examples to some of the syntax descriptions.
10. Added NAME option to the syntax description BNF in 2.2.4.

RESCINDED IN -01!!

11. Added a note deprecating the UTCTime attribute syntax description in 3.41
12. In the BNF of the MatchingRuleDescription in paragraph 2.3.2, replaced "numericoid" with "oid".
13. In paragraph 2.4.1, replaced the conformance statement about attributes in 2256 with a reference.
14. Added caseIgnoreIA5Match as the EQUALITY matching rule for the altServer attribute type BNF in paragraph 5.1. Note that this could be caseExactIA5Match instead. SHOULD IT BE??

RESCINDED IN -01

15. In paragraphs 5.10 and 5.11, changed "the MODIFY operation" to "LDAP update operations"
16. Added distinguishedNameMatch as the EQUALITY matching rule for the namingContexts attribute type BNF in paragraph 5.13.

RESCINDED IN -01

17. Reworded paragraph 5.15.
18. Added distinguishedNameMatch as the EQUALITY matching rule for the namingContexts attribute type BNF in paragraph 5.13.

RESCINDED IN -01

19. Added integerMatch as the EQUALITY and integerOrderingMatch as the Ordering matching rules for the supportedLDAPVersion attribute type BNF in paragraph 5.18.

RESCINDED IN -01

20. Added caseIgnoreMatch as the EQUALITY matching rule for the supportedSASLMechanisms attribute type BNF in paragraph 5.19. Note that this could be caseExactMatch instead. SHOULD IT BE??

RESCINDED IN -01

21. Made corrections to the BNF in paragraph 3.12.
22. Added the seven syntax definitions from [RFC 2256](#) and ordered the definitions alphabetically.
23. Changed the "Bibliography" section title to "References".
24. Replaced the X.208 reference with one to X.680(1994), since X.680 is the ASN.1 referred to in the X.500(1993)-series.

25. Moved the table listing the syntaxes and their oids from paragraph 2.2.3 to a new Annex A.
26. Moved the specification of the octetStringMatch matching rule from [RFC 2256](#) to [section 4](#) of this document.
27. Throughout this I-D, cleaned up whitespace in the BNF definitions.
28. Added the specification of the octetStringSubstringAssertion syntax to [section 3](#) of this document.
29. In [Section 2.1](#):
 - * Corrected the characters defined in the p rule to match the PrintableString syntax.
 - * Deleted the letterstring rule.
 - * Modified the utf8 and dstring rules according to a suggestion from K. Zeilenga.
 - * Deleted ";" from the k rule, which affects the anhstring, keystack, and descr rules.
 - * Removed the length option from the numericoid rule
30. In [section 2.2](#), deleted the sentence about needing a new OID when a syntax is modified.

31. In [section 2.2](#), replaced the editor's proposal and subject text with explanation of the native LDAP encoding of attribute values.
32. Removed [section 2.2.2](#) (and renumbered the remainder of [section 2.2](#)), leaving the description of binary encoding to the protocol I-D.

