Network Working Group
Request for Comments: 4517
Obsoletes: 2252, 2256
Updates: 3698
Category: Standards Track

S. Legg, Ed.
eB2Bcom

June 2006

Lightweight Directory Access Protocol (LDAP): Syntaxes and Matching Rules

Status of This Memo

Copyright Notice

Abstract

Each attribute stored in a Lightweight Directory Access Protocol (LDAP) directory, whose values may be transferred in the LDAP protocol, has a defined syntax that constrains the structure and format of its values. The comparison semantics for values of a syntax are not part of the syntax definition but are instead provided through separately defined matching rules. Matching rules specify an argument, an assertion value, which also has a defined syntax. This document defines a base set of syntaxes and matching rules for use in defining attributes for LDAP directories.

Table of Contents

1. Introduction

Each attribute stored in a Lightweight Directory Access Protocol (LDAP) directory [RFC4510], whose values may be transferred in the LDAP protocol [RFC4511], has a defined syntax (i.e., data type) that constrains the structure and format of its values. The comparison semantics for values of a syntax are not part of the syntax definition but are instead provided through separately defined matching rules. Matching rules specify an argument, an assertion value, which also has a defined syntax. This document defines a base set of syntaxes and matching rules for use in defining attributes for LDAP directories.

Readers are advised to familiarize themselves with the Directory Information Models [RFC4512] before reading the rest of this document. Section 3 provides definitions for the base set of LDAP syntaxes. Section 4 provides definitions for the base set of matching rules for LDAP.

This document is an integral part of the LDAP technical specification [RFC4510], which obsoletes the previously defined LDAP technical specification, RFC 3377, in its entirety.

Sections 4, 5, and 7 of RFC 2252 are obsoleted by [RFC4512]. The remainder of RFC 2252 is obsoleted by this document. Sections 6 and 8 of RFC 2256 are obsoleted by this document. The remainder of RFC 2256 is obsoleted by [RFC4519] and [RFC4512]. All but Section 2.11 of RFC 3698 is obsoleted by this document.

A number of schema elements that were included in the previous revision of the LDAP technical specification are not included in this revision of LDAP. Public Key Infrastructure schema elements are now specified in [RFC4523]. Unless reintroduced in future technical specifications, the remainder are to be considered Historic.

The changes with respect to RFC 2252 are described in Appendix B of this document.

2. Conventions

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

Syntax definitions are written according to the <SyntaxDescription> ABNF [RFC4234] rule specified in [RFC4512], and matching rule definitions are written according to the <MatchingRuleDescription> ABNF rule specified in [RFC4512], except that the syntax and matching rule definitions provided in this document are line-wrapped for readability. When such definitions are transferred as attribute values in the LDAP protocol (e.g., as values of the ldapSyntaxes and matchingRules attributes [RFC4512], respectively), then those values would not contain line breaks.

3. Syntaxes

Syntax definitions constrain the structure of attribute values stored in an LDAP directory, and determine the representation of attribute and assertion values transferred in the LDAP protocol.

Syntaxes that are required for directory operation, or that are in common use, are specified in this section. Servers SHOULD recognize all the syntaxes listed in this document, but are not required to otherwise support them, and MAY recognise or support other syntaxes. However, the definition of additional arbitrary syntaxes is discouraged since it will hinder interoperability. Client and server implementations typically do not have the ability to dynamically recognize new syntaxes.

3.1. General Considerations

The description of each syntax specifies how attribute or assertion values conforming to the syntax are to be represented when transferred in the LDAP protocol [RFC4511]. This representation is referred to as the LDAP-specific encoding to distinguish it from other methods of encoding attribute values (e.g., the Basic Encoding Rules (BER) encoding [BER] used by X.500 [X.500] directories).

The LDAP-specific encoding of a given attribute syntax always produces octet-aligned values. To the greatest extent possible, encoding rules for LDAP syntaxes should produce character strings that can be displayed with little or no translation by clients implementing LDAP. However, clients MUST NOT assume that the LDAP-specific encoding of a value of an unrecognized syntax is a human-readable character string. There are a few cases (e.g., the JPEG syntax) when it is not reasonable to produce a human-readable representation.

Each LDAP syntax is uniquely identified with an object identifier [ASN.1] represented in the dotted-decimal format (short descriptive names are not defined for syntaxes). These object identifiers are not intended to be displayed to users. The object identifiers for the syntaxes defined in this document are summarized in Appendix A.

A suggested minimum upper bound on the number of characters in an attribute value with a string-based syntax, or the number of octets in a value for all other syntaxes, MAY be indicated by appending the bound inside of curly braces following the syntax's OBJECT IDENTIFIER in an attribute type definition (see the <noidlen> rule in [RFC4512]). Such a bound is not considered part of the syntax identifier.

For example, "1.3.6.1.4.1.1466.115.121.1.15{64}" in an attribute definition suggests that the directory server will allow a value of the attribute to be up to 64 characters long, although it may allow longer character strings. Note that a single character of the Directory String syntax can be encoded in more than one octet, since UTF-8 [RFC3629] is a variable-length encoding. Therefore, a 64-character string may be more than 64 octets in length.

3.2. Common Definitions

The following ABNF rules are used in a number of the syntax definitions in Section 3.3.

```
PrintableCharacter = ALPHA / DIGIT / SQUOTE / LPAREN / RPAREN /
                     PLUS / COMMA / HYPHEN / DOT / EQUALS /
```

```
                        SLASH / COLON / QUESTION / SPACE
   PrintableString    = 1*PrintableCharacter
   IA5String          = *(%x00-7F)
   SLASH              = %x2F  ; forward slash ("/")
   COLON              = %x3A  ; colon (":")
   QUESTION           = %x3F  ; question mark ("?")
```

The <ALPHA>, <DIGIT>, <SQUOTE>, <LPAREN>, <RPAREN>, <PLUS>, <COMMA>, <HYPHEN>, <DOT>, <EQUALS>, and <SPACE> rules are defined in [RFC4512].

3.3. Syntax Definitions

3.3.1. Attribute Type Description

A value of the Attribute Type Description syntax is the definition of an attribute type. The LDAP-specific encoding of a value of this syntax is defined by the <AttributeTypeDescription> rule in [RFC4512].

For example, the following definition of the createTimestamp attribute type from [RFC4512] is also a value of the Attribute Type Description syntax. (Note: Line breaks have been added for readability; they are not part of the value when transferred in protocol.)

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

The LDAP definition for the Attribute Type Description syntax is:

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

This syntax corresponds to the AttributeTypeDescription ASN.1 type from [X.501].

3.3.2. Bit String

A value of the Bit String syntax is a sequence of binary digits. The LDAP-specific encoding of a value of this syntax is defined by the following ABNF:

```
   BitString    = SQUOTE *binary-digit SQUOTE "B"
   binary-digit = "0" / "1"
```

The <SQUOTE> rule is defined in [RFC4512].

Example:
'0101111101'B

The LDAP definition for the Bit String syntax is:

(1.3.6.1.4.1.1466.115.121.1.6 DESC 'Bit String')

This syntax corresponds to the BIT STRING ASN.1 type from [ASN.1].

3.3.3. Boolean

A value of the Boolean syntax is one of the Boolean values, true or false. The LDAP-specific encoding of a value of this syntax is defined by the following ABNF:

Boolean = "TRUE" / "FALSE"

The LDAP definition for the Boolean syntax is:

(1.3.6.1.4.1.1466.115.121.1.7 DESC 'Boolean')

This syntax corresponds to the BOOLEAN ASN.1 type from [ASN.1].

3.3.4. Country String

A value of the Country String syntax is one of the two-character codes from ISO 3166 [ISO3166] for representing a country. The LDAP-specific encoding of a value of this syntax is defined by the following ABNF:

CountryString = 2(PrintableCharacter)

The <PrintableCharacter> rule is defined in Section 3.2.

Examples:

US
AU

The LDAP definition for the Country String syntax is:

(1.3.6.1.4.1.1466.115.121.1.11 DESC 'Country String')

This syntax corresponds to the following ASN.1 type from [X.520]:

PrintableString (SIZE (2)) -- ISO 3166 codes only

3.3.5. Delivery Method

A value of the Delivery Method syntax is a sequence of items that indicate, in preference order, the service(s) by which an entity is willing and/or capable of receiving messages. The LDAP-specific encoding of a value of this syntax is defined by the following ABNF:

```
DeliveryMethod = pdm *( WSP DOLLAR WSP pdm )

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

The <WSP> and <DOLLAR> rules are defined in [RFC4512].

Example:

```
telephone $ videotex
```

The LDAP definition for the Delivery Method syntax is:

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

This syntax corresponds to the following ASN.1 type from [X.520]:

```
SEQUENCE OF INTEGER {
    any-delivery-method     (0),
    mhs-delivery            (1),
    physical-delivery       (2),
    telex-delivery          (3),
    teletex-delivery        (4),
    g3-facsimile-delivery   (5),
    g4-facsimile-delivery   (6),
    ia5-terminal-delivery   (7),
    videotex-delivery       (8),
    telephone-delivery      (9) }
```

3.3.6. Directory String

A value of the Directory String syntax is a string of one or more arbitrary characters from the Universal Character Set (UCS) [UCS]. A zero-length character string is not permitted. The LDAP-specific encoding of a value of this syntax is the UTF-8 encoding [RFC3629] of the character string. Such encodings conform to the following ABNF:

```
DirectoryString = 1*UTF8
```

The <UTF8> rule is defined in [RFC4512].

Example:
This is a value of Directory String containing #!%#@.

Servers and clients MUST be prepared to receive arbitrary UCS code points, including code points outside the range of printable ASCII and code points not presently assigned to any character.

Attribute type definitions using the Directory String syntax should not restrict the format of Directory String values, e.g., by requiring that the character string conforms to specific patterns described by ABNF. A new syntax should be defined in such cases.

The LDAP definition for the Directory String syntax is:

(1.3.6.1.4.1.1466.115.121.1.15 DESC 'Directory String')

This syntax corresponds to the DirectoryString parameterized ASN.1 type from [X.520].

The DirectoryString ASN.1 type allows a choice between the TeletexString, PrintableString, or UniversalString ASN.1 types from [ASN.1]. However, note that the chosen alternative is not indicated in the LDAP-specific encoding of a Directory String value.

Implementations that convert Directory String values from the LDAP-specific encoding to the BER encoding used by X.500 must choose an alternative that permits the particular characters in the string and must convert the characters from the UTF-8 encoding into the character encoding of the chosen alternative. When converting Directory String values from the BER encoding to the LDAP-specific encoding, the characters must be converted from the character encoding of the chosen alternative into the UTF-8 encoding. These conversions SHOULD be done in a manner consistent with the Transcode step of the string preparation algorithms [RFC4518] for LDAP.

3.3.7. DIT Content Rule Description

A value of the DIT Content Rule Description syntax is the definition of a DIT (Directory Information Tree) content rule. The LDAP-specific encoding of a value of this syntax is defined by the <DITContentRuleDescription> rule in [RFC4512].

Example:
(2.5.6.4 DESC 'content rule for organization'
NOT (x121Address $ telexNumber))

Note: A line break has been added for readability; it is not part of the value.

The LDAP definition for the DIT Content Rule Description syntax is:

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

This syntax corresponds to the DITContentRuleDescription ASN.1 type from [X.501].

3.3.8. DIT Structure Rule Description

A value of the DIT Structure Rule Description syntax is the definition of a DIT structure rule. The LDAP-specific encoding of a value of this syntax is defined by the <DITStructureRuleDescription> rule in [RFC4512].

Example:

```
( 2 DESC 'organization structure rule' FORM 2.5.15.3 )
```

The LDAP definition for the DIT Structure Rule Description syntax is:

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

This syntax corresponds to the DITStructureRuleDescription ASN.1 type from [X.501].

3.3.9. DN

A value of the DN syntax is the (purported) distinguished name (DN) of an entry [RFC4512]. The LDAP-specific encoding of a value of this syntax is defined by the <distinguishedName> rule from the string representation of distinguished names [RFC4514].

Examples (from [RFC4514]):

```
UID=jsmith,DC=example,DC=net
OU=Sales+CN=J. Smith,DC=example,DC=net
CN=John Smith\, III,DC=example,DC=net
CN=Before\0dAfter,DC=example,DC=net
1.3.6.1.4.1.1466.0=#04024869,DC=example,DC=com
CN=Lu\C4\8Di\C4\87
```

The LDAP definition for the DN syntax is:

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

The DN syntax corresponds to the DistinguishedName ASN.1 type from [X.501]. Note that a BER encoded distinguished name (as used by X.500) re-encoded into the LDAP-specific encoding is not necessarily

reversible to the original BER encoding since the chosen string type in any DirectoryString components of the distinguished name is not indicated in the LDAP-specific encoding of the distinguished name (see Section 3.3.6).

3.3.10. Enhanced Guide

A value of the Enhanced Guide syntax suggests criteria, which consist of combinations of attribute types and filter operators, to be used in constructing filters to search for entries of particular object classes. The Enhanced Guide syntax improves upon the Guide syntax by allowing the recommended depth of the search to be specified.

The LDAP-specific encoding of a value of this syntax is defined by the following ABNF:

```
EnhancedGuide = object-class SHARP WSP criteria WSP
                    SHARP WSP subset
object-class  = WSP oid WSP
subset        = "baseobject" / "oneLevel" / "wholeSubtree"

criteria   = and-term *( BAR and-term )
and-term   = term *( AMPERSAND term )
term       = EXCLAIM term /
             attributetype DOLLAR match-type /
             LPAREN criteria RPAREN /
             true /
             false
match-type = "EQ" / "SUBSTR" / "GE" / "LE" / "APPROX"
true       = "?true"
false      = "?false"
BAR        = %x7C  ; vertical bar ("|")
AMPERSAND  = %x26  ; ampersand ("&")
EXCLAIM    = %x21  ; exclamation mark ("!")
```

The <SHARP>, <WSP>, <oid>, <LPAREN>, <RPAREN>, <attributetype>, and <DOLLAR> rules are defined in [RFC4512].

The LDAP definition for the Enhanced Guide syntax is:

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

Example:

```
person#(sn$EQ)#oneLevel
```

The Enhanced Guide syntax corresponds to the EnhancedGuide ASN.1 type from [X.520]. The EnhancedGuide type references the Criteria ASN.1 type, also from [X.520]. The <true> rule, above, represents an empty

"and" expression in a value of the Criteria type. The <false> rule, above, represents an empty "or" expression in a value of the Criteria type.

3.3.11. Facsimile Telephone Number

A value of the Facsimile Telephone Number syntax is a subscriber number of a facsimile device on the public switched telephone network. The LDAP-specific encoding of a value of this syntax is defined by the following ABNF:

```
fax-number       = telephone-number *( DOLLAR fax-parameter )
telephone-number = PrintableString
fax-parameter    = "twoDimensional" /
                   "fineResolution" /
                   "unlimitedLength" /
                   "b4Length" /
                   "a3Width" /
                   "b4Width" /
                   "uncompressed"
```

The <telephone-number> is a string of printable characters that complies with the internationally agreed format for representing international telephone numbers [E.123]. The <PrintableString> rule is defined in Section 3.2. The <DOLLAR> rule is defined in [RFC4512].

The LDAP definition for the Facsimile Telephone Number syntax is:

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

The Facsimile Telephone Number syntax corresponds to the FacsimileTelephoneNumber ASN.1 type from [X.520].

3.3.12. Fax

A value of the Fax syntax is an image that is produced using the Group 3 facsimile process [FAX] to duplicate an object, such as a memo. The LDAP-specific encoding of a value of this syntax is the string of octets for a Group 3 Fax image as defined in [FAX].

The LDAP definition for the Fax syntax is:

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

The ASN.1 type corresponding to the Fax syntax is defined as follows, assuming EXPLICIT TAGS:

```
Fax ::= CHOICE {
  g3-facsimile  [3] G3FacsimileBodyPart
}
```

The G3FacsimileBodyPart ASN.1 type is defined in [X.420].

3.3.13. Generalized Time

A value of the Generalized Time syntax is a character string representing a date and time. The LDAP-specific encoding of a value of this syntax is a restriction of the format defined in [ISO8601], and is described by the following ABNF:

```
GeneralizedTime = century year month day hour
                     [ minute [ second / leap-second ] ]
                     [ fraction ]
                     g-time-zone

century = 2(%x30-39) ; "00" to "99"
year    = 2(%x30-39) ; "00" to "99"
month   =   ( %x30 %x31-39 ) ; "01" (January) to "09"
          / ( %x31 %x30-32 ) ; "10" to "12"
day     =   ( %x30 %x31-39 )    ; "01" to "09"
          / ( %x31-32 %x30-39 ) ; "10" to "29"
          / ( %x33 %x30-31 )    ; "30" to "31"
hour    = ( %x30-31 %x30-39 ) / ( %x32 %x30-33 ) ; "00" to "23"
minute  = %x30-35 %x30-39                        ; "00" to "59"

second      = ( %x30-35 %x30-39 ) ; "00" to "59"
leap-second = ( %x36 %x30 )       ; "60"

fraction        = ( DOT / COMMA ) 1*(%x30-39)
g-time-zone     = %x5A  ; "Z"
                  / g-differential
g-differential  = ( MINUS / PLUS ) hour [ minute ]
MINUS           = %x2D  ; minus sign ("-")
```

The <DOT>, <COMMA>, and <PLUS> rules are defined in [RFC4512].

The above ABNF allows character strings that do not represent valid dates (in the Gregorian calendar) and/or valid times (e.g., February 31, 1994). Such character strings SHOULD be considered invalid for this syntax.

The time value represents coordinated universal time (equivalent to Greenwich Mean Time) if the "Z" form of <g-time-zone> is used; otherwise, the value represents a local time in the time zone indicated by <g-differential>. In the latter case, coordinated

universal time can be calculated by subtracting the differential from the local time. The "Z" form of <g-time-zone> SHOULD be used in preference to <g-differential>.

If <minute> is omitted, then <fraction> represents a fraction of an hour; otherwise, if <second> and <leap-second> are omitted, then <fraction> represents a fraction of a minute; otherwise, <fraction> represents a fraction of a second.

Examples:
199412161032Z
199412160532-0500

Both example values represent the same coordinated universal time: 10:32 AM, December 16, 1994.

The LDAP definition for the Generalized Time syntax is:

(1.3.6.1.4.1.1466.115.121.1.24 DESC 'Generalized Time')

This syntax corresponds to the GeneralizedTime ASN.1 type from [ASN.1], with the constraint that local time without a differential SHALL NOT be used.

3.3.14. Guide

A value of the Guide syntax suggests criteria, which consist of combinations of attribute types and filter operators, to be used in constructing filters to search for entries of particular object classes. The Guide syntax is obsolete and should not be used for defining new attribute types.

The LDAP-specific encoding of a value of this syntax is defined by the following ABNF:

Guide = [object-class SHARP] criteria

The <object-class> and <criteria> rules are defined in Section 3.3.10. The <SHARP> rule is defined in [RFC4512].

The LDAP definition for the Guide syntax is:

(1.3.6.1.4.1.1466.115.121.1.25 DESC 'Guide')

The Guide syntax corresponds to the Guide ASN.1 type from [X.520].

3.3.15. IA5 String

A value of the IA5 String syntax is a string of zero, one, or more characters from International Alphabet 5 (IA5) [T.50], the international version of the ASCII character set. The LDAP-specific encoding of a value of this syntax is the unconverted string of characters, which conforms to the <IA5String> rule in Section 3.2.

The LDAP definition for the IA5 String syntax is:

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

This syntax corresponds to the IA5String ASN.1 type from [ASN.1].

3.3.16. Integer

A value of the Integer syntax is a whole number of unlimited magnitude. The LDAP-specific encoding of a value of this syntax is the optionally signed decimal digit character string representation of the number (for example, the number 1321 is represented by the character string "1321"). The encoding is defined by the following ABNF:

```
Integer = ( HYPHEN LDIGIT *DIGIT ) / number
```

The <HYPHEN>, <LDIGIT>, <DIGIT>, and <number> rules are defined in [RFC4512].

The LDAP definition for the Integer syntax is:

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

This syntax corresponds to the INTEGER ASN.1 type from [ASN.1].

3.3.17. JPEG

A value of the JPEG syntax is an image in the JPEG File Interchange Format (JFIF), as described in [JPEG]. The LDAP-specific encoding of a value of this syntax is the sequence of octets of the JFIF encoding of the image.

The LDAP definition for the JPEG syntax is:

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

The JPEG syntax corresponds to the following ASN.1 type:

```
JPEG ::= OCTET STRING (CONSTRAINED BY
             { -- contents octets are an image in the --
               -- JPEG File Interchange Format -- })
```

3.3.18. LDAP Syntax Description

A value of the LDAP Syntax Description syntax is the description of an LDAP syntax. The LDAP-specific encoding of a value of this syntax is defined by the <SyntaxDescription> rule in [RFC4512].

The LDAP definition for the LDAP Syntax Description syntax is:

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

The above LDAP definition for the LDAP Syntax Description syntax is itself a legal value of the LDAP Syntax Description syntax.

The ASN.1 type corresponding to the LDAP Syntax Description syntax is defined as follows, assuming EXPLICIT TAGS:

```
LDAPSyntaxDescription ::= SEQUENCE {
    identifier      OBJECT IDENTIFIER,
    description     DirectoryString { ub-schema } OPTIONAL }
```

The DirectoryString parameterized ASN.1 type is defined in [X.520].

The value of ub-schema (an integer) is implementation defined. A non-normative definition appears in [X.520].

3.3.19. Matching Rule Description

A value of the Matching Rule Description syntax is the definition of a matching rule. The LDAP-specific encoding of a value of this syntax is defined by the <MatchingRuleDescription> rule in [RFC4512].

```
Example:
   ( 2.5.13.2 NAME 'caseIgnoreMatch'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

Note: A line break has been added for readability; it is not part of the syntax.

The LDAP definition for the Matching Rule Description syntax is:

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

This syntax corresponds to the MatchingRuleDescription ASN.1 type from [X.501].

3.3.20. Matching Rule Use Description

A value of the Matching Rule Use Description syntax indicates the attribute types to which a matching rule may be applied in an extensibleMatch search filter [RFC4511]. The LDAP-specific encoding of a value of this syntax is defined by the <MatchingRuleUseDescription> rule in [RFC4512].

Example:
(2.5.13.16 APPLIES (givenName $ surname))

The LDAP definition for the Matching Rule Use Description syntax is:

(1.3.6.1.4.1.1466.115.121.1.31
DESC 'Matching Rule Use Description')

This syntax corresponds to the MatchingRuleUseDescription ASN.1 type from [X.501].

3.3.21. Name and Optional UID

A value of the Name and Optional UID syntax is the distinguished name [RFC4512] of an entity optionally accompanied by a unique identifier that serves to differentiate the entity from others with an identical distinguished name.

The LDAP-specific encoding of a value of this syntax is defined by the following ABNF:

NameAndOptionalUID = distinguishedName [SHARP BitString]

The <BitString> rule is defined in Section 3.3.2. The <distinguishedName> rule is defined in [RFC4514]. The <SHARP> rule is defined in [RFC4512].

Note that although the '#' character may occur in the string representation of a distinguished name, no additional escaping of this character is performed when a <distinguishedName> is encoded in a <NameAndOptionalUID>.

Example:
1.3.6.1.4.1.1466.0=#04024869,O=Test,C=GB#'0101'B

The LDAP definition for the Name and Optional UID syntax is:

(1.3.6.1.4.1.1466.115.121.1.34 DESC 'Name And Optional UID')

This syntax corresponds to the NameAndOptionalUID ASN.1 type from [X.520].

3.3.22. Name Form Description

A value of the Name Form Description syntax is the definition of a name form, which regulates how entries may be named. The LDAP-specific encoding of a value of this syntax is defined by the <NameFormDescription> rule in [RFC4512].

Example:
(2.5.15.3 NAME 'orgNameForm' OC organization MUST o)

The LDAP definition for the Name Form Description syntax is:

(1.3.6.1.4.1.1466.115.121.1.35 DESC 'Name Form Description')

This syntax corresponds to the NameFormDescription ASN.1 type from [X.501].

3.3.23. Numeric String

A value of the Numeric String syntax is a sequence of one or more numerals and spaces. The LDAP-specific encoding of a value of this syntax is the unconverted string of characters, which conforms to the following ABNF:

NumericString = 1*(DIGIT / SPACE)

The <DIGIT> and <SPACE> rules are defined in [RFC4512].

Example:
15 079 672 281

The LDAP definition for the Numeric String syntax is:

(1.3.6.1.4.1.1466.115.121.1.36 DESC 'Numeric String')

This syntax corresponds to the NumericString ASN.1 type from [ASN.1].

3.3.24. Object Class Description

A value of the Object Class Description syntax is the definition of an object class. The LDAP-specific encoding of a value of this syntax is defined by the <ObjectClassDescription> rule in [RFC4512].

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

Note: A line break has been added for readability; it is not part of the syntax.

The LDAP definition for the Object Class Description syntax is:

(1.3.6.1.4.1.1466.115.121.1.37 DESC 'Object Class Description')

This syntax corresponds to the ObjectClassDescription ASN.1 type from [X.501].

3.3.25. Octet String

A value of the Octet String syntax is a sequence of zero, one, or more arbitrary octets. The LDAP-specific encoding of a value of this syntax is the unconverted sequence of octets, which conforms to the following ABNF:

OctetString = *OCTET

The <OCTET> rule is defined in [RFC4512]. Values of this syntax are not generally human-readable.

The LDAP definition for the Octet String syntax is:

(1.3.6.1.4.1.1466.115.121.1.40 DESC 'Octet String')

This syntax corresponds to the OCTET STRING ASN.1 type from [ASN.1].

3.3.26. OID

A value of the OID syntax is an object identifier: a sequence of two or more non-negative integers that uniquely identify some object or item of specification. Many of the object identifiers used in LDAP also have IANA registered names [RFC4520].

The LDAP-specific encoding of a value of this syntax is defined by the <oid> rule in [RFC4512].

Examples:
1.2.3.4
cn

The LDAP definition for the OID syntax is:

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

This syntax corresponds to the OBJECT IDENTIFIER ASN.1 type from [ASN.1].

3.3.27. Other Mailbox

A value of the Other Mailbox syntax identifies an electronic mailbox, in a particular named mail system. The LDAP-specific encoding of a value of this syntax is defined by the following ABNF:

```
OtherMailbox = mailbox-type DOLLAR mailbox
mailbox-type = PrintableString
mailbox      = IA5String
```

The <mailbox-type> rule represents the type of mail system in which the mailbox resides (for example, "MCIMail"), and <mailbox> is the actual mailbox in the mail system described by <mailbox-type>. The <PrintableString> and <IA5String> rules are defined in Section 3.2. The <DOLLAR> rule is defined in [RFC4512].

The LDAP definition for the Other Mailbox syntax is:

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

The ASN.1 type corresponding to the Other Mailbox syntax is defined as follows, assuming EXPLICIT TAGS:

```
OtherMailbox ::= SEQUENCE {
    mailboxType  PrintableString,
    mailbox      IA5String
}
```

3.3.28. Postal Address

A value of the Postal Address syntax is a sequence of strings of one or more arbitrary UCS characters, which form an address in a physical mail system.

The LDAP-specific encoding of a value of this syntax is defined by the following ABNF:

```
PostalAddress = line *( DOLLAR line )
line          = 1*line-char
line-char     = %x00-23
                / (%x5C "24")  ; escaped "$"
                / %x25-5B
                / (%x5C "5C")  ; escaped "\"
                / %x5D-7F
                / UTFMB
```

Each character string (i.e., <line>) of a postal address value is encoded as a UTF-8 [RFC3629] string, except that "\" and "$" characters, if they occur in the string, are escaped by a "\" character followed by the two hexadecimal digit code for the character. The <DOLLAR> and <UTFMB> rules are defined in [RFC4512].

Many servers limit the postal address to no more than six lines of no more than thirty characters each.

Example:

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

The LDAP definition for the Postal Address syntax is:

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

This syntax corresponds to the PostalAddress ASN.1 type from [X.520]; that is

```
PostalAddress ::= SEQUENCE SIZE(1..ub-postal-line) OF
    DirectoryString { ub-postal-string }
```

The values of ub-postal-line and ub-postal-string (both integers) are implementation defined. Non-normative definitions appear in [X.520].

3.3.29. Printable String

A value of the Printable String syntax is a string of one or more latin alphabetic, numeric, and selected punctuation characters as specified by the <PrintableCharacter> rule in Section 3.2.

The LDAP-specific encoding of a value of this syntax is the unconverted string of characters, which conforms to the <PrintableString> rule in Section 3.2.

Example:

```
This is a PrintableString.
```

The LDAP definition for the PrintableString syntax is:

(1.3.6.1.4.1.1466.115.121.1.44 DESC 'Printable String')

This syntax corresponds to the PrintableString ASN.1 type from [ASN.1].

3.3.30. Substring Assertion

A value of the Substring Assertion syntax is a sequence of zero, one, or more character substrings used as an argument for substring extensible matching of character string attribute values; i.e., as the matchValue of a MatchingRuleAssertion [RFC4511]. Each substring is a string of one or more arbitrary characters from the Universal Character Set (UCS) [UCS]. A zero-length substring is not permitted.

The LDAP-specific encoding of a value of this syntax is defined by the following ABNF:

```
SubstringAssertion = [ initial ] any [ final ]

initial  = substring
any      = ASTERISK *(substring ASTERISK)
final    = substring
ASTERISK = %x2A  ; asterisk ("*")

substring           = 1*substring-character
substring-character = %x00-29
                      / (%x5C "2A")  ; escaped "*"
                      / %x2B-5B
                      / (%x5C "5C")  ; escaped "\"
                      / %x5D-7F
                      / UTFMB
```

Each <substring> of a Substring Assertion value is encoded as a UTF-8 [RFC3629] string, except that "\" and "*" characters, if they occur in the substring, are escaped by a "\" character followed by the two hexadecimal digit code for the character.

The Substring Assertion syntax is used only as the syntax of assertion values in the extensible match. It is not used as an attribute syntax, or in the SubstringFilter [RFC4511].

The LDAP definition for the Substring Assertion syntax is:

(1.3.6.1.4.1.1466.115.121.1.58 DESC 'Substring Assertion')

This syntax corresponds to the SubstringAssertion ASN.1 type from [X.520].

3.3.31. Telephone Number

A value of the Telephone Number syntax is a string of printable characters that complies with the internationally agreed format for representing international telephone numbers [E.123].

The LDAP-specific encoding of a value of this syntax is the unconverted string of characters, which conforms to the <PrintableString> rule in Section 3.2.

Examples:

```
+1 512 315 0280
+1-512-315-0280
+61 3 9896 7830
```

The LDAP definition for the Telephone Number syntax is:

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

The Telephone Number syntax corresponds to the following ASN.1 type from [X.520]:

```
PrintableString (SIZE(1..ub-telephone-number))
```

The value of ub-telephone-number (an integer) is implementation defined. A non-normative definition appears in [X.520].

3.3.32. Teletex Terminal Identifier

A value of this syntax specifies the identifier and (optionally) parameters of a teletex terminal.

The LDAP-specific encoding of a value of this syntax is defined by the following ABNF:

```
teletex-id = ttx-term *(DOLLAR ttx-param)
ttx-term   = PrintableString          ; terminal identifier
ttx-param  = ttx-key COLON ttx-value  ; parameter
ttx-key    = "graphic" / "control" / "misc" / "page" / "private"
ttx-value  = *ttx-value-octet

ttx-value-octet = %x00-23
                  / (%x5C "24")  ; escaped "$"
                  / %x25-5B
                  / (%x5C "5C")  ; escaped "\"
```

```
                        / %x5D-FF
```

The <PrintableString> and <COLON> rules are defined in Section 3.2. The <DOLLAR> rule is defined in [RFC4512].

The LDAP definition for the Teletex Terminal Identifier syntax is:

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

This syntax corresponds to the TeletexTerminalIdentifier ASN.1 type from [X.520].

3.3.33. Telex Number

A value of the Telex Number syntax specifies the telex number, country code, and answerback code of a telex terminal.

The LDAP-specific encoding of a value of this syntax is defined by the following ABNF:

```
telex-number  = actual-number DOLLAR country-code
                    DOLLAR answerback
actual-number = PrintableString
country-code  = PrintableString
answerback    = PrintableString
```

The <PrintableString> rule is defined in Section 3.2. The <DOLLAR> rule is defined in [RFC4512].

The LDAP definition for the Telex Number syntax is:

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

This syntax corresponds to the TelexNumber ASN.1 type from [X.520].

3.3.34. UTC Time

A value of the UTC Time syntax is a character string representing a date and time to a precision of one minute or one second. The year is given as a two-digit number. The LDAP-specific encoding of a value of this syntax follows the format defined in [ASN.1] for the UTCTime type and is described by the following ABNF:

```
UTCTime         = year month day hour minute [ second ]
                      [ u-time-zone ]
u-time-zone     = %x5A  ; "Z"
                  / u-differential
```

```
u-differential  = ( MINUS / PLUS ) hour minute
```

The <year>, <month>, <day>, <hour>, <minute>, <second>, and <MINUS> rules are defined in Section 3.3.13. The <PLUS> rule is defined in [RFC4512].

The above ABNF allows character strings that do not represent valid dates (in the Gregorian calendar) and/or valid times. Such character strings SHOULD be considered invalid for this syntax.

The time value represents coordinated universal time if the "Z" form of <u-time-zone> is used; otherwise, the value represents a local time. In the latter case, if <u-differential> is provided, then coordinated universal time can be calculated by subtracting the differential from the local time. The <u-time-zone> SHOULD be present in time values, and the "Z" form of <u-time-zone> SHOULD be used in preference to <u-differential>.

The LDAP definition for the UTC Time syntax is:

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

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

The UTC Time syntax corresponds to the UTCTime ASN.1 type from [ASN.1].

4. Matching Rules

Matching rules are used by directory implementations to compare attribute values against assertion values when performing Search and Compare operations [RFC4511]. They are also used when comparing a purported distinguished name [RFC4512] with the name of an entry. When modifying entries, matching rules are used to identify values to be deleted and to prevent an attribute from containing two equal values.

Matching rules that are required for directory operation, or that are in common use, are specified in this section.

4.1. General Considerations

A matching rule is applied to attribute values through an AttributeValueAssertion or MatchingRuleAssertion [RFC4511]. The conditions under which an AttributeValueAssertion or MatchingRuleAssertion evaluates to Undefined are specified elsewhere [RFC4511]. If an assertion is not Undefined, then the result of the

assertion is the result of applying the selected matching rule. A matching rule evaluates to TRUE, and in some cases Undefined, as specified in the description of the matching rule; otherwise, it evaluates to FALSE.

Each assertion contains an assertion value. The definition of each matching rule specifies the syntax for the assertion value. The syntax of the assertion value is typically, but not necessarily, the same as the syntax of the attribute values to which the matching rule may be applied. Note that an AssertionValue in a SubstringFilter [RFC4511] conforms to the assertion syntax of the equality matching rule for the attribute type rather than to the assertion syntax of the substrings matching rule for the attribute type. Conceptually, the entire SubstringFilter is converted into an assertion value of the substrings matching rule prior to applying the rule.

The definition of each matching rule indicates the attribute syntaxes to which the rule may be applied, by specifying conditions the corresponding ASN.1 type of a candidate attribute syntax must satisfy. These conditions are also satisfied if the corresponding ASN.1 type is a tagged or constrained derivative of the ASN.1 type explicitly mentioned in the rule description (i.e., ASN.1 tags and constraints are ignored in checking applicability), or is an alternative reference notation for the explicitly mentioned type. Each rule description lists, as examples of applicable attribute syntaxes, the complete list of the syntaxes defined in this document to which the matching rule applies. A matching rule may be applicable to additional syntaxes defined in other documents if those syntaxes satisfy the conditions on the corresponding ASN.1 type.

The description of each matching rule indicates whether the rule is suitable for use as the equality matching rule (EQUALITY), ordering matching rule (ORDERING), or substrings matching rule (SUBSTR) in an attribute type definition [RFC4512].

Each matching rule is uniquely identified with an object identifier. The definition of a matching rule should not subsequently be changed. If a change is desirable, then a new matching rule with a different object identifier should be defined instead.

Servers MAY implement the wordMatch and keywordMatch matching rules, but they SHOULD implement the other matching rules in Section 4.2. Servers MAY implement additional matching rules.

Servers that implement the extensibleMatch filter SHOULD allow the matching rules listed in Section 4.2 to be used in the extensibleMatch filter and SHOULD allow matching rules to be used with all attribute types known to the server, where the assertion

syntax of the matching rule is the same as the value syntax of the attribute.

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 filter, then the server MAY use the matchingRuleUse attribute to indicate the applicability (in an extensibleMatch filter) of selected matching rules to nominated attribute types.

4.2. Matching Rule Definitions

Nominated character strings in assertion and attribute values are prepared according to the string preparation algorithms [RFC4518] for LDAP when evaluating the following matching rules:

numericStringMatch,
numericStringSubstringsMatch,
caseExactMatch,
caseExactOrderingMatch,
caseExactSubstringsMatch,
caseExactIA5Match,
caseIgnoreIA5Match,
caseIgnoreIA5SubstringsMatch,
caseIgnoreListMatch,
caseIgnoreListSubstringsMatch,
caseIgnoreMatch,
caseIgnoreOrderingMatch,
caseIgnoreSubstringsMatch,
directoryStringFirstComponentMatch,
telephoneNumberMatch,
telephoneNumberSubstringsMatch and
wordMatch.

The Transcode, Normalize, Prohibit, and Check bidi steps are the same for each of the matching rules. However, the Map and Insignificant Character Handling steps depend on the specific rule, as detailed in the description of these matching rules in the sections that follow.

4.2.1. bitStringMatch

The bitStringMatch rule compares an assertion value of the Bit String syntax to an attribute value of a syntax (e.g., the Bit String syntax) whose corresponding ASN.1 type is BIT STRING.

If the corresponding ASN.1 type of the attribute syntax does not have a named bit list [ASN.1] (which is the case for the Bit String syntax), then the rule evaluates to TRUE if and only if the attribute value has the same number of bits as the assertion value and the bits match on a bitwise basis.

If the corresponding ASN.1 type does have a named bit list, then bitStringMatch operates as above, except that trailing zero bits in the attribute and assertion values are treated as absent.

The LDAP definition for the bitStringMatch rule is:

```
( 2.5.13.16 NAME 'bitStringMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.6 )
```

The bitStringMatch rule is an equality matching rule.

4.2.2. booleanMatch

The booleanMatch rule compares an assertion value of the Boolean syntax to an attribute value of a syntax (e.g., the Boolean syntax) whose corresponding ASN.1 type is BOOLEAN.

The rule evaluates to TRUE if and only if the attribute value and the assertion value are both TRUE or both FALSE.

The LDAP definition for the booleanMatch rule is:

```
( 2.5.13.13 NAME 'booleanMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )
```

The booleanMatch rule is an equality matching rule.

4.2.3. caseExactIA5Match

The caseExactIA5Match rule compares an assertion value of the IA5 String syntax to an attribute value of a syntax (e.g., the IA5 String syntax) whose corresponding ASN.1 type is IA5String.

The rule evaluates to TRUE if and only if the prepared attribute value character string and the prepared assertion value character string have the same number of characters and corresponding characters have the same code point.

In preparing the attribute value and assertion value for comparison, characters are not case folded in the Map preparation step, and only Insignificant Space Handling is applied in the Insignificant Character Handling step.

The LDAP definition for the caseExactIA5Match rule is:

```
( 1.3.6.1.4.1.1466.109.114.1 NAME 'caseExactIA5Match'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

The caseExactIA5Match rule is an equality matching rule.

4.2.4. caseExactMatch

The caseExactMatch rule compares an assertion value of the Directory String syntax to an attribute value of a syntax (e.g., the Directory String, Printable String, Country String, or Telephone Number syntax) whose corresponding ASN.1 type is DirectoryString or one of the alternative string types of DirectoryString, such as PrintableString (the other alternatives do not correspond to any syntax defined in this document).

The rule evaluates to TRUE if and only if the prepared attribute value character string and the prepared assertion value character string have the same number of characters and corresponding characters have the same code point.

In preparing the attribute value and assertion value for comparison, characters are not case folded in the Map preparation step, and only Insignificant Space Handling is applied in the Insignificant Character Handling step.

The LDAP definition for the caseExactMatch rule is:

```
( 2.5.13.5 NAME 'caseExactMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

The caseExactMatch rule is an equality matching rule.

4.2.5. caseExactOrderingMatch

The caseExactOrderingMatch rule compares an assertion value of the Directory String syntax to an attribute value of a syntax (e.g., the Directory String, Printable String, Country String, or Telephone Number syntax) whose corresponding ASN.1 type is DirectoryString or one of its alternative string types.

The rule evaluates to TRUE if and only if, in the code point collation order, the prepared attribute value character string appears earlier than the prepared assertion value character string; i.e., the attribute value is "less than" the assertion value.

In preparing the attribute value and assertion value for comparison, characters are not case folded in the Map preparation step, and only Insignificant Space Handling is applied in the Insignificant Character Handling step.

The LDAP definition for the caseExactOrderingMatch rule is:

```
( 2.5.13.6 NAME 'caseExactOrderingMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

The caseExactOrderingMatch rule is an ordering matching rule.

4.2.6. caseExactSubstringsMatch

The caseExactSubstringsMatch rule compares an assertion value of the Substring Assertion syntax to an attribute value of a syntax (e.g., the Directory String, Printable String, Country String, or Telephone Number syntax) whose corresponding ASN.1 type is DirectoryString or one of its alternative string types.

The rule evaluates to TRUE if and only if (1) the prepared substrings of the assertion value match disjoint portions of the prepared attribute value character string in the order of the substrings in the assertion value, (2) an <initial> substring, if present, matches the beginning of the prepared attribute value character string, and (3) a <final> substring, if present, matches the end of the prepared attribute value character string. A prepared substring matches a portion of the prepared attribute value character string if corresponding characters have the same code point.

In preparing the attribute value and assertion value substrings for comparison, characters are not case folded in the Map preparation step, and only Insignificant Space Handling is applied in the Insignificant Character Handling step.

The LDAP definition for the caseExactSubstringsMatch rule is:

```
( 2.5.13.7 NAME 'caseExactSubstringsMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.58 )
```

The caseExactSubstringsMatch rule is a substrings matching rule.

4.2.7. caseIgnoreIA5Match

The caseIgnoreIA5Match rule compares an assertion value of the IA5 String syntax to an attribute value of a syntax (e.g., the IA5 String syntax) whose corresponding ASN.1 type is IA5String.

The rule evaluates to TRUE if and only if the prepared attribute value character string and the prepared assertion value character string have the same number of characters and corresponding characters have the same code point.

In preparing the attribute value and assertion value for comparison, characters are case folded in the Map preparation step, and only Insignificant Space Handling is applied in the Insignificant Character Handling step.

The LDAP definition for the caseIgnoreIA5Match rule is:

```
( 1.3.6.1.4.1.1466.109.114.2 NAME 'caseIgnoreIA5Match'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

The caseIgnoreIA5Match rule is an equality matching rule.

4.2.8. caseIgnoreIA5SubstringsMatch

The caseIgnoreIA5SubstringsMatch rule compares an assertion value of the Substring Assertion syntax to an attribute value of a syntax (e.g., the IA5 String syntax) whose corresponding ASN.1 type is IA5String.

The rule evaluates to TRUE if and only if (1) the prepared substrings of the assertion value match disjoint portions of the prepared attribute value character string in the order of the substrings in the assertion value, (2) an <initial> substring, if present, matches the beginning of the prepared attribute value character string, and (3) a <final> substring, if present, matches the end of the prepared attribute value character string. A prepared substring matches a portion of the prepared attribute value character string if corresponding characters have the same code point.

In preparing the attribute value and assertion value substrings for comparison, characters are case folded in the Map preparation step, and only Insignificant Space Handling is applied in the Insignificant Character Handling step.

```
( 1.3.6.1.4.1.1466.109.114.3 NAME 'caseIgnoreIA5SubstringsMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.58 )
```

The caseIgnoreIA5SubstringsMatch rule is a substrings matching rule.

4.2.9. caseIgnoreListMatch

The caseIgnoreListMatch rule compares an assertion value that is a sequence of strings to an attribute value of a syntax (e.g., the

Postal Address syntax) whose corresponding ASN.1 type is a SEQUENCE OF the DirectoryString ASN.1 type.

The rule evaluates to TRUE if and only if the attribute value and the assertion value have the same number of strings and corresponding strings (by position) match according to the caseIgnoreMatch matching rule.

In [X.520], the assertion syntax for this matching rule is defined to be:

```
SEQUENCE OF DirectoryString {ub-match}
```

That is, it is different from the corresponding type for the Postal Address syntax. The choice of the Postal Address syntax for the assertion syntax of the caseIgnoreListMatch in LDAP should not be seen as limiting the matching rule to apply only to attributes with the Postal Address syntax.

The LDAP definition for the caseIgnoreListMatch rule is:

```
( 2.5.13.11 NAME 'caseIgnoreListMatch'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.41 )
```

The caseIgnoreListMatch rule is an equality matching rule.

4.2.10. caseIgnoreListSubstringsMatch

The caseIgnoreListSubstringsMatch rule compares an assertion value of the Substring Assertion syntax to an attribute value of a syntax (e.g., the Postal Address syntax) whose corresponding ASN.1 type is a SEQUENCE OF the DirectoryString ASN.1 type.

The rule evaluates to TRUE if and only if the assertion value matches, per the caseIgnoreSubstringsMatch rule, the character string formed by concatenating the strings of the attribute value, except that none of the <initial>, <any>, or <final> substrings of the assertion value are considered to match a substring of the concatenated string which spans more than one of the original strings of the attribute value.

Note that, in terms of the LDAP-specific encoding of the Postal Address syntax, the concatenated string omits the <DOLLAR> line separator and the escaping of "\" and "$" characters.

The LDAP definition for the caseIgnoreListSubstringsMatch rule is:

```
( 2.5.13.12 NAME 'caseIgnoreListSubstringsMatch'
```

```
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.58 )
```

The caseIgnoreListSubstringsMatch rule is a substrings matching rule.

4.2.11. caseIgnoreMatch

The caseIgnoreMatch rule compares an assertion value of the Directory String syntax to an attribute value of a syntax (e.g., the Directory String, Printable String, Country String, or Telephone Number syntax) whose corresponding ASN.1 type is DirectoryString or one of its alternative string types.

The rule evaluates to TRUE if and only if the prepared attribute value character string and the prepared assertion value character string have the same number of characters and corresponding characters have the same code point.

In preparing the attribute value and assertion value for comparison, characters are case folded in the Map preparation step, and only Insignificant Space Handling is applied in the Insignificant Character Handling step.

The LDAP definition for the caseIgnoreMatch rule is:

```
   ( 2.5.13.2 NAME 'caseIgnoreMatch'
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

The caseIgnoreMatch rule is an equality matching rule.

4.2.12. caseIgnoreOrderingMatch

The caseIgnoreOrderingMatch rule compares an assertion value of the Directory String syntax to an attribute value of a syntax (e.g., the Directory String, Printable String, Country String, or Telephone Number syntax) whose corresponding ASN.1 type is DirectoryString or one of its alternative string types.

The rule evaluates to TRUE if and only if, in the code point collation order, the prepared attribute value character string appears earlier than the prepared assertion value character string; i.e., the attribute value is "less than" the assertion value.

In preparing the attribute value and assertion value for comparison, characters are case folded in the Map preparation step, and only Insignificant Space Handling is applied in the Insignificant Character Handling step.

The LDAP definition for the caseIgnoreOrderingMatch rule is:

```
( 2.5.13.3 NAME 'caseIgnoreOrderingMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

The caseIgnoreOrderingMatch rule is an ordering matching rule.

4.2.13. caseIgnoreSubstringsMatch

The caseIgnoreSubstringsMatch rule compares an assertion value of the Substring Assertion syntax to an attribute value of a syntax (e.g., the Directory String, Printable String, Country String, or Telephone Number syntax) whose corresponding ASN.1 type is DirectoryString or one of its alternative string types.

The rule evaluates to TRUE if and only if (1) the prepared substrings of the assertion value match disjoint portions of the prepared attribute value character string in the order of the substrings in the assertion value, (2) an <initial> substring, if present, matches the beginning of the prepared attribute value character string, and (3) a <final> substring, if present, matches the end of the prepared attribute value character string. A prepared substring matches a portion of the prepared attribute value character string if corresponding characters have the same code point.

In preparing the attribute value and assertion value substrings for comparison, characters are case folded in the Map preparation step, and only Insignificant Space Handling is applied in the Insignificant Character Handling step.

The LDAP definition for the caseIgnoreSubstringsMatch rule is:

```
( 2.5.13.4 NAME 'caseIgnoreSubstringsMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.58 )
```

The caseIgnoreSubstringsMatch rule is a substrings matching rule.

4.2.14. directoryStringFirstComponentMatch

The directoryStringFirstComponentMatch rule compares an assertion value of the Directory String syntax to an attribute value of a syntax whose corresponding ASN.1 type is a SEQUENCE with a mandatory first component of the DirectoryString ASN.1 type.

Note that the assertion syntax of this matching rule differs from the attribute syntax of attributes for which this is the equality matching rule.

The rule evaluates to TRUE if and only if the assertion value matches the first component of the attribute value using the rules of caseIgnoreMatch.

The LDAP definition for the directoryStringFirstComponentMatch matching rule is:

```
( 2.5.13.31 NAME 'directoryStringFirstComponentMatch'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

The directoryStringFirstComponentMatch rule is an equality matching rule. When using directoryStringFirstComponentMatch to compare two attribute values (of an applicable syntax), an assertion value must first be derived from one of the attribute values. An assertion value can be derived from an attribute value by taking the first component of that attribute value.

4.2.15. distinguishedNameMatch

The distinguishedNameMatch rule compares an assertion value of the DN syntax to an attribute value of a syntax (e.g., the DN syntax) whose corresponding ASN.1 type is DistinguishedName.

The rule evaluates to TRUE if and only if the attribute value and the assertion value have the same number of relative distinguished names and corresponding relative distinguished names (by position) are the same. A relative distinguished name (RDN) of the assertion value is the same as an RDN of the attribute value if and only if they have the same number of attribute value assertions and each attribute value assertion (AVA) of the first RDN is the same as the AVA of the second RDN with the same attribute type. The order of the AVAs is not significant. Also note that a particular attribute type may appear in at most one AVA in an RDN. Two AVAs with the same attribute type are the same if their values are equal according to the equality matching rule of the attribute type. If one or more of the AVA comparisons evaluate to Undefined and the remaining AVA comparisons return TRUE then the distinguishedNameMatch rule evaluates to Undefined.

The LDAP definition for the distinguishedNameMatch rule is:

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

The distinguishedNameMatch rule is an equality matching rule.

4.2.16. generalizedTimeMatch

The generalizedTimeMatch rule compares an assertion value of the Generalized Time syntax to an attribute value of a syntax (e.g., the Generalized Time syntax) whose corresponding ASN.1 type is GeneralizedTime.

The rule evaluates to TRUE if and only if the attribute value represents the same universal coordinated time as the assertion value. If a time is specified with the minutes or seconds absent, then the number of minutes or seconds (respectively) is assumed to be zero.

The LDAP definition for the generalizedTimeMatch rule is:

```
( 2.5.13.27 NAME 'generalizedTimeMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )
```

The generalizedTimeMatch rule is an equality matching rule.

4.2.17. generalizedTimeOrderingMatch

The generalizedTimeOrderingMatch rule compares the time ordering of an assertion value of the Generalized Time syntax to an attribute value of a syntax (e.g., the Generalized Time syntax) whose corresponding ASN.1 type is GeneralizedTime.

The rule evaluates to TRUE if and only if the attribute value represents a universal coordinated time that is earlier than the universal coordinated time represented by the assertion value.

The LDAP definition for the generalizedTimeOrderingMatch rule is:

```
( 2.5.13.28 NAME 'generalizedTimeOrderingMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )
```

The generalizedTimeOrderingMatch rule is an ordering matching rule.

4.2.18. integerFirstComponentMatch

The integerFirstComponentMatch rule compares an assertion value of the Integer syntax to an attribute value of a syntax (e.g., the DIT Structure Rule Description syntax) whose corresponding ASN.1 type is a SEQUENCE with a mandatory first component of the INTEGER ASN.1 type.

Note that the assertion syntax of this matching rule differs from the attribute syntax of attributes for which this is the equality matching rule.

The rule evaluates to TRUE if and only if the assertion value and the first component of the attribute value are the same integer value.

The LDAP definition for the integerFirstComponentMatch matching rule is:

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

The integerFirstComponentMatch rule is an equality matching rule. When using integerFirstComponentMatch to compare two attribute values (of an applicable syntax), an assertion value must first be derived from one of the attribute values. An assertion value can be derived from an attribute value by taking the first component of that attribute value.

4.2.19. integerMatch

The integerMatch rule compares an assertion value of the Integer syntax to an attribute value of a syntax (e.g., the Integer syntax) whose corresponding ASN.1 type is INTEGER.

The rule evaluates to TRUE if and only if the attribute value and the assertion value are the same integer value.

The LDAP definition for the integerMatch matching rule is:

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

The integerMatch rule is an equality matching rule.

4.2.20. integerOrderingMatch

The integerOrderingMatch rule compares an assertion value of the Integer syntax to an attribute value of a syntax (e.g., the Integer syntax) whose corresponding ASN.1 type is INTEGER.

The rule evaluates to TRUE if and only if the integer value of the attribute value is less than the integer value of the assertion value.

The LDAP definition for the integerOrderingMatch matching rule is:

```
( 2.5.13.15 NAME 'integerOrderingMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )
```

The integerOrderingMatch rule is an ordering matching rule.

4.2.21. keywordMatch

The keywordMatch rule compares an assertion value of the Directory String syntax to an attribute value of a syntax (e.g., the Directory String syntax) whose corresponding ASN.1 type is DirectoryString.

The rule evaluates to TRUE if and only if the assertion value character string matches any keyword in the attribute value. The identification of keywords in the attribute value and the exactness of the match are both implementation specific.

The LDAP definition for the keywordMatch rule is:

```
( 2.5.13.33 NAME 'keywordMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

4.2.22. numericStringMatch

The numericStringMatch rule compares an assertion value of the Numeric String syntax to an attribute value of a syntax (e.g., the Numeric String syntax) whose corresponding ASN.1 type is NumericString.

The rule evaluates to TRUE if and only if the prepared attribute value character string and the prepared assertion value character string have the same number of characters and corresponding characters have the same code point.

In preparing the attribute value and assertion value for comparison, characters are not case folded in the Map preparation step, and only numericString Insignificant Character Handling is applied in the Insignificant Character Handling step.

The LDAP definition for the numericStringMatch matching rule is:

```
( 2.5.13.8 NAME 'numericStringMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.36 )
```

The numericStringMatch rule is an equality matching rule.

4.2.23. numericStringOrderingMatch

The numericStringOrderingMatch rule compares an assertion value of the Numeric String syntax to an attribute value of a syntax (e.g., the Numeric String syntax) whose corresponding ASN.1 type is NumericString.

The rule evaluates to TRUE if and only if, in the code point collation order, the prepared attribute value character string appears earlier than the prepared assertion value character string; i.e., the attribute value is "less than" the assertion value.

In preparing the attribute value and assertion value for comparison, characters are not case folded in the Map preparation step, and only numericString Insignificant Character Handling is applied in the Insignificant Character Handling step.

The rule is identical to the caseIgnoreOrderingMatch rule except that all space characters are skipped during comparison (case is irrelevant as the characters are numeric).

The LDAP definition for the numericStringOrderingMatch matching rule is:

```
( 2.5.13.9 NAME 'numericStringOrderingMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.36 )
```

The numericStringOrderingMatch rule is an ordering matching rule.

4.2.24. numericStringSubstringsMatch

The numericStringSubstringsMatch rule compares an assertion value of the Substring Assertion syntax to an attribute value of a syntax (e.g., the Numeric String syntax) whose corresponding ASN.1 type is NumericString.

The rule evaluates to TRUE if and only if (1) the prepared substrings of the assertion value match disjoint portions of the prepared attribute value character string in the order of the substrings in the assertion value, (2) an <initial> substring, if present, matches the beginning of the prepared attribute value character string, and (3) a <final> substring, if present, matches the end of the prepared attribute value character string. A prepared substring matches a portion of the prepared attribute value character string if corresponding characters have the same code point.

In preparing the attribute value and assertion value for comparison, characters are not case folded in the Map preparation step, and only

numericString Insignificant Character Handling is applied in the Insignificant Character Handling step.

The LDAP definition for the numericStringSubstringsMatch matching rule is:

```
( 2.5.13.10 NAME 'numericStringSubstringsMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.58 )
```

The numericStringSubstringsMatch rule is a substrings matching rule.

4.2.25. objectIdentifierFirstComponentMatch

The objectIdentifierFirstComponentMatch rule compares an assertion value of the OID syntax to an attribute value of a syntax (e.g., the Attribute Type Description, DIT Content Rule Description, LDAP Syntax Description, Matching Rule Description, Matching Rule Use Description, Name Form Description, or Object Class Description syntax) whose corresponding ASN.1 type is a SEQUENCE with a mandatory first component of the OBJECT IDENTIFIER ASN.1 type.

Note that the assertion syntax of this matching rule differs from the attribute syntax of attributes for which this is the equality matching rule.

The rule evaluates to TRUE if and only if the assertion value matches the first component of the attribute value using the rules of objectIdentifierMatch.

The LDAP definition for the objectIdentifierFirstComponentMatch matching rule is:

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

The objectIdentifierFirstComponentMatch rule is an equality matching rule. When using objectIdentifierFirstComponentMatch to compare two attribute values (of an applicable syntax), an assertion value must first be derived from one of the attribute values. An assertion value can be derived from an attribute value by taking the first component of that attribute value.

4.2.26. objectIdentifierMatch

The objectIdentifierMatch rule compares an assertion value of the OID syntax to an attribute value of a syntax (e.g., the OID syntax) whose corresponding ASN.1 type is OBJECT IDENTIFIER.

The rule evaluates to TRUE if and only if the assertion value and the attribute value represent the same object identifier; that is, the same sequence of integers, whether represented explicitly in the <numericoid> form of <oid> or implicitly in the <descr> form (see [RFC4512]).

If an LDAP client supplies an assertion value in the <descr> form and the chosen descriptor is not recognized by the server, then the objectIdentifierMatch rule evaluates to Undefined.

The LDAP definition for the objectIdentifierMatch matching rule is:

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

The objectIdentifierMatch rule is an equality matching rule.

4.2.27. octetStringMatch

The octetStringMatch rule compares an assertion value of the Octet String syntax to an attribute value of a syntax (e.g., the Octet String or JPEG syntax) whose corresponding ASN.1 type is the OCTET STRING ASN.1 type.

The rule evaluates to TRUE if and only if the attribute value and the assertion value are the same length and corresponding octets (by position) are the same.

The LDAP definition for the octetStringMatch matching rule is:

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

The octetStringMatch rule is an equality matching rule.

4.2.28. octetStringOrderingMatch

The octetStringOrderingMatch rule compares an assertion value of the Octet String syntax to an attribute value of a syntax (e.g., the Octet String or JPEG syntax) whose corresponding ASN.1 type is the OCTET STRING ASN.1 type.

The rule evaluates to TRUE if and only if the attribute value appears earlier in the collation order than the assertion value. The rule compares octet strings from the first octet to the last octet, and from the most significant bit to the least significant bit within the octet. The first occurrence of a different bit determines the ordering of the strings. A zero bit precedes a one bit. If the

strings contain different numbers of octets but the longer string is identical to the shorter string up to the length of the shorter string, then the shorter string precedes the longer string.

The LDAP definition for the octetStringOrderingMatch matching rule is:

```
( 2.5.13.18 NAME 'octetStringOrderingMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 )
```

The octetStringOrderingMatch rule is an ordering matching rule.

4.2.29. telephoneNumberMatch

The telephoneNumberMatch rule compares an assertion value of the Telephone Number syntax to an attribute value of a syntax (e.g., the Telephone Number syntax) whose corresponding ASN.1 type is a PrintableString representing a telephone number.

The rule evaluates to TRUE if and only if the prepared attribute value character string and the prepared assertion value character string have the same number of characters and corresponding characters have the same code point.

In preparing the attribute value and assertion value for comparison, characters are case folded in the Map preparation step, and only telephoneNumber Insignificant Character Handling is applied in the Insignificant Character Handling step.

The LDAP definition for the telephoneNumberMatch matching rule is:

```
( 2.5.13.20 NAME 'telephoneNumberMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.50 )
```

The telephoneNumberMatch rule is an equality matching rule.

4.2.30. telephoneNumberSubstringsMatch

The telephoneNumberSubstringsMatch rule compares an assertion value of the Substring Assertion syntax to an attribute value of a syntax (e.g., the Telephone Number syntax) whose corresponding ASN.1 type is a PrintableString representing a telephone number.

The rule evaluates to TRUE if and only if (1) the prepared substrings of the assertion value match disjoint portions of the prepared attribute value character string in the order of the substrings in the assertion value, (2) an <initial> substring, if present, matches the beginning of the prepared attribute value character string, and

(3) a <final> substring, if present, matches the end of the prepared attribute value character string. A prepared substring matches a portion of the prepared attribute value character string if corresponding characters have the same code point.

In preparing the attribute value and assertion value substrings for comparison, characters are case folded in the Map preparation step, and only telephoneNumber Insignificant Character Handling is applied in the Insignificant Character Handling step.

The LDAP definition for the telephoneNumberSubstringsMatch matching rule is:

```
( 2.5.13.21 NAME 'telephoneNumberSubstringsMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.58 )
```

The telephoneNumberSubstringsMatch rule is a substrings matching rule.

4.2.31. uniqueMemberMatch

The uniqueMemberMatch rule compares an assertion value of the Name And Optional UID syntax to an attribute value of a syntax (e.g., the Name And Optional UID syntax) whose corresponding ASN.1 type is NameAndOptionalUID.

The rule evaluates to TRUE if and only if the <distinguishedName> components of the assertion value and attribute value match according to the distinguishedNameMatch rule and either, (1) the <BitString> component is absent from both the attribute value and assertion value, or (2) the <BitString> component is present in both the attribute value and the assertion value and the <BitString> component of the assertion value matches the <BitString> component of the attribute value according to the bitStringMatch rule.

Note that this matching rule has been altered from its description in X.520 [X.520] in order to make the matching rule commutative. Server implementors should consider using the original X.520 semantics (where the matching was less exact) for approximate matching of attributes with uniqueMemberMatch as the equality matching rule.

The LDAP definition for the uniqueMemberMatch matching rule is:

```
( 2.5.13.23 NAME 'uniqueMemberMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.34 )
```

The uniqueMemberMatch rule is an equality matching rule.

4.2.32. wordMatch

The wordMatch rule compares an assertion value of the Directory String syntax to an attribute value of a syntax (e.g., the Directory String syntax) whose corresponding ASN.1 type is DirectoryString.

The rule evaluates to TRUE if and only if the assertion value word matches, according to the semantics of caseIgnoreMatch, any word in the attribute value. The precise definition of a word is implementation specific.

The LDAP definition for the wordMatch rule is:

```
( 2.5.13.32 NAME 'wordMatch'
   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

5. Security Considerations

In general, the LDAP-specific encodings for syntaxes defined in this document do not define canonical encodings. That is, a transformation from an LDAP-specific encoding into some other encoding (e.g., BER) and back into the LDAP-specific encoding will not necessarily reproduce exactly the original octets of the LDAP-specific encoding. Therefore, an LDAP-specific encoding should not be used where a canonical encoding is required.

Furthermore, the LDAP-specific encodings do not necessarily enable an alternative encoding of values of the Directory String and DN syntaxes to be reconstructed; e.g., a transformation from a Distinguished Encoding Rules (DER) [BER] encoding to an LDAP-specific encoding and back to a DER encoding may not reproduce the original DER encoding. Therefore, LDAP-specific encodings should not be used where reversibility to DER is needed; e.g., for the verification of digital signatures. Instead, DER or a DER-reversible encoding should be used.

When interpreting security-sensitive fields (in particular, fields used to grant or deny access), implementations MUST ensure that any matching rule comparisons are done on the underlying abstract value, regardless of the particular encoding used.

6. Acknowledgements

This document is primarily a revision 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 on input from the IETF LDAPBIS working group. The author would like to thank Kathy Dally for editing the early drafts of this document, and Jim Sermersheim and Kurt Zeilenga for their significant contributions to this revision.

7. IANA Considerations

The Internet Assigned Numbers Authority (IANA) has updated the LDAP descriptors registry [BCP64] as indicated by the following templates:

Subject: Request for LDAP Descriptor Registration Update
Descriptor (short name): see comment
Object Identifier: see comment
Person & email address to contact for further information:
Steven Legg <steven.legg@eb2bcom.com>
Usage: see comment
Specification: RFC 4517
Author/Change Controller: IESG

NAME	Type	OID
bitStringMatch	M	2.5.13.16
booleanMatch	M	2.5.13.13
caseExactIA5Match	M	1.3.6.1.4.1.1466.109.114.1
caseExactMatch	M	2.5.13.5
caseExactOrderingMatch	M	2.5.13.6
caseExactSubstringsMatch	M	2.5.13.7
caseIgnoreIA5Match	M	1.3.6.1.4.1.1466.109.114.2
caseIgnoreListMatch	M	2.5.13.11
caseIgnoreListSubstringsMatch	M	2.5.13.12
caseIgnoreMatch	M	2.5.13.2
caseIgnoreOrderingMatch	M	2.5.13.3
caseIgnoreSubstringsMatch	M	2.5.13.4
directoryStringFirstComponentMatch	M	2.5.13.31
distinguishedNameMatch	M	2.5.13.1
generalizedTimeMatch	M	2.5.13.27
generalizedTimeOrderingMatch	M	2.5.13.28
integerFirstComponentMatch	M	2.5.13.29
integerMatch	M	2.5.13.14
integerOrderingMatch	M	2.5.13.15
keywordMatch	M	2.5.13.33
numericStringMatch	M	2.5.13.8
numericStringOrderingMatch	M	2.5.13.9
numericStringSubstringsMatch	M	2.5.13.10
objectIdentifierFirstComponentMatch	M	2.5.13.30
octetStringMatch	M	2.5.13.17
octetStringOrderingMatch	M	2.5.13.18
telephoneNumberMatch	M	2.5.13.20

```
telephoneNumberSubstringsMatch       M  2.5.13.21
uniqueMemberMatch                    M  2.5.13.23
wordMatch                            M  2.5.13.32
```

The descriptor for the object identifier 2.5.13.0 was incorrectly registered as objectIdentifiersMatch (extraneous \'s') in BCP 64. It has been changed to the following, with a reference to RFC 4517.

```
NAME                                Type  OID
-------------------------------------------------------------------
objectIdentifierMatch                M  2.5.13.0
```

```
Subject: Request for LDAP Descriptor Registration
Descriptor (short name): caseIgnoreIA5SubstringsMatch
Object Identifier: 1.3.6.1.4.1.1466.109.114.3
Person & email address to contact for further information:
  Steven Legg <steven.legg@eb2bcom.com>
Usage: other (M)
Specification: RFC 4517
Author/Change Controller: IESG
```

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC4234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

[RFC4510] Zeilenga, K., Ed., "Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map", RFC 4510, June 2006.

[RFC4511] Sermersheim, J., Ed., "Lightweight Directory Access Protocol (LDAP): The Protocol", RFC 4511, June 2006.

[RFC4512] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): Directory Information Models", RFC 4512, June 2006.

[RFC4514] Zeilenga, K., Ed., "Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names", RFC 4514, June 2006.

[RFC4518] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): Internationalized String Preparation", RFC 4518, June 2006.

[RFC4520] Zeilenga, K., "Internet Assigned Numbers Authority (IANA) Considerations for the Lightweight Directory Access Protocol (LDAP)", BCP 64, RFC 4520, June 2006.

[E.123] Notation for national and international telephone numbers, ITU-T Recommendation E.123, 1988.

[FAX] Standardization of Group 3 facsimile apparatus for document transmission - Terminal Equipment and Protocols for Telematic Services, ITU-T Recommendation T.4, 1993

[T.50] 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

[X.420] ITU-T Recommendation X.420 (1996) | ISO/IEC 10021-7:1997, Information Technology - Message Handling Systems (MHS): Interpersonal messaging system

[X.501] ITU-T Recommendation X.501 (1993) | ISO/IEC 9594-2:1994, Information Technology - Open Systems Interconnection - The Directory: Models

[X.520] ITU-T Recommendation X.520 (1993) | ISO/IEC 9594-6:1994, Information Technology - Open Systems Interconnection - The Directory: Selected attribute types

[ASN.1] ITU-T Recommendation X.680 (07/02) | ISO/IEC 8824-1:2002, Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation

[ISO3166] ISO 3166, "Codes for the representation of names of countries".

[ISO8601] ISO 8601:2004, "Data elements and interchange formats -- Information interchange -- Representation of dates and times".

[UCS] Universal Multiple-Octet Coded Character Set (UCS) - Architecture and Basic Multilingual Plane, ISO/IEC 10646-1: 1993 (with amendments).

[JPEG] JPEG File Interchange Format (Version 1.02). Eric Hamilton, C-Cube Microsystems, Milpitas, CA, September 1, 1992.

8.2. Informative References

[RFC4519] Sciberras, A., Ed., "Lightweight Directory Access Protocol (LDAP): Schema for User Applications", RFC 4519, June 2006.

[RFC4523] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP) Schema Definitions for X.509 Certificates", RFC 4523, June 2006.

[X.500] ITU-T Recommendation X.500 (1993) | ISO/IEC 9594-1:1994, Information Technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services

[BER] ITU-T Recommendation X.690 (07/02) | ISO/IEC 8825-1:2002, Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)

Appendix A. Summary of Syntax Object Identifiers

The following list summarizes the object identifiers assigned to the syntaxes defined in this document.

Syntax	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
Boolean	1.3.6.1.4.1.1466.115.121.1.7
Country String	1.3.6.1.4.1.1466.115.121.1.11
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
DN	1.3.6.1.4.1.1466.115.121.1.12
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
Fax	1.3.6.1.4.1.1466.115.121.1.23
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
JPEG	1.3.6.1.4.1.1466.115.121.1.28
LDAP Syntax Description	1.3.6.1.4.1.1466.115.121.1.54
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
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
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
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

Appendix B. Changes from RFC 2252

This annex lists the significant differences between this specification and RFC 2252.

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

1. The IESG Note has been removed.

2. The major part of Sections 4, 5 and 7 has been moved to [RFC4512] and revised. Changes to the parts of these sections moved to [RFC4512] are detailed in [RFC4512].

3. BNF descriptions of syntax formats have been replaced by ABNF [RFC4234] specifications.

4. The ambiguous statement in RFC 2252, Section 4.3 regarding the use of a backslash quoting mechanism to escape separator symbols has been removed. The escaping mechanism is now explicitly represented in the ABNF for the syntaxes where this provision applies.

5. The description of each of the LDAP syntaxes has been expanded so that they are less dependent on knowledge of X.500 for interpretation.

6. The relationship of LDAP syntaxes to corresponding ASN.1 type definitions has been made explicit.

7. The set of characters allowed in a <PrintableString> (formerly <printablestring>) has been corrected to align with the PrintableString ASN.1 type in [ASN.1]. Specifically, the double quote character has been removed and the single quote character and equals sign have been added.

8. Values of the Directory String, Printable String and Telephone Number syntaxes are now required to have at least one character.

9. The <DITContentRuleDescription>, <NameFormDescription> and <DITStructureRuleDescription> rules have been moved to [RFC4512].

10. The corresponding ASN.1 type for the Other Mailbox syntax has been incorporated from RFC 1274.

11. A corresponding ASN.1 type for the LDAP Syntax Description syntax has been invented.

12. The Binary syntax has been removed because it was not adequately specified, implementations with different incompatible interpretations exist, and it was confused with the ;binary transfer encoding.

13. All discussion of transfer options, including the ";binary" option, has been removed. All imperatives regarding binary transfer of values have been removed.

14. The Delivery Method, Enhanced Guide, Guide, Octet String, Teletex Terminal Identifier and Telex Number syntaxes from RFC 2256 have been incorporated.

15. The <criteria> rule for the Enhanced Guide and Guide syntaxes has been extended to accommodate empty "and" and "or" expressions.

16. An encoding for the <ttx-value> rule in the Teletex Terminal Identifier syntax has been defined.

17. The PKI-related syntaxes (Certificate, Certificate List and Certificate Pair) have been removed. They are reintroduced in [RFC4523] (as is the Supported Algorithm syntax from RFC 2256).

18. The MHS OR Address syntax has been removed since its specification (in RFC 2156) is not at draft standard maturity.

19. The DL Submit Permission syntax has been removed as it depends on the MHS OR Address syntax.

20. The Presentation Address syntax has been removed since its specification (in RFC 1278) is not at draft standard maturity.

21. The ACI Item, Access Point, Audio, Data Quality, DSA Quality, DSE Type, LDAP Schema Description, Master And Shadow Access Points, Modify Rights, Protocol Information, Subtree Specification, Supplier Information, Supplier Or Consumer and Supplier And Consumer syntaxes have been removed. These syntaxes are referenced in RFC 2252, but not defined.

22. The LDAP Schema Definition syntax (defined in RFC 2927) and the Mail Preference syntax have been removed on the grounds that they are out of scope for the core specification.

23. The description of each of the matching rules has been expanded so that they are less dependent on knowledge of X.500 for interpretation.

24. The caseIgnoreIA5SubstringsMatch matching rule from RFC 2798 has been added.

25. The caseIgnoreListSubstringsMatch, caseIgnoreOrderingMatch and caseIgnoreSubstringsMatch matching rules have been added to the list of matching rules for which the provisions for handling

leading, trailing and multiple adjoining whitespace characters apply (now through string preparation). This is consistent with the definitions of these matching rules in X.500. The caseIgnoreIA5SubstringsMatch rule has also been added to the list.

26. The specification of the octetStringMatch matching rule from RFC 2256 has been added to this document.

27. The presentationAddressMatch matching rule has been removed as it depends on an assertion syntax (Presentation Address) that is not at draft standard maturity.

28. The protocolInformationMatch matching rule has been removed as it depends on an undefined assertion syntax (Protocol Information).

29. The definitive reference for ASN.1 has been changed from X.208 to X.680 since X.680 is the version of ASN.1 referred to by X.500.

30. The specification of the caseIgnoreListSubstringsMatch matching rule from RFC 2798 & X.520 has been added.

31. String preparation algorithms have been applied to the character string matching rules.

32. The specifications of the booleanMatch, caseExactMatch, caseExactOrderingMatch, caseExactSubstringsMatch, directoryStringFirstComponentMatch, integerOrderingMatch, keywordMatch, numericStringOrderingMatch, octetStringOrderingMatch and wordMatch matching rules from RFC 3698 & X.520 have been added.

Author's Address

Steven Legg
eB2Bcom
Suite3, Woodhouse Corporate Centre
935 Station Street
Box Hill North, Victoria 3129
AUSTRALIA

Phone: +61 3 9896 7830
Fax: +61 3 9896 7801
EMail: steven.legg@eb2bcom.com

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).