Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 16 September 2026

C. E. Cod竪re, Ed.
Optima SC Inc.

15 March 2026

Lightweight Directory Access Protocol (LDAP): Additional Syntaxes
draft-codere-ldapsyntax-10

Abstract

This document registers additional syntax definitions for use in Lightweight Directory Access Protocol (LDAP) directory and Directory services series X.500. This includes widely used datatypes and syntaxes.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on 16 September 2026.

Table of Contents

1. Introduction

The Lightweight Directory Access Protocol (LDAP) directory defines several data types which specify the syntax definitions of attributes. These are identified by ASN.1 OBJECT IDENTIFIER types. Furthermore, these syntax definitions can be used to uniquely identify data types as character representations in other applications. Some widely used syntax specifications are missing from the initial LDAP specification. This document provides additional syntax definitions that have been registered and may be used by application providers.

1.1. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Some of the syntax definitions are defined according to the regular expressions defined in [RFC9485].

2. Syntaxes

The following additional syntaxes and their associated descriptions and OBJECT IDENTIFIER types are defined.

2.1. ASN.1 Syntax Definitions

The following additional syntaxes are defined and are based on [ASN.1].

2.1.1. Date

The Date type represents a date in the Gregorian calendar. It is defined as a useful TIME type in [ASN.1] and conforms to the extended format syntax of a calendar date as defined in [ISO.8601.2004].

A Date value SHALL be written using the following syntax: YYYY-MM-DD where YYYY represents a year between 1582 and 9999, MM the month value from 01 to 12 and DD a day in the month from 01 to 31.

Examples:

* 9999-02-25

* 1583-01-31

The LDAP definition for the Date syntax is:

* (1.3.6.1.4.1.61799.5.40.31 DESC 'Date')

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

2.1.2. Date-Time

The Date-time type represents a date and local time using a 24 hour clock. It is defined as a useful TIME type in [ASN.1] and conforms to the extended format syntax of a date and time without any timezone specifier as defined in [ISO.8601.2004].

A Date-Time value SHALL be written using the following syntax: YYYY-MM-DDThh:mm:ss where YYYY represents a year between 1582 and 9999, MM the month value from 01 to 12, DD a day in the month from 01 to 31, hh the hour from 00 to 24, mm the minute from 00 to 59, and ss the seconds with allowed values of 00 to 60 where 60 represents a leap second.

Examples:

* 1583-01-01T00:59:59

* 1975-01-19T23:45:34

The LDAP definition for the Date-Time syntax is:

* (1.3.6.1.4.1.61799.5.40.33 DESC 'Date-Time')

This syntax corresponds to the DATE-TIME ASN.1 type from [ASN.1].

2.1.3. Duration

The Duration type represents an elapsed time with a resolution of up to a fractions of seconds. It is defined as a useful TIME type in [ASN.1] and conforms to a subset of the extended format syntax of a time interval by duration as defined in [ISO.8601.2004].

A duration syntax value SHALL conform to the following regular expression, and additionally at least one component designator and its associated valueSHALL be present (i.e., the strings "P" and "PT" are not valid Duration values):

```
P([0-9]+Y)?([0-9]+M)?([0-9]+D)?(T([0-9]+H)?([0-9]+M)?([0-9]+(\.[0-9]{1,3})?S)?)?
```

If the number of years, months, days, hours, minutes, or seconds in any expression equals zero, the number and its corresponding designator MAY be omitted. However, at least one number and its designator SHALL be present.

Examples:

* P29M0D -- 29 months, 0 days to an accuracy of 1 day.

* P29MT0S -- 29 months, 0 days, 0 hours, 0 minutes, 0 seconds, to an accuracy of 1 second.

* PT3445.5S -- 3445.5 seconds

The LDAP definition for the Duration syntax is:

* (1.3.6.1.4.1.61799.5.40.34 DESC ’Duration’)

This syntax corresponds to a very strict subset of the DURATION ASN.1 type from [ASN.1], in that the order of parameters need to be respected and fractional values SHALL only apply to the seconds designation and when present, are limited up to 3 digits (millisecond precision).

2.1.4. Real

The Real type represents the computational approximations to the mathematical "real number". The format for the Real is as defined in Section 21 of [ASN.1].

A Real syntax value SHALL conform to the following regular expression:

```
([-]?[0-9]+(\.[0-9]+)?([E][-]?[0-9]+)?)|PLUS-INFINITY|MINUS-INFINITY|NOT-A-NUMBER
```

Examples:

* 3.14159

* MINUS-INFINITY

* 0

* -5.3E4 -- Equal to -53000

* 5E3 -- 5000

The LDAP definition for the Real syntax is:

* (1.3.6.1.4.1.61799.5.40.9 DESC 'Real')

This syntax corresponds to a subset of the REAL ASN.1 type from [ASN.1] where the sequence syntax is not allowed, the values are limited to base ten, where a digit before the decimal point is required, where only the character 'E' is allowed to specify the exponent, where the preceding optional "+" sign is prohibited in both the mantissa and the exponent, and where leading zeros in the integer part SHALL NOT be present unless the integer part is "0".

2.1.5. Time Of Day

The Time Of Day type represents a local time using a 24 hour clock. It is defined as a useful TIME type in [ASN.1] and conforms to the extended format syntax of a local time as defined in [ISO.8601.2004].

A Time Of Day value SHALL be written using the following syntax: hh:mm:ss where hh represents the hour from 00 to 24, mm represents the minute from 00 to 59, and ss represents the seconds with allowed values of 00 to 60 where 60 represents a leap second.

Examples for Time Of Day:

* 00:59:59

* 01:45:54

The LDAP definition for the Time Of Day syntax is:

* (1.3.6.1.4.1.61799.5.40.32 DESC 'Time Of Day')

This syntax corresponds to the TIME-OF-DAY ASN.1 type from [ASN.1].

2.1.6. Visible String

The Visible String type represents a character repertoire that contains the printable ASCII character set (in the Unicode range U+0020 to U+007E). It is defined in [ASN.1].

This syntax value SHALL conform to the following regular expression:

[0-9A-Za-z !"#$%&’()*+,\-./:;<=>?@\[\\\]^_‘{|}~]*

Examples:

* hello world

* (x+y)=z

The LDAP definition for the Visible String syntax is:

* (1.3.6.1.4.1.61799.5.40.26 DESC ’Visible String’)

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

2.2. Constrained ASN.1 Syntax Definitions

The following additional syntaxes are defined as constraints of basic ASN.1 types that may be used to be more precise in encoding and input validation.

2.2.1. Float32

The Float32 type represents a real number which fits in the range of a [IEEE_754_2019] single precision floating point value.

The Float32 syntax follows the base 10 syntax of the real type (See Section 2.1.4) with a constrained range. For equality matching purposes, negative zero SHALL be considered equal to positive zero and NOT-A-NUMBER SHALL NOT be considered equal to any value, including itself.

Examples:

* 3.14159 -- value will be rounded to the nearest representable value

* MINUS-INFINITY

* -5.3E4 -- Equal to -53000

The LDAP definition for the Float32 type syntax is:

* (1.3.6.1.4.1.61799.5.40.9.4 DESC ’Float32’)

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

```
Float32 ::= REAL (WITH COMPONENTS {
  mantissa (-16777215..16777215),
  base (2),
  exponent (-149..104) })
```

2.2.2. Float64

The Float64 type represents a real number which fits in the range of a [IEEE_754_2019] double precision floating point value.

The Float64 syntax follows the base 10 syntax of the real type (See Section 2.1.4) with a constrained range. For equality matching purposes, negative zero SHALL be considered equal to positive zero and NOT-A-NUMBER SHALL NOT be considered equal to any value, including itself.

Examples:

* 3.1415926535897932 -- value will be rounded to the nearest representable value

* NOT-A-NUMBER

* -5.3E4 -- Equal to -53000

The LDAP definition for the Float64 type syntax is:

* (1.3.6.1.4.1.61799.5.40.9.8 DESC 'Float64')

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

```
Float64 ::= REAL (WITH COMPONENTS {
 mantissa (-9007199254740991..9007199254740991),
 base (2),
 exponent (-1074..971) })
```

2.2.3. UInt8

The UInt8 type represents an unsigned integer value within the range 0 to 255 inclusive.

Examples:

* 0

* 34

The LDAP definition for the UInt8 type syntax is:

* (1.3.6.1.4.1.61799.5.40.2.21 DESC 'UInt8')

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

UInt8 ::= INTEGER(0..255)

2.2.4. UInt16

The UInt16 type represents an unsigned integer value within the range 0 to 65535 inclusive.

Examples:

* 0

* 64991

The LDAP definition for the UInt16 type syntax is:

* (1.3.6.1.4.1.61799.5.40.2.22 DESC 'UInt16')

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

UInt16 ::= INTEGER(0..65535)

2.2.5. UInt32

The UInt32 type represents an unsigned integer value within the range 0 to 4294967295 inclusive.

Examples:

* 0

* 40000000

The LDAP definition for the UInt32 type syntax is:

* (1.3.6.1.4.1.61799.5.40.2.24 DESC 'UInt32')

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

UInt32 ::= INTEGER(0..4294967295)

2.2.6. UInt64

The UInt64 type represents an unsigned integer value within the range 0 to 18446744073709551615 inclusive.

Examples:

* 0

* 844674407370955

The LDAP definition for the UInt64 type syntax is:

* (1.3.6.1.4.1.61799.5.40.2.28 DESC 'UInt64')

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

UInt64 ::= INTEGER(0..18446744073709551615)

2.2.7. Int8

The Int8 type represents a signed integer value within the range -128 to 127 inclusive.

Examples:

* 0

* -123

The LDAP definition for the Int8 type syntax is:

* (1.3.6.1.4.1.61799.5.40.2.1 DESC 'Int8')

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

Int8 ::= INTEGER(-128..127)

2.2.8. Int16

The Int16 type represents a signed integer value within the range -32768 to 32767 inclusive.

Examples:

* 15667

* -32000

The LDAP definition for the Int16 type syntax is:

* (1.3.6.1.4.1.61799.5.40.2.2 DESC 'Int16')

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

```
Int16 ::= INTEGER(-32768 .. 32767)
```

2.2.9. Int32

The Int32 type represents a signed integer value within the range -2147483648 to 2147483647 inclusive.

Examples:

* 15667

* -3200000

The LDAP definition for the Int32 type syntax is:

* (1.3.6.1.4.1.61799.5.40.2.4 DESC 'Int32')

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

```
Int32 ::= INTEGER(-2147483648..2147483647)
```

2.2.10. Int64

The Int64 type represents a signed integer value within the range -9223372036854775808 to 9223372036854775807 inclusive.

Examples:

* -2337203685477580

* 3372036854775807

The LDAP definition for the Int64 type syntax is:

* (1.3.6.1.4.1.61799.5.40.2.8 DESC 'Int64')

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

```
Int64 ::= INTEGER(-9223372036854775808..9223372036854775807)
```

2.2.11. Percentage

The Percentage type represents a percentage value, that is an unsigned integer in the range 0 to 100 inclusive.

Examples:

* 0

* 99

The LDAP definition for the Percentage type syntax is:

* (1.3.6.1.4.1.61799.5.40.2.20 DESC 'Percentage')

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

Percentage ::= INTEGER(0..100)

2.3. Other Syntax Definitions

The following additional syntaxes are defined and are based on IETF RFC's, or other international standards.

2.3.1. Language

A language provides a representation of a spoken or written language as well as an optional region and other specifiers. The exact syntax allowed is defined in Section 2 of [RFC5646].

A Language syntax value SHALL conform to the following regular expression:

([a-zA-Z]{2,8}|x-[a-zA-Z0-9]{1,8})(-[a-zA-Z0-9]{1,8})*

The regular expression provides a structural check only. Conformance to the full syntax defined in [RFC5646] is additionally required.

Examples:

* en

* fr-CA

The LDAP definition for the Language syntax is:

* (1.3.6.1.4.1.61799.5.40.19.1 DESC 'Language')

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

Language ::= PrintableString (PATTERN "([a-zA-Z]#(2,8)|x-[a-zA-Z0-9]#(1,8))(-[a-zA-Z0-9]#(1,8))*")
-- ISO 639 code minimally

2.3.2. Media type

The Media Type syntax identifies values that represent an IANA registered Media type [IANAREG]. The format for the MIME Media type is defined in Section 4.2 of [RFC6838].

This syntax value SHALL conform to the following regular expression:

[A-Za-z0-9]([A-Za-z0-9!#$&^_.+-]){0,126}/[A-Za-z0-9]([A-Za-z0-9!#$&^_.+-]){0,126}

Examples:

* text/xhtml

* application/alto-costmap+json

The LDAP definition for the MIME Media type syntax is:

* (1.3.6.1.4.1.61799.5.40.26.5 DESC 'Media Type')

This syntax corresponds to the following ASN.1 type from [ITU.X520.2019]:

MediaType ::= VisibleString (SIZE(3..255))

2.3.3. OpenDate

An OpenDate represents either part of a Date or a Date and Time in extended format as specified in ISO 8601. The exact syntax allowed is defined by W3C Date and Time formats [W3C.NOTE-datetime-19980827] with a 3 digit fraction. The time component, when present, always contains timezone information.

Examples:

* 2034

* 1975-01

* 1975-01-19

* 1975-01-19T19:20+01:00

* 1975-01-19T19:20:30+01:00

* 1975-01-19T19:20:30.451+01:00

* 1975-01-19T18:20Z

* 1975-01-19T18:20:30Z

* 1975-01-19T18:20:30.451Z

The LDAP definition for the OpenDate syntax is:

* (1.3.6.1.4.1.61799.5.40.14.1 DESC 'OpenDate')

This syntax corresponds to a subset of the TIME ASN.1 type from [ASN.1] with the specified configuration:

```
OpenDate ::=  TIME((SETTINGS "Basic=Date Date=Y Year=Basic")|
 (SETTINGS "Basic=Date Date=YM Year=Basic")|
 (SETTINGS "Basic=Date Date=YMD Year=Basic")|
 (SETTINGS "Basic=Date-Time Date=YMD Year=Basic Time=HM Local-or-UTC=LD")|
 (SETTINGS "Basic=Date-Time Date=YMD Year=Basic Time=HMS Local-or-UTC=LD")|
 (SETTINGS "Basic=Date-Time Date=YMD Year=Basic Time=HMSF3 Local-or-UTC=LD")|
 (SETTINGS "Basic=Date-Time Date=YMD Year=Basic Time=HM Local-or-UTC=Z")|
 (SETTINGS "Basic=Date-Time Date=YMD Year=Basic Time=HMS Local-or-UTC=Z")|
 (SETTINGS "Basic=Date-Time Date=YMD Year=Basic Time=HMSF3 Local-or-UTC=Z"))
```

2.3.4. URI

The URI syntax type identifies values that are referenced by a Uniform Resource Identifier (URI). The format and encoding for the URI is as defined in [RFC3986]. Even if relative URIs are allowed, it is RECOMMENDED they not be used unless the context of use is known.

Examples:

* http://www.example.com/my/picture.jpg

* ldap://ldap.example.com/cn=babs%20jensen

The LDAP definition for the URI syntax is:

* (1.3.6.1.4.1.61799.5.40.26.4 DESC 'URI')

This syntax corresponds to the following ASN.1 type from [ITU.X520.2019]:

```
URI ::= VisibleString (SIZE (1..ub-uri-length)) -- ub-uri-length MUST be at least 2000
```

The value of ub-uri-length (an integer) is implementation defined but MUST be at least 2000 characters.

2.3.5. NCName

The NCName syntax type should be used to identify values that represent identifiers and local attribute names. A name is a subset of the NCName definition in [W3C.xml-names11].

This syntax value SHALL conform to the following regular expression:

```
[\p{L}\p{Nl}_][\p{L}\p{Nl}\p{Nd}._-]*
```

Examples:

* MyID

* attribute.0.subdivision

The LDAP definition for the NCName type syntax is:

* (1.3.6.1.4.1.61799.5.40.12.6 DESC 'NCName')

This syntax corresponds to the following ASN.1 type from [ITU.X520.2019]:

```
-- NOTE: This type does not capture the
-- structural constraints on permitted characters and start characters.
-- The regular expression is the normative definition of valid values
NCName ::= UnboundedDirectoryString
```

2.3.6. Qualified Name

The Qualified Name syntax type identifies values that represent identifiers and attribute names using namespaces. This is a subset of the QName definition in [W3C.xml-names11].

This syntax value SHALL conform to the following regular expression:

```
[\p{L}\p{Nl}_][\p{L}\p{Nl}\p{Nd}._-]*(:[\p{L}\p{Nl}_][\p{L}\p{Nl}\p{Nd}._-]*)?
```

Examples:

* MyID

* attribute.0:subdivision

The LDAP definition for the QualifiedName type syntax is:

* (1.3.6.1.4.1.61799.5.40.12.7 DESC 'QualifiedName')

This syntax corresponds to the following ASN.1 type from [ITU.X520.2019]:

```
-- NOTE: This type does not capture the
-- structural constraints on permitted characters and start characters.
-- The regular expression is the normative definition of valid values
QualifiedName ::= UnboundedDirectoryString
```

2.3.7. Text

The Text type represents a Unicode string with a restricted character repertoire. A Text string value MUST NOT contain any code point that has any of the following properties, as defined in the Unicode Character Database [Unicode]:

* General_Category of Cc (Control), with the exception of U+0009 (CHARACTER TABULATION), U+000A (LINE FEED), and U+000D (CARRIAGE RETURN), which are permitted;

* General_Category of Cs (Surrogate);

* General_Category of Co (Private_Use);

* Noncharacter_Code_Point, as defined in PropList.txt of the Unicode Character Database. This is a fixed set of 66 code points: U+FDD0..U+FDEF and U+nFFFE..U+nFFFF (where n ranges from 0 to 10 in hexadecimal).

See Section 5 for security considerations regarding the handling, display, and comparison of values containing format characters or bidirectional controls.

Examples:

* Hello world

* Ceci est une phrase qui est encore plus longue que la pr迪c迪dente

The LDAP definition for the Text type syntax is:

* (1.3.6.1.4.1.61799.5.40.12.3 DESC 'Text')

This syntax corresponds to the following ASN.1 type from [ITU.X520.2019]:

```
Text ::= CHOICE {
  printableString  PrintableString,
  bmpString        BMPString
    (FROM(
      {0, 0, 0, 9}   .. {0, 0, 0, 10}   | -- U+0009..U+000A
      {0, 0, 0, 13}  .. {0, 0, 0, 13}   | -- U+000D
      {0, 0, 0, 32}  .. {0, 0, 0, 126}  | -- U+0020..U+007E
      {0, 0, 0, 160} .. {0, 0, 215, 255} | -- U+00A0..U+D7FF
      {0, 0, 249, 0} .. {0, 0, 253, 207} | -- U+F900..U+FDCF
      {0, 0, 253, 240} .. {0, 0, 255, 253} -- U+FDF0..U+FFFD
    )),
  universalString  UniversalString
    (FROM(
      {0, 0, 0, 9}   .. {0, 0, 0, 10}   | -- U+0009..U+000A
      {0, 0, 0, 13}  .. {0, 0, 0, 13}   | -- U+000D
      {0, 0, 0, 32}  .. {0, 0, 0, 126}  | -- U+0020..U+007E
      {0, 0, 0, 160} .. {0, 0, 215, 255} | -- U+00A0..U+D7FF
      {0, 0, 249, 0} .. {0, 0, 253, 207} | -- U+F900..U+FDCF
      {0, 0, 253, 240} .. {0, 0, 255, 253} | -- U+FDF0..U+FFFD
      {0, 1, 0, 0}   .. {0, 1, 255, 253} | -- U+10000..U+1FFFD
      {0, 2, 0, 0}   .. {0, 2, 255, 253} | -- U+20000..U+2FFFD
      {0, 3, 0, 0}   .. {0, 3, 255, 253} | -- U+30000..U+3FFFD
      {0, 4, 0, 0}   .. {0, 4, 255, 253} | -- U+40000..U+4FFFD
      {0, 5, 0, 0}   .. {0, 5, 255, 253} | -- U+50000..U+5FFFD
      {0, 6, 0, 0}   .. {0, 6, 255, 253} | -- U+60000..U+6FFFD
      {0, 7, 0, 0}   .. {0, 7, 255, 253} | -- U+70000..U+7FFFD
      {0, 8, 0, 0}   .. {0, 8, 255, 253} | -- U+80000..U+8FFFD
      {0, 9, 0, 0}   .. {0, 9, 255, 253} | -- U+90000..U+9FFFD
      {0, 10, 0, 0}  .. {0, 10, 255, 253} | -- U+A0000..U+AFFFD
      {0, 11, 0, 0}  .. {0, 11, 255, 253} | -- U+B0000..U+BFFFD
      {0, 12, 0, 0}  .. {0, 12, 255, 253} | -- U+C0000..U+CFFFD
      {0, 13, 0, 0}  .. {0, 13, 255, 253} | -- U+D0000..U+DFFFD
      {0, 14, 0, 0}  .. {0, 14, 255, 253}  -- U+E0000..U+EFFFD
    )),
  utf8String       UTF8String
    (FROM(
      {0, 0, 0, 9}   .. {0, 0, 0, 10}   | -- U+0009..U+000A
      {0, 0, 0, 13}  .. {0, 0, 0, 13}   | -- U+000D
      {0, 0, 0, 32}  .. {0, 0, 0, 126}  | -- U+0020..U+007E
      {0, 0, 0, 160} .. {0, 0, 215, 255} | -- U+00A0..U+D7FF
      {0, 0, 249, 0} .. {0, 0, 253, 207} | -- U+F900..U+FDCF
      {0, 0, 253, 240} .. {0, 0, 255, 253} | -- U+FDF0..U+FFFD
      {0, 1, 0, 0}   .. {0, 1, 255, 253} | -- U+10000..U+1FFFD
      {0, 2, 0, 0}   .. {0, 2, 255, 253} | -- U+20000..U+2FFFD
      {0, 3, 0, 0}   .. {0, 3, 255, 253} | -- U+30000..U+3FFFD
      {0, 4, 0, 0}   .. {0, 4, 255, 253} | -- U+40000..U+4FFFD
      {0, 5, 0, 0}   .. {0, 5, 255, 253} | -- U+50000..U+5FFFD
      {0, 6, 0, 0}   .. {0, 6, 255, 253} | -- U+60000..U+6FFFD
```

```
          {0, 7, 0, 0}   .. {0, 7, 255, 253} | -- U+70000..U+7FFFD
          {0, 8, 0, 0}   .. {0, 8, 255, 253} | -- U+80000..U+8FFFD
          {0, 9, 0, 0}   .. {0, 9, 255, 253} | -- U+90000..U+9FFFD
          {0, 10, 0, 0}  .. {0, 10, 255, 253} | -- U+A0000..U+AFFFD
          {0, 11, 0, 0}  .. {0, 11, 255, 253} | -- U+B0000..U+BFFFD
          {0, 12, 0, 0}  .. {0, 12, 255, 253} | -- U+C0000..U+CFFFD
          {0, 13, 0, 0}  .. {0, 13, 255, 253} | -- U+D0000..U+DFFFD
          {0, 14, 0, 0}  .. {0, 14, 255, 253}  -- U+E0000..U+EFFFD
        ))
    }
```

2.3.8. Normalized String

The Normalized String type represents a whitespace-normalized Unicode string. A Normalized String value SHALL conform to all the restrictions of the Text type (Section 2.3.7), with the following additional constraints:

Prior to validation, the following code points SHALL be mapped to U+0020 (SPACE):

* U+0009 (CHARACTER TABULATION), U+000A (LINE FEED), and U+000D (CARRIAGE RETURN);

* Code points with General_Category Zs (Space_Separator) other than U+0020 (SPACE);

* Code points with General_Category Zl (Line_Separator) or Zp (Paragraph_Separator).

The canonical form of a Normalized String contains only U+0020 SPACE as a whitespace character. All other Unicode scalar values permitted by the Text type are permitted.

See Section 5 for security considerations regarding the handling, display, and comparison of values containing format characters or bidirectional controls.

Examples:

* Paragraph start with some leading spaces.

* This is some other text with spaces.

The LDAP definition for the Normalized String syntax is:

* (1.3.6.1.4.1.61799.5.40.12.4 DESC ’Normalized String’)

This syntax corresponds to the following ASN.1 type definition:

```
NormalizedString ::= CHOICE {
  printableString  PrintableString,
  bmpString        BMPString
    (FROM(
      {0, 0, 0, 32}  .. {0, 0, 0, 126}  |  -- U+0020..U+007E
      {0, 0, 0, 161} .. {0, 0, 22, 127} |  -- U+00A1..U+167F
      {0, 0, 22, 129} .. {0, 0, 31, 255} | -- U+1681..U+1FFF
      {0, 0, 32, 11} .. {0, 0, 32, 39}  |  -- U+200B..U+2027
      {0, 0, 32, 42} .. {0, 0, 32, 46}  |  -- U+202A..U+202E
      {0, 0, 32, 48} .. {0, 0, 32, 94}  |  -- U+2030..U+205E
      {0, 0, 32, 96} .. {0, 0, 47, 255} |  -- U+2060..U+2FFF
      {0, 0, 48, 1}  .. {0, 0, 215, 255} | -- U+3001..U+D7FF
      {0, 0, 249, 0} .. {0, 0, 253, 207} | -- U+F900..U+FDCF
      {0, 0, 253, 240} .. {0, 0, 255, 253} -- U+FDF0..U+FFFD
    )),
  universalString  UniversalString
    (FROM(
      {0, 0, 0, 32}  .. {0, 0, 0, 126}  |  -- U+0020..U+007E
      {0, 0, 0, 161} .. {0, 0, 22, 127} |  -- U+00A1..U+167F
      {0, 0, 22, 129} .. {0, 0, 31, 255} | -- U+1681..U+1FFF
      {0, 0, 32, 11} .. {0, 0, 32, 39}  |  -- U+200B..U+2027
      {0, 0, 32, 42} .. {0, 0, 32, 46}  |  -- U+202A..U+202E
      {0, 0, 32, 48} .. {0, 0, 32, 94}  |  -- U+2030..U+205E
      {0, 0, 32, 96} .. {0, 0, 47, 255} |  -- U+2060..U+2FFF
      {0, 0, 48, 1}  .. {0, 0, 215, 255} | -- U+3001..U+D7FF
      {0, 0, 249, 0} .. {0, 0, 253, 207} | -- U+F900..U+FDCF
      {0, 0, 253, 240} .. {0, 0, 255, 253} | -- U+FDF0..U+FFFD
      {0, 1, 0, 0}   .. {0, 1, 255, 253} | -- U+10000..U+1FFFD
      {0, 2, 0, 0}   .. {0, 2, 255, 253} | -- U+20000..U+2FFFD
      {0, 3, 0, 0}   .. {0, 3, 255, 253} | -- U+30000..U+3FFFD
      {0, 4, 0, 0}   .. {0, 4, 255, 253} | -- U+40000..U+4FFFD
      {0, 5, 0, 0}   .. {0, 5, 255, 253} | -- U+50000..U+5FFFD
      {0, 6, 0, 0}   .. {0, 6, 255, 253} | -- U+60000..U+6FFFD
      {0, 7, 0, 0}   .. {0, 7, 255, 253} | -- U+70000..U+7FFFD
      {0, 8, 0, 0}   .. {0, 8, 255, 253} | -- U+80000..U+8FFFD
      {0, 9, 0, 0}   .. {0, 9, 255, 253} | -- U+90000..U+9FFFD
      {0, 10, 0, 0}  .. {0, 10, 255, 253} | -- U+A0000..U+AFFFD
      {0, 11, 0, 0}  .. {0, 11, 255, 253} | -- U+B0000..U+BFFFD
      {0, 12, 0, 0}  .. {0, 12, 255, 253} | -- U+C0000..U+CFFFD
      {0, 13, 0, 0}  .. {0, 13, 255, 253} | -- U+D0000..U+DFFFD
      {0, 14, 0, 0}  .. {0, 14, 255, 253}  -- U+E0000..U+EFFFD
    )),
  utf8String       UTF8String
    (FROM(
      {0, 0, 0, 32}  .. {0, 0, 0, 126}  |  -- U+0020..U+007E
      {0, 0, 0, 161} .. {0, 0, 22, 127} |  -- U+00A1..U+167F
```

```
          {0, 0, 22, 129} .. {0, 0, 31, 255} | -- U+1681..U+1FFF
          {0, 0, 32, 11} .. {0, 0, 32, 39}  |  -- U+200B..U+2027
          {0, 0, 32, 42} .. {0, 0, 32, 46}  |  -- U+202A..U+202E
          {0, 0, 32, 48} .. {0, 0, 32, 94}  |  -- U+2030..U+205E
          {0, 0, 32, 96} .. {0, 0, 47, 255} |  -- U+2060..U+2FFF
          {0, 0, 48, 1}  .. {0, 0, 215, 255} | -- U+3001..U+D7FF
          {0, 0, 249, 0} .. {0, 0, 253, 207} | -- U+F900..U+FDCF
          {0, 0, 253, 240} .. {0, 0, 255, 253} | -- U+FDF0..U+FFFD
          {0, 1, 0, 0}   .. {0, 1, 255, 253} | -- U+10000..U+1FFFD
          {0, 2, 0, 0}   .. {0, 2, 255, 253} | -- U+20000..U+2FFFD
          {0, 3, 0, 0}   .. {0, 3, 255, 253} | -- U+30000..U+3FFFD
          {0, 4, 0, 0}   .. {0, 4, 255, 253} | -- U+40000..U+4FFFD
          {0, 5, 0, 0}   .. {0, 5, 255, 253} | -- U+50000..U+5FFFD
          {0, 6, 0, 0}   .. {0, 6, 255, 253} | -- U+60000..U+6FFFD
          {0, 7, 0, 0}   .. {0, 7, 255, 253} | -- U+70000..U+7FFFD
          {0, 8, 0, 0}   .. {0, 8, 255, 253} | -- U+80000..U+8FFFD
          {0, 9, 0, 0}   .. {0, 9, 255, 253} | -- U+90000..U+9FFFD
          {0, 10, 0, 0}  .. {0, 10, 255, 253} | -- U+A0000..U+AFFFD
          {0, 11, 0, 0}  .. {0, 11, 255, 253} | -- U+B0000..U+BFFFD
          {0, 12, 0, 0}  .. {0, 12, 255, 253} | -- U+C0000..U+CFFFD
          {0, 13, 0, 0}  .. {0, 13, 255, 253} | -- U+D0000..U+DFFFD
          {0, 14, 0, 0}  .. {0, 14, 255, 253}  -- U+E0000..U+EFFFD
        ))
    }
```

2.3.9. Token

The Token type represents a whitespace-normalized Unicode string. A Token value SHALL conform to all the restrictions of the Normalized String type (Section 2.3.8), with the following additional constraints:

* No leading and trailing U+0020 SPACE characters.

* No consecutive U+0020 SPACE characters.

This is similar to the Token datatype in [W3C.xmlschema11-2].

The resulting canonical form is either the empty string, or a string that does not begin or end with U+0020 SPACE and contains no consecutive U+0020 SPACE characters.

See Section 5 for security considerations regarding the handling, display, and comparison of values containing format characters or bidirectional controls.

Examples:

* This is a token with spaces.

* _Token_

The LDAP definition for the Token type syntax is:

* (1.3.6.1.4.1.61799.5.40.12.5 DESC 'Token')

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

```
  Token ::= CHOICE {
    printableString  PrintableString
      (PATTERN "([^ ]+( [^ ]+)*)?"),
    bmpString        BMPString
      (FROM(
        {0, 0, 0, 32}  .. {0, 0, 0, 126}  |  -- U+0020..U+007E
        {0, 0, 0, 161} .. {0, 0, 22, 127} |  -- U+00A1..U+167F
        {0, 0, 22, 129} .. {0, 0, 31, 255} | -- U+1681..U+1FFF
        {0, 0, 32, 11} .. {0, 0, 32, 39}  |  -- U+200B..U+2027
        {0, 0, 32, 42} .. {0, 0, 32, 46}  |  -- U+202A..U+202E
        {0, 0, 32, 48} .. {0, 0, 32, 94}  |  -- U+2030..U+205E
        {0, 0, 32, 96} .. {0, 0, 47, 255} |  -- U+2060..U+2FFF
        {0, 0, 48, 1}  .. {0, 0, 215, 255} | -- U+3001..U+D7FF
        {0, 0, 249, 0} .. {0, 0, 253, 207} | -- U+F900..U+FDCF
        {0, 0, 253, 240} .. {0, 0, 255, 253} -- U+FDF0..U+FFFD
      ))
      (PATTERN "([^ ]+( [^ ]+)*)?"),
    universalString  UniversalString
      (FROM(
        {0, 0, 0, 32}  .. {0, 0, 0, 126}  |  -- U+0020..U+007E
        {0, 0, 0, 161} .. {0, 0, 22, 127} |  -- U+00A1..U+167F
        {0, 0, 22, 129} .. {0, 0, 31, 255} | -- U+1681..U+1FFF
        {0, 0, 32, 11} .. {0, 0, 32, 39}  |  -- U+200B..U+2027
        {0, 0, 32, 42} .. {0, 0, 32, 46}  |  -- U+202A..U+202E
        {0, 0, 32, 48} .. {0, 0, 32, 94}  |  -- U+2030..U+205E
        {0, 0, 32, 96} .. {0, 0, 47, 255} |  -- U+2060..U+2FFF
        {0, 0, 48, 1}  .. {0, 0, 215, 255} | -- U+3001..U+D7FF
        {0, 0, 249, 0} .. {0, 0, 253, 207} | -- U+F900..U+FDCF
        {0, 0, 253, 240} .. {0, 0, 255, 253} | -- U+FDF0..U+FFFD
        {0, 1, 0, 0}   .. {0, 1, 255, 253} | -- U+10000..U+1FFFD
        {0, 2, 0, 0}   .. {0, 2, 255, 253} | -- U+20000..U+2FFFD
        {0, 3, 0, 0}   .. {0, 3, 255, 253} | -- U+30000..U+3FFFD
        {0, 4, 0, 0}   .. {0, 4, 255, 253} | -- U+40000..U+4FFFD
        {0, 5, 0, 0}   .. {0, 5, 255, 253} | -- U+50000..U+5FFFD
        {0, 6, 0, 0}   .. {0, 6, 255, 253} | -- U+60000..U+6FFFD
        {0, 7, 0, 0}   .. {0, 7, 255, 253} | -- U+70000..U+7FFFD
        {0, 8, 0, 0}   .. {0, 8, 255, 253} | -- U+80000..U+8FFFD
        {0, 9, 0, 0}   .. {0, 9, 255, 253} | -- U+90000..U+9FFFD
```

```
          {0, 10, 0, 0}  .. {0, 10, 255, 253} | -- U+A0000..U+AFFFD
          {0, 11, 0, 0}  .. {0, 11, 255, 253} | -- U+B0000..U+BFFFD
          {0, 12, 0, 0}  .. {0, 12, 255, 253} | -- U+C0000..U+CFFFD
          {0, 13, 0, 0}  .. {0, 13, 255, 253} | -- U+D0000..U+DFFFD
          {0, 14, 0, 0}  .. {0, 14, 255, 253}  -- U+E0000..U+EFFFD
        ))
        (PATTERN "([^ ]+( [^ ]+)*)?"),
      utf8String       UTF8String
        (FROM(
          {0, 0, 0, 32}  .. {0, 0, 0, 126}  |  -- U+0020..U+007E
          {0, 0, 0, 161} .. {0, 0, 22, 127} |  -- U+00A1..U+167F
          {0, 0, 22, 129} .. {0, 0, 31, 255} | -- U+1681..U+1FFF
          {0, 0, 32, 11} .. {0, 0, 32, 39}  |  -- U+200B..U+2027
          {0, 0, 32, 42} .. {0, 0, 32, 46}  |  -- U+202A..U+202E
          {0, 0, 32, 48} .. {0, 0, 32, 94}  |  -- U+2030..U+205E
          {0, 0, 32, 96} .. {0, 0, 47, 255} |  -- U+2060..U+2FFF
          {0, 0, 48, 1}  .. {0, 0, 215, 255} | -- U+3001..U+D7FF
          {0, 0, 249, 0} .. {0, 0, 253, 207} | -- U+F900..U+FDCF
          {0, 0, 253, 240} .. {0, 0, 255, 253} | -- U+FDF0..U+FFFD
          {0, 1, 0, 0}   .. {0, 1, 255, 253} | -- U+10000..U+1FFFD
          {0, 2, 0, 0}   .. {0, 2, 255, 253} | -- U+20000..U+2FFFD
          {0, 3, 0, 0}   .. {0, 3, 255, 253} | -- U+30000..U+3FFFD
          {0, 4, 0, 0}   .. {0, 4, 255, 253} | -- U+40000..U+4FFFD
          {0, 5, 0, 0}   .. {0, 5, 255, 253} | -- U+50000..U+5FFFD
          {0, 6, 0, 0}   .. {0, 6, 255, 253} | -- U+60000..U+6FFFD
          {0, 7, 0, 0}   .. {0, 7, 255, 253} | -- U+70000..U+7FFFD
          {0, 8, 0, 0}   .. {0, 8, 255, 253} | -- U+80000..U+8FFFD
          {0, 9, 0, 0}   .. {0, 9, 255, 253} | -- U+90000..U+9FFFD
          {0, 10, 0, 0}  .. {0, 10, 255, 253} | -- U+A0000..U+AFFFD
          {0, 11, 0, 0}  .. {0, 11, 255, 253} | -- U+B0000..U+BFFFD
          {0, 12, 0, 0}  .. {0, 12, 255, 253} | -- U+C0000..U+CFFFD
          {0, 13, 0, 0}  .. {0, 13, 255, 253} | -- U+D0000..U+DFFFD
          {0, 14, 0, 0}  .. {0, 14, 255, 253}  -- U+E0000..U+EFFFD
        ))
        (PATTERN "([^ ]+( [^ ]+)*)?")
    }
```

2.3.10. Time Of Day with Timezone

The Time Of Day with Timezone syntax type represents a time with explicit timezone information using a 24 hour clock. It is defined as a TIME type in [ASN.1] and conforms to the extended format syntax of a time either represented as a local time followed by a UTC offset or as a UTC time indicated by the ’Z’ designator, as defined in [ISO.8601.2004].

A Time Of Day with Timezone value SHALL be written using the following syntax: hh:mm:ss where hh represents the hour from 00 to 24, mm represents the minute from 00 to 59, and ss represents the seconds with allowed values of 00 to 60 where 60 represents a leap second followed by a timezone indicator.

The timezone indicator is in the form +hh:mm where hh represents the number of hours and mm the number of minutes if the local time is ahead of or equal to UTC time. The timezone indicator is -hh:mm if the local time is behind UTC time. If the time represents an UTC time, the time shall be followed without space, by the timezone UTC designator [Z]. This standard supports time differences in the range 15 hours to +15 hours to align with [ASN.1].

Examples:

* 00:59:59Z

* 01:45:54-01:00

The LDAP definition for the Time Of Day with Timezone syntax is:

* (1.3.6.1.4.1.61799.5.40.35 DESC 'Time Of Day with Timezone')

This syntax corresponds to a subset of the TIME ASN.1 type from [ASN.1] with the specified configuration:

```
Time-with-timezone ::=
  TIME((SETTINGS "Basic=Time Time=HMS Local-or-UTC=LD")|
       (SETTINGS "Basic=Time Time=HMS Local-or-UTC=Z"))
```

3. Matching rules

This section defines matching rules for syntaxes that require comparison semantics beyond simple string matching. Syntaxes not listed here use existing matching rules from [RFC4517]: integer types (Int8, Int16, Int32, Int64, UInt8, UInt16, UInt32, UInt64, Percentage) use integerMatch and integerOrderingMatch; string types (Text, Normalized String, Token, Visible String, NCName, Qualified Name) use caseExactMatch; Language and Media Type use caseIgnoreMatch; and Date and Date-Time use caseExactMatch and caseExactOrderingMatch.

3.1. uriMatch

The uriMatch rule compares for equality a presented value with an attribute value of type URI (Section 2.3.4).

The comparison is done using syntax-based normalization as defined in Section 6.2.2 of [RFC3986].

The LDAP definition for the uriMatch rule is:

* (1.3.6.1.4.1.61799.5.13.26.4 NAME 'uriMatch' SYNTAX 1.3.6.1.4.1.61799.5.40.26.4)

The rule returns TRUE if the attribute value represents the same URI as the presented value.

3.2. openDateMatch

The openDateMatch rule compares for equality a presented value with an attribute value of type OpenDate (Section 2.3.3).

For comparison purposes, OpenDate values SHALL first be normalized to a full date-time in UTC timezone using the following rules:

1. If the month component is absent it is assumed to be one.

2. If the day component is absent it is assumed to be one.

3. If the hours component is absent it is assumed to be zero.

4. If the minutes component is absent it is assumed to be zero.

5. If the seconds component is absent it is assumed to be zero.

6. If the decimal fraction component is absent it is assumed to be zero.

7. If a timezone is present the date and time is normalized to the UTC timezone.

8. If no timezone is present it is assumed, by convention, to be in the UTC timezone.

9. If the time component indicates a value of 24:00:00 it is replaced by 00:00:00 and the day is incremented by one.

This normalization procedure is inspired by, but not identical to, the implicit timezone and value comparison rules defined in Section 9.4 of [W3C.xpath-functions-31]. After normalization, values are compared chronologically.

The LDAP definition for the openDateMatch rule is:

* (1.3.6.1.4.1.61799.5.13.14.1 NAME 'openDateMatch' SYNTAX 1.3.6.1.4.1.61799.5.40.14.1)

The rule returns TRUE if the attribute value represents the same OpenDate as the presented value.

3.3. openDateOrderingMatch

The openDateOrderingMatch rule compares the ordering a presented value with an attribute value of type OpenDate (Section 2.3.3).

Values are first normalized using the algorithm in Section 3.2. The normalized values are then compared chronologically.

The LDAP definition for the openDateOrderingMatch rule is:

* (1.3.6.1.4.1.61799.5.13.14.1.1 NAME 'openDateOrderingMatch' SYNTAX 1.3.6.1.4.1.61799.5.40.14.1)

The rule returns TRUE if the attribute value represents an OpenDate which is earlier than the presented OpenDate.

3.4. timeMatch

The timeMatch rule compares for equality a presented value with an attribute value of type Time Of Day (Section 2.1.5).

Values are compared component by component (hours, minutes, seconds) in order of significance. A time value of 24:00:00 is considered end of day, and is therefore not equal to 00:00:00.

The LDAP definition for the timeMatch rule is:

* (1.3.6.1.4.1.61799.5.13.32 NAME 'timeMatch' SYNTAX 1.3.6.1.4.1.61799.5.40.32)

The rule returns TRUE if the attribute value represents the same Time Of Day as the presented value.

3.5. timeOrderingMatch

The timeOrderingMatch rule compares the ordering of a presented value with an attribute value of type Time Of Day (Section 2.1.5).

Values are compared component by component (hours, minutes, seconds) in order of significance. A time value of 24:00:00 is considered end of day, and is therefore greater than 00:00:00.

The LDAP definition for the timeOrderingMatch rule is:

* (1.3.6.1.4.1.61799.5.13.32.1 NAME 'timeOrderingMatch' SYNTAX 1.3.6.1.4.1.61799.5.40.32)

The rule returns TRUE if the attribute value represents a Time Of Day which is earlier than the presented Time Of Day.

3.6. timeWithTZMatch

The timeWithTZMatch rule compares for equality a presented value with an attribute value of type Time Of Day with Timezone (Section 2.3.10).

Before comparison, both values are normalized to UTC. During normalization, a time value of 24:00:00 is replaced with 00:00:00. After normalization, values are equal if and only if their hours, minutes and seconds components are identical.

The LDAP definition for the timeWithTZMatch rule is:

* (1.3.6.1.4.1.61799.5.13.35 NAME 'timeWithTZMatch' SYNTAX 1.3.6.1.4.1.61799.5.40.35)

The rule returns TRUE if the attribute value represents the same Time Of Day with Timezone as the presented value.

3.7. timeWithTZOrderingMatch

The timeWithTZOrderingMatch rule compares the ordering of a presented value with an attribute value of type Time Of Day with Timezone (Section 2.3.10).

Before comparison, both values are normalized to UTC. During normalization, a time value of 24:00:00 is replaced with 00:00:00. After normalization, values are compared component by component (hours, minutes, seconds) in order of significance.

The LDAP definition for the timeWithTZOrderingMatch rule is:

* (1.3.6.1.4.1.61799.5.13.35.1 NAME 'timeWithTZOrderingMatch' SYNTAX 1.3.6.1.4.1.61799.5.40.35)

The rule returns TRUE if the attribute value represents a Time Of Day with Timezone which is earlier than the presented Time Of Day with Timezone.

3.8. realMatch

The realMatch rule compares for equality a presented value with an attribute value of type Real (Section 2.1.4).

This matching rule also applies to the Float32 and Float64 syntaxes, which are constrained subsets of the Real syntax.

Values are compared by their mathematical value. NOT-A-NUMBER is not equal to any value, including itself. Negative zero is equal to positive zero. MINUS-INFINITY is equal only to itself, and PLUS-INFINITY is equal only to itself.

The LDAP definition for the realMatch rule is:

* (1.3.6.1.4.1.61799.5.13.9 NAME 'realMatch' SYNTAX 1.3.6.1.4.1.61799.5.40.9)

The rule returns TRUE if the attribute value represents the same Real value as the presented value.

3.9. realOrderingMatch

The realOrderingMatch rule compares the ordering of a presented value with an attribute value of type Real (Section 2.1.4).

This matching rule also applies to the Float32 and Float64 syntaxes, which are constrained subsets of the Real syntax.

MINUS-INFINITY is less than all finite values and PLUS-INFINITY is greater than all finite values. Negative zero is equal to positive zero for ordering purposes. NOT-A-NUMBER is greater than all other values, including PLUS-INFINITY. All NOT-A-NUMBER values are considered equal for ordering purposes.

Note: IEEE-754 [IEEE_754_2019] arithmetic comparison leaves NOT-A-NUMBER unordered with respect to all values. This rule instead assigns NOT-A-NUMBER a conventional position to ensure a total ordering, which is required for consistent indexing, sorting, and evaluation of inequality filters by directory implementations.

The LDAP definition for the realOrderingMatch rule is:

* (1.3.6.1.4.1.61799.5.13.9.1 NAME 'realOrderingMatch' SYNTAX 1.3.6.1.4.1.61799.5.40.9)

The rule returns TRUE if the attribute value is less than the presented value.

3.10. durationMatch

The durationMatch rule compares for equality a presented value with an attribute value of type Duration (Section 2.1.3).

Two duration values are equal if and only if they represent the same total number of months and the same total number of milliseconds, where:

* Total months is (years x 12) + months.

* Total milliseconds is (days x 86400000) + (hours x 3600000) + (minutes x 60000) + (seconds x 1000) + milliseconds.

Note that month components and day-time components are compared independently and are never converted into each other. Consequently, a duration such as P1M (one month) is never equal to P30D (thirty days), even though they may represent the same elapsed time for certain dates. This behavior is intentional and consistent with the abstract definition of duration.

Implementations SHOULD use integer arithmetic for these computations to avoid floating-point precision issues.

The LDAP definition for the durationMatch rule is:

* (1.3.6.1.4.1.61799.5.13.34 NAME ’durationMatch’ SYNTAX 1.3.6.1.4.1.61799.5.40.34)

The rule returns TRUE if the attribute value represents the same duration as the presented value.

4. IANA Considerations

IANA is requested to assign the LDAP values [RFC4520] specified in this document to https://www.iana.org/assignments/ldap-parameters/ldap-parameters.xhtml.

4.1. Syntax registration

Subject: Request for LDAP Syntax Registration

Object Identifier: See table below

Description: List of additional useful LDAP syntaxes

Person & email address to contact for further information:
carl.codere@optimasc.com

Specification/Reference: [RFC-to-be]

Author/Change Controller/Owner: IESG

Comments: See table for list of additional syntaxes

Object Identifier	Syntax
1.3.6.1.4.1.61799.5.40.31	Date
1.3.6.1.4.1.61799.5.40.33	Date-Time
1.3.6.1.4.1.61799.5.40.34	Duration
1.3.6.1.4.1.61799.5.40.9.4	Float32
1.3.6.1.4.1.61799.5.40.9.8	Float64
1.3.6.1.4.1.61799.5.40.2.1	Int8
1.3.6.1.4.1.61799.5.40.2.2	Int16
1.3.6.1.4.1.61799.5.40.2.4	Int32
1.3.6.1.4.1.61799.5.40.2.8	Int64
1.3.6.1.4.1.61799.5.40.19.1	Language
1.3.6.1.4.1.61799.5.40.12.6	NCName
1.3.6.1.4.1.61799.5.40.12.4	Normalized String
1.3.6.1.4.1.61799.5.40.26.5	Media Type
1.3.6.1.4.1.61799.5.40.14.1	OpenDate
1.3.6.1.4.1.61799.5.40.2.20	Percentage
1.3.6.1.4.1.61799.5.40.12.7	QualifiedName
1.3.6.1.4.1.61799.5.40.9	Real
1.3.6.1.4.1.61799.5.40.12.3	Text
1.3.6.1.4.1.61799.5.40.12.5	Token
1.3.6.1.4.1.61799.5.40.32	Time Of Day

1.3.6.1.4.1.61799.5.40.35	Time Of Day with Timezone
1.3.6.1.4.1.61799.5.40.2.21	UInt8
1.3.6.1.4.1.61799.5.40.2.22	UInt16
1.3.6.1.4.1.61799.5.40.2.24	UInt32
1.3.6.1.4.1.61799.5.40.2.28	UInt64
1.3.6.1.4.1.61799.5.40.26.4	URI
1.3.6.1.4.1.61799.5.40.26	Visible String

Table 1: List of additional LDAP syntaxes

4.2. Object Identifier Descriptors registration

Subject: Request for LDAP Descriptor Registration

Descriptor (short name): See table below

Object Identifier: See table below

Usage: matching rule

Person & email address to contact for further information:
carl.codere@optimasc.com

Specification/Reference: [RFC-to-be]

Author/Change Controller/Owner: IESG

Comments: See table for list of additional matching rules

Name	OID
durationMatch	1.3.6.1.4.1.61799.5.13.34
openDateMatch	1.3.6.1.4.1.61799.5.13.14.1
openDateOrderingMatch	1.3.6.1.4.1.61799.5.13.14.1.1
realMatch	1.3.6.1.4.1.61799.5.13.9
realOrderingMatch	1.3.6.1.4.1.61799.5.13.9.1
timeMatch	1.3.6.1.4.1.61799.5.13.32
timeOrderingMatch	1.3.6.1.4.1.61799.5.13.32.1
timeWithTZMatch	1.3.6.1.4.1.61799.5.13.35
timeWithTZOrderingMatch	1.3.6.1.4.1.61799.5.13.35.1
uriMatch	1.3.6.1.4.1.61799.5.13.26.4

Table 2: List of additional descriptor registrations

5. Security Considerations

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.

5.1. Text, Normalized String and Token

The Text, Normalized String, and Token syntaxes have been defined for general textual information. For this reason, and for forward compatibility with future versions of the Unicode Standard, the prohibited code points have been kept to a minimum.

These syntaxes permit Unicode format characters (General_Category Cf) because they are required for the correct representation of text in several scripts. However, some format characters may pose security risks when values are displayed or compared without appropriate normalization. Applications MAY impose additional restrictions on permitted code points as a matter of local policy.

These syntaxes SHOULD NOT be used for security-sensitive attributes such as identifiers, group names, or access control labels. The NCName (Section 2.3.5) or Qualified Name (Section 2.3.6) syntaxes SHOULD be used for such purposes instead.

Text, Normalized String, and Token values SHOULD be matched and compared according to specific comparison rules, such as those defined in [RFC4518] or the PRECIS OpaqueString profile ([RFC8265]).

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC9485] Bormann, C. and T. Bray, "I-Regexp: An Interoperable Regular Expression Format", RFC 9485, DOI 10.17487/RFC9485, October 2023, <https://www.rfc-editor.org/info/rfc9485>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[IEEE_754_2019] IEEE, "IEEE Standard for Floating-Point Arithmetic", IEEE IEEE 754-2019, DOI 10.1109/IEEESTD.2019.8766229, 18 July 2019, <https://ieeexplore.ieee.org/document/8766229>.

[RFC4520] Zeilenga, K., "Internet Assigned Numbers Authority (IANA) Considerations for the Lightweight Directory Access Protocol (LDAP)", BCP 64, RFC 4520, DOI 10.17487/RFC4520, June 2006, <https://www.rfc-editor.org/info/rfc4520>.

[W3C.NOTE-datetime-19980827]
Wicksteed, C., Ed. and M. Wolf, Ed., "Date and Time Formats", W3C NOTE NOTE-datetime-19980827, W3C NOTE-datetime-19980827, 27 August 1998, <http://www.w3.org/TR/1998/NOTE-datetime-19980827>.

[W3C.xml-names11]
"Namespaces in XML 1.1 (Second Edition)", W3C REC xml-names11, W3C xml-names11, <https://www.w3.org/TR/xml-names11/>.

[W3C.xmlschema11-2]
"W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes", W3C REC xmlschema11-2, W3C xmlschema11-2, <https://www.w3.org/TR/xmlschema11-2/>.

[ASN.1] International Telecommunication Union, "Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, February 2021.

[ISO.8601.2004]
International Organization for Standardization, "Data elements and interchange formats - Information interchange - Representation of dates and times", ISO Standard 8601, December 2004.

[ITU.X520.2019]
International Telecommunications Union, "Information Technology - Open Systems Interconnection - The Directory: Selected attribute types", ITU-T Recommendation X.520, ISO Standard 9594-7, October 2019.

[Unicode] The Unicode Consortium, "The Unicode Standard", <http://www.unicode.org/versions/latest/>.

6.2. Informative References

[RFC4517] Legg, S., Ed., "Lightweight Directory Access Protocol (LDAP): Syntaxes and Matching Rules", RFC 4517, DOI 10.17487/RFC4517, June 2006, <https://www.rfc-editor.org/info/rfc4517>.

[RFC4518] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): Internationalized String Preparation", RFC 4518, DOI 10.17487/RFC4518, June 2006, <https://www.rfc-editor.org/info/rfc4518>.

[RFC8265] Saint-Andre, P. and A. Melnikov, "Preparation, Enforcement, and Comparison of Internationalized Strings Representing Usernames and Passwords", RFC 8265, DOI 10.17487/RFC8265, October 2017, <https://www.rfc-editor.org/info/rfc8265>.

[W3C.xpath-functions-31] "XPath and XQuery Functions and Operators 3.1", W3C REC xpath-functions-31, W3C xpath-functions-31, <https://www.w3.org/TR/xpath-functions-31/>.

[ITU.X500.2019] International Telecommunications Union, "Information Technology - Open Systems Interconnection - The Directory: Overview of Concepts, Models and Services", ITU-T Recommendation X.500, ISO Standard 9594-1, October 2019.

[IANAREG] IANA, "Media Types", <https://www.iana.org/assignments/media-types/media-types.xhtml>.

Acknowledgements

This document was sponsored by Andy Newton of the IETF ART group.

Sean Turner acted as a shepherd for this document to help it become a proposed standard.

Contributors

This document was reviewed and improved with the help of Howard Chu.

Author’s Address

Carl Eric Codere (editor)
Optima SC Inc.
Canada
Email: carl.codere@optimasc.com
URI: http://www.optimasc.com