INTERNET-DRAFT
Intended Category: Standard Track
Expires: 1 October 2001
Obsoletes: 2253

Editor: Kurt D. Zeilenga
OpenLDAP Foundation
1 April 2001

Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names <draft-ietf-ldapbis-dn-03.txt>

Status of Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

This document is intended to be, after appropriate review and revision, submitted to the RFC Editor as a Standard Track document replacing RFC 2253. Distribution of this memo is unlimited. Technical discussion of this document will take place on the IETF LDAP Revision Working Group (LDAPbis) mailing list <ietf-ldapbis@openldap.org>. Please send editorial comments directly to the document editor <Kurt@OpenLDAP.org>.

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

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

Abstract

The X.500 Directory uses distinguished names as the primary keys to entries in the directory. Distinguished Names are encoded in ASN.1 in

the X.500 Directory protocols. In the Lightweight Directory Access Protocol, a string representation of distinguished names is transferred. This specification defines the string format for representing names, which is designed to give a clean representation of commonly used distinguished names, while being able to represent any distinguished name.

1. Background

This specification assumes familiarity with X.500 [X.500], and the concept of Distinguished Name (DN). It is important to have a common format to be able to unambiguously represent a distinguished name. The primary goal of this specification is ease of encoding and decoding. A secondary goal is to have names that are human readable. It is not expected that LDAP clients with a human user interface would display these strings directly to the user, but would most likely be performing translations (such as expressing attribute type names in one of the local national languages).

This document obsoletes RFC 2253.

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

2. Converting DistinguishedName from ASN.1 to a String

In X.501 [X.501] the ASN.1 structure of distinguished name is defined as:

```
    DistinguishedName ::= RDNSequence

    RDNSequence ::= SEQUENCE OF RelativeDistinguishedName

    RelativeDistinguishedName ::= SET SIZE (1..MAX) OF
        AttributeTypeAndValue

    AttributeTypeAndValue ::= SEQUENCE {
        type  AttributeType,
        value AttributeValue }
```

The following sections define the algorithm for converting from an ASN.1 structured representation to a UTF-8 [RFC2279] string representation.

2.1. Converting the RDNSequence

If the RDNSequence is an empty sequence, the result is the empty or zero length string.

Otherwise, the output consists of the string encodings of each RelativeDistinguishedName in the RDNSequence (according to 2.2), starting with the last element of the sequence and moving backwards toward the first.

The encodings of adjoining RelativeDistinguishedNames are separated by a comma character (',' ASCII 44).

2.2. Converting RelativeDistinguishedName

When converting from an ASN.1 RelativeDistinguishedName to a string, the output consists of the string encodings of each AttributeTypeAndValue (according to 2.3), in any order.

Where there is a multi-valued RDN, the outputs from adjoining AttributeTypeAndValues are separated by a plus ('+' ASCII 43) character.

2.3. Converting AttributeTypeAndValue

The AttributeTypeAndValue is encoded as the string representation of the AttributeType, followed by an equals character ('=' ASCII 61), followed by the string representation of the AttributeValue. The encoding of the AttributeValue is given in section 2.4.

If the AttributeType is in the following table of attribute types associated with LDAP [RFC2252bis], then the type name string from that table is used, otherwise it is encoded as the dotted-decimal encoding of the AttributeType's OBJECT IDENTIFIER. The dotted-decimal notation is described in [RFC2251bis].

The type name string is not case sensitive.

String	X.500 AttributeType
CN	commonName (2.5.4.3)
L	localityName (2.5.4.7)
ST	stateOrProvinceName (2.5.4.8)
O	organizationName (2.5.4.10)
OU	organizationalUnitName (2.5.4.11)
C	countryName (2.5.4.6)

```
STREET  streetAddress (2.5.4.9)
DC      domainComponent (0.9.2342.19200300.100.1.25)
UID     userId (0.9.2342.19200300.100.1.1)
```

2.4. Converting an AttributeValue from ASN.1 to a String

If the AttributeValue is of a type which does not have a string representation defined for it, then it is simply encoded as an octothorpe character ('#' ASCII 35) followed by the hexadecimal representation of each of the octets of the BER encoding of the X.500 AttributeValue. This form SHOULD be used if the AttributeType is of the dotted-decimal form.

Otherwise, if the AttributeValue is of a type which has a string representation, the value is converted first to a UTF-8 string according to its syntax specification (see for example section 6 of [RFC2252bis]).

If the UTF-8 string does not have any of the following characters which need escaping, then that string can be used as the string representation of the value.

- a space (' ' ASCII 32) or octothorpe ('#' ASCII 35) occurring at the beginning of the string

- a space (' ' ASCII 32) character occurring at the end of the string

- one of the characters ",", "+", """, "\", "<", ">" or ";" (ASCII 44, 43, 34, 92, 60, 62, or 59, respectively)

Implementations MAY escape other characters.

Each octet of the character to be escaped is replaced by a backslash and two hex digits, which form a single octet in the code of the character. Alternatively, if and only if the character to be escaped is one of

```
",", "+", """, "\", "<", ">", ";", "#", "=", or " "
(ASCII 44, 43, 34, 92, 60, 62, 59, 35, or 32, respectively)
```

it may be prefixed by a backslash ('\' ASCII 92).

Examples of the escaping mechanism are shown in section 4.

3. Parsing a String back to a Distinguished Name

The structure of the UTF-8 [RFC2279] string is specified using the following Augmented BNF [RFC2234] grammar.

```
distinguishedName = [name]
                    ; may be empty

name              = name-component *(COMMA name-component)

name-component    = attributeTypeAndValue *(PLUS attributeTypeAndValue)

attributeTypeAndValue
                  = attributeType EQUALS attributeValue

attributeType     = keyword / oid

keyword           = ALPHA 1*keychar

keychar           = ALPHA / DIGIT / MINUS

oid               = number *(DOT number)

number            = ( LDIGIT *DIGIT ) / DIGIT

attributeValue    = string / hexstring

string            = *( stringchar / pair )
                    ; the string MUST NOT start with SHARP or SP
                    ; and MUST NOT end with SP

stringchar        = <any UTF-8 character (can be multiple octets)
                    excepting escaped or ESC>

pair              = ESC ( ESC / special / hexpair )

special           = escaped / SHARP / EQUALS / SP

escaped           = COMMA / PLUS / %x22 / %x3C / %x3E / %3B
                    ; "," / "+" / """ / "<" /  ">" / ";"

hexstring         = SHARP 1*hexpair

hexpair           = HEX HEX

HEX               = DIGIT / %x41-46 / %x61-66
                     ; 0-9 / A-F / a-f
```

```
ALPHA           = %x41-5A / %x61-7A
                    ; A-Z / a-z

LDIGIT          = %x31-39
                    ; 1-9

DIGIT           = %x30 / LDIGIT
                    ; 0-9

SP              = %x20 ; space (" ")
SHARP           = %x23 ; sharp sign ("#")
PLUS            = %x2B ; plus sign ("+")
COMMA           = %x2C ; comma (",")
MINUS           = %x2D ; minus sign ("-")
DOT             = %x2E ; period (".")
EQUALS          = %x3D ; equals sign ("=")
ESC             = %x5C ; backslash ("\")
```

Implementations MUST recognize AttributeType string type names (keywords) listed in the Section 2 table, but MAY recognize other names (keywords). Implementations MAY recognize other DN string representations (such as that described in RFC 1779) but SHALL only generate DN strings in accordance with Section 2 of this document.

4. Examples

This notation is designed to be convenient for common forms of name. This section gives a few examples of distinguished names written using this notation. First is a name containing three relative distinguished names (RDNs):

```
UID=jsmith,DC=example,DC=net
```

Here is an example name containing three RDNs, in which the first RDN is multi-valued:

```
OU=Sales+CN=J. Smith,DC=example,DC=net
```

This example shows the method of quoting of a comma in a common name:

```
CN=John Smith\, III,DC=example,DC=net
```

An example name in which a value contains a carriage return character:

```
CN=Before\0dAfter,DC=example,DC=net
```

An example name in which an RDN was of an unrecognized type. The

value is the BER encoding of an OCTET STRING containing two octets 0x48 and 0x69.

```
1.3.6.1.4.1.1466.0=#04024869,DC=example,DC=com
```

Finally, an example of an RDN commonName value consisting of 5 letters:

Unicode Letter Description	10646 code	UTF-8	Quoted
LATIN CAPITAL LETTER L	U0000004C	0x4C	L
LATIN SMALL LETTER U	U00000075	0x75	u
LATIN SMALL LETTER C WITH CARON	U0000010D	0xC48D	\C4\8D
LATIN SMALL LETTER I	U00000069	0x69	i
LATIN SMALL LETTER C WITH ACUTE	U00000107	0xC487	\C4\87

could be written in printable ASCII (useful for debugging purposes):

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

5. Security Considerations

The following security considerations are specific to the handling of distinguished names. LDAP security considerations are discussed in [RFC2251bis] and its normative references.

5.1. Disclosure

Distinguished Names typically consist of descriptive information about the entries they name, which can be people, organizations, devices or other real-world objects. This frequently includes some of the following kinds of information:

- the common name of the object (i.e. a person's full name)
- an email or TCP/IP address
- its physical location (country, locality, city, street address)
- organizational attributes (such as department name or affiliation)

Most countries have privacy laws regarding the publication of information about people.

5.2. Use of Distinguished Names in Security Applications

The transformations of an AttributeValue value from its X.501 form to an LDAP string representation are not always reversible back to the

same BER or DER form. An example of a situation which requires the DER form of a distinguished name is the verification of an X.509 certificate.

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

Applications which require the reconstruction of the DER form of the value SHOULD NOT use the string representation of attribute syntaxes when converting a distinguished name to the LDAP format. Instead, they SHOULD use the hexadecimal form prefixed by the octothorpe ('#') as described in the first paragraph of section 2.4.

6. References

[X.500] "The Directory -- overview of concepts, models and services," ITU-T Rec. X.500(1993).

[X.501] "The Directory -- Models," ITU-T Rec. X.501(1993).

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

[RFC2234] Crocker, D., and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[RFC2279] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

[RFC2251bis] LDAPbis WG, "Lightweight Directory Access Protocol (v3)", a work in progress.

[RFC2252bis] LDAPbis WG, "LDAPv3: Attribute Syntax Definitions", a work in progress.

[RFC2256bis] LDAPbis WG, "LDAPv3: User Schema", a work in progress.

7. Acknowledgment

This document is an update to RFC 2253, by Mark Wahl, Tim Howes, and Steve Kille. RFC 2253 was a product of the IETF ASID Working Group.

This document is a product of the IETF LDAPbis Working Group.

8. Document Editor's Address

Kurt D. Zeilenga
OpenLDAP Foundation
<Kurt@OpenLDAP.org>

Appendix A. Changes made since RFC 2253

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

The following substantive changes were made to RFC 2253:

- Removed IESG Note. The IESG Note is addressed by RFC 2829.
- Replaced specification of additional requirements for LDAPv2 implementations which also support LDAPv3 (Section 4) with a statement (in Section 3) allowing recognization of alternative string representations.
- Updated 2.3 to clarify which table is the published table of names which may be appear in DNs. Remove "as an example" language. Added statement (in Section 3) allowing recognization of additional names. - Updated 2.3 to indicate attribute type name strings are not case sensitive.
- Updated 2.4 to allow hex pair escaping of all characters and clarified escaping for when multiple octet UTF-8 characters are present.
- Rewrote Section 3 to use ABNF as defined in RFC 2234.
- Rewrote Section 3 ABNF to be consistent with 2.4.
- Rewrote examples.
- Added reference to documentations containing LDAP-specific security considerations.

In addition, numerous editorial changes were made.