INTERNET-DRAFT
Intended Category: Standard Track
Expires in six months
Obsoletes: 2253

Editor: Kurt D. Zeilenga
OpenLDAP Foundation
18 August 2002

LDAP: String Representation of Distinguished Names
<draft-ietf-ldapbis-dn-08.txt>

Status of Memo

Abstract

The X.500 Directory uses distinguished names (DNs) as primary keys to entries in the directory. This document defines the string representation used in the Lightweight Directory Access Protocol (LDAP) to transfer distinguished names. The string representation is

designed to give a clean representation of commonly used distinguished names, while being able to represent any distinguished name.

1. Background and Intended Usage

In X.500-based directory systems [X.500], including those accessed using the Lightweight Directory Access Protocol (LDAP) [LDAPTS], distinguished names (DNs) are used to unambiguously refer to a directory entry [X.501][Models].

The structure of a DN [X.501] is described in terms of ASN.1 [X.680]. In the X.500 Directory Access Protocol [X.511] (and other ITU-defined directory protocols), DNs are encoded using the Basic Encoding Rules (BER) [X.690]. In LDAP, DNs are represented in string form.

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 implementations 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 defines the string representation of Distinguished Names used in LDAP [Protocol][Syntaxes]. Section 2 details how to convert a DN from ASN.1 structured representation to a string. Section 3 details how to convert a DN from string to ASN.1 structured representation.

This document does not define a canonical string representation for DNs. Comparison of DNs for equality is to be performed in accordance with the distinguishedNameMatch matching rule [Syntaxes].

This document is an integral part of the LDAP Technical Specification [Roadmap].

This document obsoletes RFC 2253. Changes since RFC 2253 are summarized in Appendix B.

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 BCP 14 [RFC2119].

This specification assumes familiarity with X.500 [X.500], and the concept of Distinguished Name [X.501][Models].

2. Converting DistinguishedName from ASN.1 to a String

In X.501 [X.501] the ASN.1 [X.680] 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 }
```

This section defines the RECOMMENDED algorithm for converting a distinguished name from an ASN.1 structured representation to an UTF-8 [RFC2279] encoded Universal Character Set (UCS) [ISO10646] character 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 Section 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 ("," U+0002C).

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 Section 2.3), in any order.

Where there is a multi-valued RDN, the outputs from adjoining AttributeTypeAndValues are separated by a plus sign ("+" U+0002B) character.

2.3. Converting AttributeTypeAndValue

The AttributeTypeAndValue is encoded as the string representation of the AttributeType, followed by an equals character ("=" U+0003D), 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 [Schema], then the type name string, a <descr>, from that table is used, otherwise it is encoded as the dotted-decimal encoding, a <numericoid>, of the AttributeType's OBJECT IDENTIFIER. The <descr> and <numericoid> is defined in [Models].

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)

Note: This table lists the complete set of type name strings which all implementations MUST recognize in DN string representation. As no extension could reasonable require all existing implementations be updated to recognize additional type name strings, this table is not extensible.

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

If the AttributeType is of the dotted-decimal form, the AttributeValue is represented by an number sign character ("#" U+00023) followed by the hexadecimal encoding of each of the octets of the BER encoding of the X.500 AttributeValue. This form is also used when the syntax of the AttributeValue does not have a native string encoding defined for it or the native string encoding is not restricted to UTF-8 encoded UCS (or a subset of UCS) characters. This form may also be used in other cases, such as when a reversible string representation is desired (see Section 5.2).

Otherwise, if the AttributeValue is of a syntax which has a native string encoding, the value is converted first to a UTF-8 encoded UCS string according to its syntax specification (see for example Section

6 of [Syntaxes]). If that UTF-8 encoded UCS 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 (" " U+00020) or number sign ("#" U+00023) occurring at the beginning of the string;

- a space (" " U+00020) character occurring at the end of the string;

- one of the characters """, "+", ",", ";", "<", ">", or "\" (U+00022, U+0002B, U+0002C, U+0003B, U+0003C, U+0003E, or U+0005C respectively);

- the null (U+00000) character.

Other characters may be escaped.

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 "\"
(U+00020, U+00022, U+00023, U+0002B, U+0002C, U+0003B,
U+0003C, U+0003D, U+0003E, U+0005C respectively)

it can be prefixed by a backslash ("\" U+00005C).

Examples of the escaping mechanism are shown in Section 4.

3. Parsing a String back to a Distinguished Name

The string representation of Distinguished Names is restricted to UTF-8 [RFC2279] encoded characters from the Universal Character Set (UCS) [ISO10646]. The structure of this string representation is specified using the following Augmented BNF [RFC2234] grammar using the common productions defined in [Models].

```
distinguishedName = [ relativeDistinguishedName
    *( COMMA relativeDistinguishedName ) ]

relativeDistinguishedName = attributeTypeAndValue
    *( PLUS attributeTypeAndValue )

attributeTypeAndValue = attributeType EQUALS attributeValue
```

```
attributeType = descr / numericoid

attributeValue = string / hexstring

; The UTF-8 string shall not contain NULL, ESC, or
; one of escaped, shall not start with SHARP or SPACE,
; and shall must not end with SPACE.
string     = [ (leadchar / pair)
               [ *( stringchar / pair ) ( trailchar / pair ) ] ]

leadchar   = LUTF1 / UTFMB
LUTF1      = %x01-1F / %x21 / %x24-2A / %x2D-3A /
             %x3D / %x3F-5B / %x5D-7F

trailchar  = TUTF1 / UTFMB
TUTF1      = %x01-1F / %x21 / %x23-2A / %x2D-3A /
             %x3D / %x3F-5B / %x5D-7F

stringchar = SUTF1 / UTFMB
SUTF1      = %x01-21 / %x23-2A / %x2D-3A /
             %x3D / %x3F-5B / %x5D-7F

pair       = ESC ( ESC / special / hexpair )

special    = escaped / SPACE / SHARP / EQUALS

escaped    = DQUOTE / PLUS / COMMA / SEMI / LANGLE / RANGLE

hexstring  = SHARP 1*hexpair

hexpair    = HEX HEX
```

where the productions <descr>, <numericoid>, <COMMA>, <DQUOTE>, <EQUALS>, <ESC>, <HEX>, <LANGLE>, <NULL>, <PLUS>, <RANGLE>, <SEMI>, <SPACE>, <SHARP>, <UTFMB> are defined in [Models].

Implementations MUST recognize AttributeType name strings (descriptors) listed in the Section 2.3 table, but MAY recognize other name strings. Implementations MAY recognize other DN string representations (such as that described in RFC 1779). However, as there is no requirement for other names or alternative DN string representations to be recognized (and, if so, how), implementations SHOULD 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 escaping 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	UCS code	UTF-8	Quoted
LATIN CAPITAL LETTER L	U+0004C	0x4C	L
LATIN SMALL LETTER U	U+00075	0x75	u
LATIN SMALL LETTER C WITH CARON	U+0010D	0xC48D	\C4\8D
LATIN SMALL LETTER I	U+00069	0x69	i
LATIN SMALL LETTER C WITH ACUTE	U+00107	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 [Protocol] and other documents comprising the LDAP Technical Specification [Roadmap].

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 number sign ('#') as described in the first paragraph of Section 2.3.

5.3. Use of Other Names

Attribute type names are not unique. A string representation generated with names other than those in the Section 2.3 table is ambiguous. That is, two applications may recognize the string as representing two different DNs possibly associated with two different entries. This may lead to a wide range of unexpected behaviors which can have both direct and indirect impacts upon security.

For example, a distinguished name consisting of one RDN with one AVA

of the known locally attribute type FOO and the value "BAR" (an octetString) could be represented in LDAP as the string FOO=BAR. As the name FOO does not uniquely identify an attribute type, the DN FOO=BAR is ambiguous. That is, FOO could be recognized as the attribute type 1.1.1 by one application and 1.2.3.4 in another and not recognized by another. This may lead to operations not behaving as intended.

Applications desiring to generate an unambiguous string representation of a DN SHOULD generate string representation per section 2, not use names other than those in the Section 2.3 table, and while taking Section 5.2 into consideration.

It is noted that while a registry for attribute type names (descriptors) has been established [LDAPIANA], this registry does not remove the ambiguity of attribute types names used in LDAP. It only removes the ambiguity of attribute type names used in Standard Track technical specifications.

6. 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.

7. Document Editor's Address

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

8. Normative References

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

[X.680] ITU-T, "Abstract Syntax Notation One (ASN.1) - Specification of Basic Notation", X.680, 1994.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14 (also 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.

[Models] K. Zeilenga (editor), "LDAP: Directory Information Models", draft-ietf-ldapbis-models-xx.txt, a work in progress.

[Roadmap] K. Zeilenga, "LDAP: Technical Specification Road Map", draft-ietf-ldapbis-roadmap-xx.txt, a work in progress.

[Protocol] J. Sermersheim (editor), "LDAP: The Protocol", draft-ietf-ldapbis-protocol-xx.txt, a work in progress.

[Syntaxes] S. Legg (editor), "LDAP: Syntaxes", draft-ietf-ldapbis-syntaxes-xx.txt, a work in progress.

[Schema] K. Dally (editor), "LDAP: User Schema", draft-ietf-ldapbis-user-schema-xx.txt, a work in progress.

[ISO10646] Universal Multiple-Octet Coded Character Set (UCS) - Architecture and Basic Multilingual Plane, ISO/IEC 10646-1 : 1993.

9. Informative References

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

[X.690] ITU-T, "Specification of ASN.1 encoding rules: Basic, Canonical, and Distinguished Encoding Rules", X.690, 1994.

[LDAPIANA] K. Zeilenga, "IANA Considerations for LDAP", draft-ietf-ldapbis-xx.txt (a work in progress).

[RFC2849] G. Good, "The LDAP Data Interchange Format (LDIF) - Technical Specification", RFC 2849, June 2000.

Appendix A. Presentation Issues

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

The string representation described in this document is not intended to be presented to humans without translation. However, at times it

may be desirable to present non-translated DN strings to users. This section discusses presentation issues associated with non-translated DN strings. Presentation of translated DN strings issues are not discussed in this document. Transcoding issues are also not discussed in this document.

This appendix provides guidance for applications presenting DN strings to users. This section is not comprehensive, it does not discuss all presentation issues which implementors may face.

Not all user interfaces are capable of displaying the full set of UCS characters. Some UCS characters are not displayable.

It is recommended that human interfaces use the optional hex pair escaping mechanism (Section 2.3) to produce a string representation suitable for display to the human. For example, an application only capable of displaying printable characters can generate a DN string for display which escapes all non-printable characters appearing in the AttributeValue's string representation (as demonstrated in the final example of Section 4).

When a DN string is displayed in free form text, it is necessary to distinguish the DN string from surrounding text. While this is often done with white space (as demonstrated in Section 4), it is noted that DN strings may end with white space. Careful readers of Section 3 will note that characters "<" and ">" may only appear in the DN string if escaped. These characters are intended to be used in free form text to distinguish a DN string from surrounding text. For example, <CN=Sam\ > distinguished the string representation of the DN comprised of one RDN consisting of the AVA: the commonName (CN) value "Sam " from the surrounding text. It should be noted to the user that the wrapping "<" and ">" characters are not part of the DN string.

DN strings can be quite long. It is often desirable to line-wrap overly long DN strings in presentations. Line wrapping should be done by inserting white space after the RDN separator character or, if necessary, after the AVA separator character in such presentations. It should be noted to the user that the inserted white space is not part of the DN string and is to be removed before use in LDAP. For example,

    The following DN string is long:
        CN=Kurt D. Zeilenga,OU=Engineering,L=Redwood Shores,
        O=OpenLDAP Foundation,ST=California,C=US
    so it has been line-wrapped for readability. The extra white
    space is to be removed the DN string is used in LDAP.

It is not advised to insert white space otherwise as it may not be

obvious to the user what white space is part of the DN string and what white space was added for readability.

Another alternative is to use the LDAP Data Interchange Format (LDIF) [RFC2849]. For example,

```
The following entry has a long DN:
    dn: CN=Kurt D. Zeilenga,OU=Engineering,L=Redwood Shores,
     O=OpenLDAP Foundation,ST=California,C=US
    CN: Kurt D. Zeilenga
    SN: Zeilenga
    objectClass: person
```

It is noted that that is often desirable to replace dotted-decimal OIDs appearing in DN strings with attribute type names. Such replacement is viewed as a translation and, hence, not discussed here.

Appendix B. 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 has been addressed.
- Clarified (in Section 1), that this document does not define a canonical string representation.
- Replaced specification of additional requirements for LDAPv2 implementations which also support LDAPv3 (RFC 2253, Section 4) with a statement (in Section 3) allowing recognition of alternative string representations.
- Clarified (in Section 2.3) that the "published" table of names which may be appear in DNs is the table which Section 2.3 provides. Remove "as an example" language. Noted this table is not extensible. Added statement (in Section 3) allowing recognition of additional names. Added security considerations (Section 5.3) regarding the use of other names.
- Updated Section 2.3 to indicate attribute type name strings are case insensitive.
- Updated Section 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 general LDAP security considerations.
- Added discussion of presentation issues (Appendix A).

- Added this appendix.

In addition, numerous editorial changes were made.