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 [1], and the concept of Distinguished Name. 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 the local national language).
2. Converting DistinguishedName from ASN.1 to a String

In X.501 [2] the ASN.1 structure of distinguished name is defined as:

\[ \text{DistinguishedName ::= RDNSequence} \]
\[ \text{RDNSequence ::= SEQUENCE OF RelativeDistinguishedName} \]
\[ \text{RelativeDistinguishedName ::= SET SIZE (1..MAX) OF} \]
\[ \text{AttributeTypeAndValue} \]
\[ \text{AttributeTypeAndValue ::= SEQUENCE \{} \]
\[ \text{type AttributeType,} \]
\[ \text{value AttributeValue } \]

The following sections defines the algorithm for converting from an ASN.1 structured representation to a UTF-8 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.
used, otherwise it is encoded as the dotted-decimal encoding of the
AttributeType’s OBJECT IDENTIFIER. The dotted-decimal notation is
described in [3]. As an example, strings for a few of the attribute
types frequently seen in RDNs include:

<table>
<thead>
<tr>
<th>String X.500 AttributeType</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN  commonName</td>
</tr>
<tr>
<td>L   localityName</td>
</tr>
<tr>
<td>ST  stateOrProvinceName</td>
</tr>
<tr>
<td>O   organizationName</td>
</tr>
<tr>
<td>OU  organizationalUnitName</td>
</tr>
<tr>
<td>C   countryName</td>
</tr>
<tr>
<td>STREET  streetAddress</td>
</tr>
<tr>
<td>DC  domainComponent</td>
</tr>
<tr>
<td>UID  userid</td>
</tr>
</tbody>
</table>

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 the
each of the bytes 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.

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 or ”#” character occurring at the beginning of the string
- a space character occurring at the end of the string
- one of the characters ”,”, ”+”, ” “”, ”\”, ”<”, ”>” or ”;”

Implementations MAY escape other characters.

If a character to be escaped is a one of the list shown above, then it is
prefixed by a backslash (’\’ ASCII 92).

Otherwise the character to be escaped is replaced by a backslash and two
hex digits, which form a single byte in the code of the character.

Examples of the escaping mechanism are shown in section 5.
client MUST also accept (and ignore) the variants given in section 4 of this document.

\[<\text{distinguishedName}> ::= <\text{name}> | "" \quad \text{-- empty string}\]

\[<\text{name}> ::= <\text{name-component}> | <\text{name-component}> \"," <\text{name}>\]

\[<\text{name-component}> ::= \langle\text{attributeTypeAndValue}\rangle \quad \text{or}\quad <\text{attributeTypeAndValue}> \ "+" <\text{name-component}>\]

\[<\text{attributeTypeAndValue}> ::= \langle\text{attributeType}\rangle \ "=" <\text{attributeValue}>\]

\[<\text{attributeType}> ::= 1*\{ \langle\text{keychar}\rangle \} | \langle\text{oid}\rangle\]

\[<\text{oid}> ::= 1*\{ \langle\text{oidchar}\rangle \}\]

\[<\text{oidchar}> ::= \text{digits and '}'\]

\[<\text{attributeValue}> ::= \langle\text{string}\rangle\]

\[<\text{string}> ::= *\{ <\text{stringchar}> | <\text{pair}> \} \quad \text{or}\quad "\" \langle\text{hexstring}\rangle\]

\[\"\" \*\{ <\text{stringchar}> | <\text{special}> | <\text{pair}> \} \"\" \quad \text{-- only from v2}\]

\[<\text{special}> ::= "," | "=" | "+" | ":" | ";" | ":\" | ":\"\]

\[<\text{pair}> ::= ",\" | "=" | "+" | ":" | ":\" | ":\"\]

\[<\text{stringchar}> ::= \text{any character except <special> or "," or '"'}\]

\[<\text{hexstring}> 1*\{ <\text{hexpair}> \}

\[<\text{hexpair}> ::= <\text{hexchar}><\text{hexchar}>\]

\[<\text{hexchar}> ::= 0-9, a-f, A-F\]

4. Relationship with RFC 1779 and LDAPv2

The syntax given in this document is more restrictive than the syntax in RFC 1779. Implementations parsing a string generated by an LDAPv2 client MUST accept the syntax of RFC 1779. Implementations MUST NOT, however, generate any of the RFC 1779 encodings which are not described above in section 2.

Implementations MUST allow a semicolon character to be used instead of a comma to separate RDNs in a distinguished name, and MUST also allow whitespace characters to be present on either side of the comma or semicolon. The whitespace characters are ignored, and the semicolon replaced with a comma.
34) characters, which are not part of the value. Inside the quoted value, the following characters can occur without any escaping:

"", ",", ",+", ",<", ",>", ",\" and ",;"

5. 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):

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

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

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

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

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

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

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

An example name in which an RDN was of an unrecognized type. The value is the BER encoding of an OCTET STRING containing two bytes 0x48 and 0x69.

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

Finally, an example of an RDN surname value consisting of five 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):

SN=Lu\C4\8Di\C4\C7

INTERNET-DRAFT  draft-ietf-asid-ldapv3-dn-02.txt  March 1997

Wahl,Kille,Howes  String Representations of Distinguished Names  Page 6

6. References


6. Security Considerations

Security issues are not discussed in this memo.

7. Author’s Address

Mark Wahl
Critical Angle Inc.
4815 W. Braker Lane #502-385
Austin, TX 78759
USA
EMail: M.Wahl@critical-angle.com

Steve Kille
Isode Ltd.
The Dome
The Square
Richmond, Surrey
TW9 1DT
England
Phone: +44-181-332-9091
EMail: S.Kille@ISODE.COM

Tim Howes
Netscape Communications Corp.
501 E. Middlefield Rd
Mountain View, CA 94043
USA
Phone: +1 415 254-1900
EMail: howes@netscape.com