Network Working Group
INTERNET DRAFT
OBSOLETES: RFC 1960
Expires in six months

Tim Howes
Netscape Communications Corp.
April 1997

The String Representation of LDAP Search Filters
<draft-ietf-asid-ldapv3-filter-01.txt>

1. Status of this Memo

2. Abstract

The Lightweight Directory Access Protocol (LDAP) [1] defines a network representation of a search filter transmitted to an LDAP server. Some applications may find it useful to have a common way of representing these search filters in a human-readable form. This document defines a human-readable string format for representing LDAP search filters.

This document replaces RFC 1960, extending the string LDAP filter definition to include support for LDAP version 3 extended match filters, and including support for representing the full range of possible LDAP search filters.

3. LDAP Search Filter Definition

An LDAPv3 search filter is defined in Section 4.5.1 of [1] as follows:

```
     Filter ::= CHOICE {
             and                  [0] SET OF Filter,
             or                   [1] SET OF Filter,
             not                  [2] Filter,
             equalityMatch        [3] AttributeValueAssertion,
             substrings           [4] SubstringFilter,
             greaterOrEqual       [5] AttributeValueAssertion,
             lessOrEqual          [6] AttributeValueAssertion,
             present              [7] AttributeDescription,
             approxMatch          [8] AttributeValueAssertion,
             extensibleMatch      [9] MatchingRuleAssertion
     }

     SubstringFilter ::= SEQUENCE {
             type    AttributeDescription,
             SEQUENCE OF CHOICE {
                     initial          [0] LDAPString,
                     any              [1] LDAPString,
                     final            [2] LDAPString
             }
     }

     AttributeValueAssertion ::= SEQUENCE {
             attributeDesc   AttributeDescription,
             attributeValue  AttributeValue
     }

     MatchingRuleAssertion ::= SEQUENCE {
             matchingRule    [1] MatchingRuleID OPTIONAL,
             type            [2] AttributeDescription OPTIONAL,
             matchValue      [3] AssertionValue,
             dnAttributes    [4] BOOLEAN DEFAULT FALSE
     }

     AttributeDescription ::= LDAPString

     AttributeValue ::= OCTET STRING

     MatchingRuleID ::= LDAPString

     AssertionValue ::= OCTET STRING

     LDAPString ::= OCTET STRING
```

where the LDAPString above is limited to the UTF-8 encoding of the ISO **10646 [4] character set.** The AttributeDescription is a string representation of the attribute description name and is defined in [1]. The AttributeValue and AssertionValue OCTET STRING have the form defined in [2]. The Filter is encoded for transmission over a network using the Basic Encoding Rules defined in [3], with simplifications described in [1].

4. String Search Filter Definition

The string representation of an LDAP search filter is defined by the following grammar, following the ABNF notation defined in [5]. The filter format uses a prefix notation.

```
     filter     = "(" filtercomp ")"
     filtercomp = and / or / not / item
     and        = "&" filterlist
     or         = "|" filterlist
     not        = "!" filter
     filterlist = 1*filter
     item       = simple / present / substring / extensible
     simple     = attr filtertype value
     filtertype = equal / approx / greater / less
     equal      = "="
     approx     = "~="
     greater    = ">="
     less       = "<="
     extensible = attr [":dn"] [":" matchingrule] ":=" value
                  / [":dn"] ":" matchingrule ":=" value
     present    = attr "=*"
     substring  = attr "=" [initial] any [final]
     initial    = value
     any        = "*" *(value "*")
     final      = value
     attr       = AttributeDescription from Section 4.1.5 of [1]
     matchingrule = MatchingRuleId from Section 4.1.9 of [1]
     value      = AttributeValue from Section 4.1.6 of [1]
```

The attr, matchingrule, and value constructs are as described in the corresponding section of [1] given above.

If a value should contain any of the following characters

Character	ASCII value
*	0x2a
(	0x28
)	0x29
\	0x5c
NUL	0x00

the character must be encoded as the backslash '\' character followed by the two hexadecimal digits representing the ASCII value of the encoded character.

This simple escaping mechanism eliminates filter-parsing ambiguities and allows any filter that can be represented in LDAP to be represented as a NUL-terminated string. The case of the two hexadecimal digits is not significant. Other characters besides the ones listed above may be escaped using this mechanism, for example, non-printing characters.

For example, the filter checking whether the "cn" attribute contained a value with the character "*" anywhere in it would be represented as "(cn=*\2a*)".

Note that although both the substring and present productions in the grammar above can produce the "attr=*" construct, this construct is used only to denote a presence filter.

5. Examples

This section gives a few examples of search filters written using this notation.

```
(cn=Babs Jensen)
(!(cn=Tim Howes))
(&(objectClass=Person)(|(sn=Jensen)(cn=Babs J*)))
(o=univ*of*mich*)
```

The following examples illustrate the use of extensible matching.

```
(cn:1.2.3.4.5:=Fred Flintstone)
(sn:dn:2.4.6.8.10:=Barney Rubble)
(o:dn:=Ace Industry)
```

The second example illustrates the use of the ":dn" notation to indicate that matching rule "2.4.6.8.10" should be used when making comparisons, and that the attributes of an entry's distinguished name should be considered part of the entry when evaluating the match.

The third example denotes an equality match, except that DN components should be considered part of the entry when doing the match.

The following examples illustrate the use of the escaping mechanism.

```
     (o=Parens R Us \28for all your parenthetical needs\29)
     (cn=*\2A*)
     (filename=C:\5cMyFile)
     (bin=\00\00\00\04)
     (sn=Lu\c4\8di\c4\c7)
```

The first example shows the use of the escaping mechanism to represent parenthesis characters. The second shows how to represent a "*" in a value, preventing it from being interpreted as a substring indicator. The third illustrates the escaping of the backslash character.

The fourth example shows a filter searching for the four-byte value 0x00000004, illustrating the use of the escaping mechanism to represent arbitrary data, including NUL characters.

The final example illustrates the use of the escaping mechanism to represent various non-ASCII UTF-8 characters.

6. Security Considerations

This memo describes a string representation of LDAP search filters. While the representation itself has no known security implications, LDAP search filters do. They are interpreted by LDAP servers to select entries from which data is retrieved. LDAP servers should take care to protect the data they maintain from unauthorized access.

7. References

[1] Lightweight Directory Access Protocol (v3), M. Wahl, T. Howes, S. Kille, Internet Draft draft-ietf-asid-ldapv3-protocol-04.txt, March, 1997.

[2] Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions, M. Wahl, A. Coulbeck, T. Howes, S. Kille, Internet Draft draft-ietf-asid-ldapv3-attributes-04.txt, March, 1997.

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

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

[5] Standard for the Format of ARPA Internet Text Messages, D. Crocker,

RFC 822, August, 1982.

8. Author's Address

Tim Howes
Netscape Communications Corp.
501 E. Middlefield Road
Mountain View, CA 94043
USA
+1 415 937-3419
howes@netscape.com