Individual Submission
Internet Draft
Document: draft-sermersheim-nds-ldap-schema-03.txt
Category: Informational

J. Sermersheim
Novell, Inc.

May, 2002

LDAP Schema for NDS

1. Status of this Memo

2. Abstract

This document defines the [LDAPV3] schema descriptions and non-standard syntaxes used by Novell Directory Services (referred to here as NDS). The ObjectClassDescription, AttributeTypeDescription, SyntaxDescriptionm and syntaxes defined in this document are unique to NDS and are meant to compliment those defined in [RFC2252], [RFC2256] and other RFCs and Internet Drafts.

3. Overview

The purpose of this document is to advertise certain LDAP schema descriptors, and syntax elements used by an NDS server which haven't previously been defined in another RFC or Internet Draft. The schema elements defined here represent those in use by NDS version 8 and later.

4. Conventions used in this document

The imperatives from [RFC2119] used in this document are to be interpreted as described there.

4.1. Encodings

This document describes encodings used in an Internet protocol.

This document builds upon [RFC2252], [RFC2256] and their predecessors. Implementers are strongly encouraged to become familiar with those documents before reading this.

The attribute syntax definitions in this document are represented as strings in BNF and in some cases, ASN.1. The intention is that the string representations are used in normal transmissions of attributes using these syntaxes. The ASN.1 is included for cases where the ";binary" Attribute Description option is used (see 4.1.5.1 of [LDAPV3]). Applications may use the ";binary" Attribute Description option when transmitting and requesting attributes, in which case the BER encoding of the ASN.1 data type will be returned.

4.1.1. BNF

The BNF descriptions used here are described in section 4.1 of [RFC2252]. The following definitions are also added:

```
distinguishedname = <Distinguished Name as described in [RFC2253]>
uint16string      = numericstring    ; values transmitted as
                                     ; uint16string have an
                                     ; upper bound of "65535"
uint32string      = numericstring    ; values transmitted as
                                     ; uint32string have an
                                     ; upper bound of "4294967295"
```

There is no historical set convention for the use of value delimiting characters. In this document, the following convention is used:
"#" is used to delimit disparate elements,
"$" is used to delimit like elements (such as those in a list),
"," is used to delimit disparate values that make up a single list element or to separate disparate elements of a complex element which itself is being separated by the "#" character.

4.1.2. ASN.1

The ASN.1 definitions include ASN.1 definitions from [LDAPV3] as well as the following:

```
uint16 ::= INTEGER(0..maxUint16)
maxUint16 ::= 65535 -- (2^^16 - 1)
```

```
uint32 ::= INTEGER(0..maxUint32)
maxUint32 ::= 4294967295 -- (2^^32 - 1)
```

4.2. Distinguished Names

One must be aware that, when storing values in attributes of any syntax listed here which contain a distinguished name, the value of the distinguished name will be validated by the DSA. If a client sends a distinguished name, which does not exist in the DIT, the

LDAP error invalidDNSyntax (34) will be returned in the LDAPResult.

4.3. Object Class Description

The NDSObjectClassDescription defined here, adds to the ObjectClassDescription, which is defined in section 4.4 of [RFC2252]. The additional terms, which begin with the characters "X-NDS ", exist to describe NDS specific object class flags and states that have not yet been adopted by LDAP object classes.

Lines have been folded for readability, transmissions of the NDSObjectClassDescription do not contain newlines. The description of whsp, qdescrs, qdstring, woid, numericstring, and noidlen are given in section 4.1 of [RFC2252].

```
NDSObjectClassDescription = "(" whsp
     numericoid whsp                      ; ObjectClass identifier
     ["NAME" qdescrs]
     ["DESC" qdstring]
     ["OBSOLETE" whsp]
     ["SUP" oids ]                        ; Superior ObjectClasses
     [("ABSTRACT" / "STRUCTURAL" / "AUXILIARY") whsp]
                                          ; default structural
     ["MUST" oids]                        ; AttributeTypes
     ["MAY" oids]                         ; AttributeTypes
     ["X-NDS_NAMING" qdstrings]
     ["X-NDS_CONTAINMENT" qdstrings]
     ["X-NDS_NAME" qdstrings]             ; legacy NDS name
     ["X-NDS_NOT_CONTAINER" qdstrings]    ; default container ('0')
     ["X-NDS_NONREMOVABLE" qdstrings]     ; default removable ('0')
      whsp ")"
```

The qdstrings following X-NDS_NAMING holds a list of all attribute type names that may be used to name this object class. If this term is not supplied when defining an object class, it will automatically

be filled with a list of all MUST and MAY attributes defined for this object class that use any of the following syntaxes: Country String, Directory String, IA5 String, and Printable String.

The qdstrings following X-NDS_CONTAINMENT contains a list of all object class names that may contain this object class. In other words, only entries that are of an object class listed here may be a direct superior in the DIT to entries of this object class. If this term is not included when defining an object class, it will be automatically filled with ('c' 'o' 'ou' 'l' 'domain').

X-NDS_NAME is followed by a qdstrings that contains the legacy NDS name for this object class. An example is ('Organizational Person'). Because NDS was created before LDAP was defined, it sometimes doesn't adhere to the exact same rules as LDAP. One such LDAP rule is that the names of schema elements cannot contain anything other than ASCII letters, the hyphen character and semicolon. NDS allows

spaces, colons, and others. For this reason, some schema elements will have LDAP names that differ from the NDS names that they were first known as.

Valid values for the qdstrings following X-NDS_NOT_CONTAINER are '0' (false) and '1' (true). If true, instances of this object class may not contain other object entries (it may be nothing other than a leaf node).

Valid values for the qdstrings following X-NDS_NONREMOVABLE are '0' (false) and '1' (true). If true, this object class SHALL NOT be removed from the schema.

4.4. Attribute Description

The NDSAttributeTypeDescription defined here, adds to the AttributeTypeDescription, which is defined in section 4.2 of [RFC2252]. The added terms, which begin with the characters "X-NDS ", exist to describe NDS specific attribute constraints which have not yet been adopted by LDAP attributes.

Lines have been folded for readability, transmissions of the NDSAttributeTypeDescription do not contain newlines. The description of whsp, qdescrs, qdstring, woid, numericstring, and noidlen are given in section 4.3.2 of [RFC2252].

NDSAttributeTypeDescription = "(" whsp

```
        numericoid whsp                 ; AttributeType identifier
        ["NAME" qdescrs]                ; name used in AttributeType
        ["DESC" qdstring]               ; description
        ["OBSOLETE" whsp]
        ["SUP" woid]                    ; derived from this other
                                        ; AttributeType
        ["EQUALITY" woid]               ; Matching Rule name
        ["ORDERING" woid]               ; Matching Rule name
        ["SUBSTR" woid]                 ; Matching Rule name
        ["SYNTAX" whsp noidlen whsp]    ; see section 4.3 of[RFC2252]
        ["SINGLE-VALUE" whsp]           ; default multi-valued
        ["COLLECTIVE" whsp]             ; default not collective
        ["NO-USER-MODIFICATION" whsp]   ; default user modifiable
        ["USAGE" whsp AttributeUsage]   ; default userApplications
        ["X-NDS_NAME" qdstrings]        ; legacy NDS name
        ["X-NDS_LOWER_BOUND" qdstrings] ; lower bound. default
                                        ; ('0')(upper is specified in
                                        ; SYNTAX)
        ["X-NDS_UPPER_BOUND" qdstrings] ; reserved
        ["X-NDS_NOT_SCHED_SYNC_IMMEDIATE" qdstrings]
                                        ; default sched sync
                                        ; immediate ('0')
        ["X-NDS_NON_REMOVABLE" qdstrings] ; default false ('0')
        ["X-NDS_PUBLIC_READ" qdstrings] ; default false ('0')
```



```
        ["X-NDS_SERVER_READ" qdstrings] ; default false ('0')
        ["X-NDS_HIDDEN" qdstrings]      ; default false ('0')
        ["X-NDS_NEVER_SYNC" qdstrings]  ; default false ('0')
        ["X-NDS_SCHED_SYNC_NEVER" qdstrings]
                                        ; default schedule sync ('0')
        ["X-NDS_NAME_VALUE_ACCESS" qdstrings] ; default false ('0')
        ["X-NDS_BOTH_MANAGED" qdstrings]; default false ('0')
        ["X-NDS_ENCRYPTED_SYNC" qdstrings] ; default false ('0')
        ["X-NDS_FILTERED_REQUIRED" qdstrings] ; default false ('0')
        ["X-NDS_FILTERED_OPERATIONAL" qdstrings] ; default false ('0')
        whsp ")"

   AttributeUsage =
        "userApplications"     /
        "directoryOperation"   /
        "distributedOperation" /        ; DSA-shared
        "dSAOperation"                  ; DSA-specific, value depends
                                        ; on server
```

X-NDS_NAME is followed by a qdstrings that contains the legacy NDS name for this attribute type. An example is ('Given Name'). Because

NDS was created before LDAP was defined, it sometimes doesn't adhere to the exact same rules as LDAP. One such LDAP rule is that the names of schema elements cannot contain anything other than ASCII letters, the hyphen character and semicolon. NDS allows spaces, colons, and others. For this reason, some schema elements will have LDAP names that differ from the NDS names that they were first known as.

Valid values for the qdstrings following X-NDS_LOWER_BOUND is a quoted uint32string. This represents the lowest value that may be used in this attribute. LDAP only allows for an upper bound (see the definition of noidlen in RFC 2252)

Valid values for the qdstrings following X-NDS_NOT_SCHED_SYNC_IMMEDIATE are '0' (false) and '1' (true). By default, any update to an attribute value will cause a replica synchronization session to occur within 10 seconds. If this flag is set to true, updates to this attribute won't immediately initiate a synchronization session, instead, a synchronization session will be initiated within 30 minutes. At that time the updates will be replicated to other servers.

The qdstrings following X-NDS_NON_REMOVABLE may either be '0' (false) and '1' (true). If true, this attribute cannot be removed from a class definition. This setting can only be set by the system and is read-only.

Valid values for the qdstrings following X-NDS_PUBLIC_READ are '0' (false) and '1' (true). Setting this value to true indicates that anyone can read the attribute without read privileges being assigned. The use of ACL's to restrict the access to this attribute

will be ineffective.

Valid values for the qdstrings following X-NDS_SERVER_READ are '0' (false) and '1' (true). When this is true, server class objects can read the attribute even though the privilege to read has not been granted. Clients cannot set or modify this value.

Valid values for the qdstrings following X-NDS_HIDDEN are '0' (false) and '1' (true). It specifies (when true) that the attribute is hidden from user applications. Typically these attributes are also operational (server generated). This setting is read-only.

Valid values for the qdstrings following X-NDS_NEVER_SYNC are '0' (false) and '1' (true). True here, indicates that this attribute is

never synchronized on other replicas. Clients may not set or modify this value.

Valid values for the qdstrings following X-NDS_SCHED_SYNC_NEVER are '0' (false) and '1' (true). If this flag is set to true, updates to this attribute will not cause a synchronization session to be scheduled. Note that this flag does not prevent the attribute from being synchronized like the X-NDS_NEVER_SYNC does. Once a synchronization session is initiated by another process, the updates to this attribute will be replicated. After the creation of an AttributeType, this field cannot be updated.

Valid values for the qdstrings following X-NDS_NAME_VALUE_ACCESS are '0' (false) and '1' (true). This is specified only when the attribute uses a Distinguished Name syntax. It specifies (when true) that the subject (user) must have management rights (write permissions on the acl attribute) to the entry which the DN names, that is being added or removed from this attribute. In other words, if this is set on my 'friends' attribute, I can't add your DN to my list of friends unless I have write permissions to your acl attribute. For those who are familiar with legacy NDS access APIs, this is the "Write Managed" flag and is renamed here for clarity.

Valid values for the qdstrings following X-NDS_BOTH_MANAGED are '0' (false) and '1' (true). This is specified only when the attribute uses a Distinguished Name syntax. It specifies (when true) that the subject (user) must have management rights (write permissions on the acl attribute) to the entry which the DN names, that is being added or removed from this attribute as well as management rights to the entry being written to. In other words, if this is set on my 'friends' attribute, I can't add your DN to my list of friends unless I have write permissions to your acl attribute, and write permissions to my acl attribute.

Valid values for the qdstrings following X-NDS_ENCRYPTED_SYNC are '0' (false) and '1' (true). It specifies (when true) that values of this attribute are encrypted during the synchronization process.

Valid values for the qdstrings following X-NDS_FILTERED_REQUIRED are '0' (false) and '1' (true). It specifies (when true) that this attribute will be present on filtered replicas, even if the filter is set to prevent it.

Valid values for the qdstrings following X-NDS_FILTERED_OPERATIONAL are '0' (false) and '1' (true). It specifies (when true) that this

attribute on an external reference will exist on a filtered replica even if the filter is set to prevent it

5. Syntaxes

The NDSSyntaxDescription defined here, adds to the SyntaxDescription, which is defined in section 4.3.3 of [RFC2252]. The added terms, which begin with the characters "X-NDS ", exist to describe NDS specific information.

Lines have been folded for readability, transmissions of the NDSSyntaxDescription do not contain newlines. The description of whsp, qdstring, and numericoid are given in section 4.1 of [RFC2252].

```
NDSSyntaxDescription = "(" whsp
     numericoid whsp                 ; Syntax identifier
     [ "DESC" qdstring ]             ; description
     [ "X-NDS_SYNTAX" qdstrings ]    ; legacy NDS syntax identifier
     whsp ")"
```

NDS servers MUST, and Clients that wish to operate with NDS servers SHOULD recognize all the syntaxes described in this section.

5.1 Case Ignore List

(2.16.840.1.113719.1.1.5.1.6 DESC 'Case Ignore List')

This syntax is the same as Postal Address (6.27 of [RFC2252]) except there is no limitation of characters per line, nor number of lines. NDS limits Postal Address to six strings.

Values in this syntax are encoded according to the following BNF:

caseIgnorelist = dstring *("$" dstring)

Backslashes and dollar characters are escaped as described in 6.27 of [RFC2252].

The following ASN.1 data type is used to represent this syntax when transferred in BER form (see 4.1):

caseIgnorelist ::= SEQUENCE OF LDAPString

Attributes of this syntax match for equality using caseIgnoreListMatch (2.5.13.11)

5.2 Tagged Data

(2.16.840.1.113719.1.1.5.1.12 DESC 'Tagged Data')

This is the Net Address syntax in NDS.

Values in this syntax are encoded according to the following BNF:

taggedData = uint32string "#" octetstring

Note that the data portion of the value is represented as an octet string, which may contain non-printable characters. No character escapement is used in the octetstring.

The following ASN.1 data type is used to represent this syntax when transferred in BER form (see 4.1):

```
taggedData ::= SEQUENCE {
     number  uint32,
     data    OCTET STRING
}
```

Attributes of this syntax match for equality if the number (using integerMatch (2.5.13.14)) and the data matches exactly.

5.3 Octet List

(2.16.840.1.113719.1.1.5.1.13 DESC 'Octet List')

Used for attributes that are ordered sequences of octet strings.

Those familiar with this syntax as it is represented in NDS will note that the length field has been omitted.

Because of problems finding a suitable separator character, Values in this syntax are not be transmitted in text form and MUST be transmitted in BER form. This is the default encoding for this syntax and thus it is not necessary to specify the ;binary option.

The following ASN.1 data type is used to represent this syntax:

octetList ::= SEQUENCE OF OCTET STRING

Attributes of this syntax match for equality if the number of octet strings is the same and each octet string matches.

Attributes of this syntax match approximately if at least one octet string matches.

5.4 Tagged String

(2.16.840.1.113719.1.1.5.1.14 DESC 'Tagged String')

This is the Email Address syntax in NDS

Values in this syntax are encoded according to the following BNF:

taggedString = uint32string "#" dstring

No character escapement is used in the dstring.

The following ASN.1 data type is used to represent this syntax when transferred in BER form (see 4.1):

```
taggedString ::= SEQUENCE {
     number  uint32,
     string  LDAPString
}
```

Attributes of this syntax match for equality if the number (using integerMatch (2.5.13.14)) and the string (using caseIgnoreMatch (1.3.6.1.4.1.1466.115.121.1.15)) matches.

5.5 Tagged Name And String

(2.16.840.1.113719.1.1.5.1.15 DESC 'Tagged Name And String')

This is the Path syntax in NDS

Values in this syntax are encoded according to the following BNF:

taggedNameAndString = distinguishedname "#" uint32string "#" dstring

Any occurance of the '#' character in the distinguishedname part MUST be escaped using the rules in Section 4.3 of [RFC2252].

No character escapement is used in the dstring.

The following ASN.1 data type is used to represent this syntax when transferred in BER form (see 4.1):

```
taggedNameAndString ::= SEQUENCE {
     name    LDAPDN,
```

```
     number  uint32,
     string  LDAPString
}
```

The string represented by the string field is compared for equality using the same rules that CaseExactIA5Match (1.3.6.1.4.1.1466.109.114.1) uses, with the following exception:

In comparing two string values, the following white space (spaces, tabs, etc.) is not significant:
Leading spaces (those preceding the first printable character)
Trailing spaces (those following the last printable character)
Multiple consecutive internal spaces (these are taken as equivalent to a single space character)

In searches and comparisons, the string field can specify a presence match by setting the string to "*".

5.6 NDS Replica Pointer

(2.16.840.1.113719.1.1.5.1.16 DESC 'NDS Replica Pointer')

Used for attributes whose values represent partition replicas. A value of this syntax is composed of five parts:

1. The distinguished name of the server that stores the replica.
2. A value describing the capabilities of this copy of the partition: master, secondary, read-only or subordinate reference.
3. A value indicating the current state of the replica (new, dying, locked, changing state, splitting, joining, moving).
4. A number representing the replica (all replicas of a partition have different numbers that are assigned when the replicas are created).
5. A referral containing one or more network addresses that hint at the node at which the server probably resides. Since servers are accessible over different protocols, the server may have an address for each supported protocol.

Values in this syntax may not be transmitted in string format. They MUST be transmitted as BER representations of the following ASN.1:

```
ndsReplicaPointer ::= SEQUENCE {
     serverName              LDAPDN,
     replicaType             uint16,
     replicaState            uint16,
```

```
     replicaNumber           uint32,
     replicaAddressHint      SEQUENCE OF NetAddress
}

NetAddress ::= SEQUENCE {
     transportType           uint32,
     addressValue            OCTET STRING
}

Values for replicaType are:
     0      Master,
     1      Secondary,
     2      Read Only,
     3      Subordinate Reference,
     4      Sparse Write,
```



```
     5      Sparse Read.

Values for replicaState are:
     0      On,
     1      New Replica,
     2      Dying Replica,
     3      Locked,
     4      Change Replica Type State 0,
     5      ChangeReplica Type State 1,
     6      Transition On,
     48     Split State 0,
     49     Split State 1,
     64     Join State 0,
     65     Join State 1,
     66     Join State 2,
     80     Move State 0,
     81     Move State 1,
     82     Move State 2.

Values for transportType are:
     0      ipx,
     1      ip,
     2      sdlc,
     3      tokenringEthernet,
     4      osi nsap,
     5      appleTalk,
     6      netbeui,
     7      sockAddr,
     8      udp,
     9      tcp,
```

```
     10     udp6,
     11     tcp6,
     12     internal,
     13     url.
```

Attributes of this syntax match for equality if the servername matches using distinguishedNameMatch (2.5.13.1)

5.7 NDS ACL

(2.16.840.1.113719.1.1.5.1.17 DESC 'NDS ACL')

Used for attributes whose values represent ACL entries. An ACL value can protect either an object or an attribute. The protected object is always the one that contains the ACL attribute.

Values in this syntax are encoded according to the following BNF:

ndsAcl = privileges "#" scope "#" subjectname "#" protectedattrname

privileges = uint32string

scope = "entry" / "subtree"

subjectname = distinguishedname / "[Self]" / "[Creator]" /
"[Public]" / "[Inheritance Mask]" / "[Root]"

protectedattrname = caseignorestring / "[Entry Rights]" /
"[All Attributes Rights]"

The privileges field is number that represents the kind of access being granted. Performing a bitwise OR on the numbers that represent the desired access arrives at this number. Below a table is shown which specifies the values:

Value	Attributes	[Entry Rights]
1	Compare Attributes	Browse Entry
2	Read Attributes	Add Entry
4	Write, Add, Delete Attrs	Delete Entry
8	Add/Delete Self	Rename Entry
16	(none)	Supervisory
32	Supervisory	(none)
536870912	*Dynamic Group	*Dynamic Group

*The dynamic group value is OR'd when this ACL names a dynamic group object in the subjectname.

The scope field specifies whether or not the privileges are applied to the target entry (the entry containing the ACL) or the target and its subtree.

The subjectname either contains the distinguished name of the entry being granted the privileges, or one of the special values:

[Self]	Indicates the user authenticated in the current connection. This can only be used in the Add Entry operation.
[Creator]	The user who created the object. This can only be used in the Add Entry operation.
[Public]	Includes all objects in the tree.
[Inheritance Mask]	Filters or masks the privileges granted to an object.
[Root]	Denotes the directory tree root object

Any occurance of the # character in the subjectname MUST be escaped using the rules in Section 4.3 of [RFC2252].

The protectedattrname either names a specific attribute that the privileges are applied to, or it contains one of the following special values:

[Entry Rights]	Privileges apply to the entire object, rather than an attribute.
[All Attributes Rights]	Privileges apply to all attributes of the object.

No characters in the protectedattrname field are escaped.

If the protectedattrname neither specifies a valid attribute as defined in the schema, nor one of the special values, an invalidSyntax error will be returned.

The following ASN.1 data type is used to represent this syntax when transferred in binary form (see 4.1):

```
ndsAcl ::= SEQUENCE {
     privileges              uint32,
     scope                   uint32,
     subjectName             LDAPDN,
     protectedAttrName       LDAPString
```

}

The special string values for protectedAttrName and subjectName are the same as given in the BNF above. The privileges field is an integer which represents the bit mask as described above. The scope field is set to either 0 for "entry" or 1 for "subtree".

Attributes of this syntax match for equality if all fields match for equality and match approximate if the attribute name and the subject name match, and any privilege bits set in the filter are also set in the target value.

5.8 NDS Timestamp

(2.16.840.1.113719.1.1.5.1.19 DESC 'NDS Timestamp')

Used for attributes whose values mark the time when a particular event occurred or will occur. A time stamp value has three components:

1. The wholeseconds value consists of the whole number of seconds, where zero equals 12:00 midnight, January 1, 1970, UTC.
2. The replicanum value identifies the server that minted the timestamp. A replica number is assigned whenever a replica is created on a server.
3. The event field is an integer that orders events occurring within the same whole-second interval. The event number restarts at one for each new second.

The initial value of a time stamp has seconds = 1 and event = 0. Values can be skipped, but MUST NOT be reused. An unknown event is coded as 0xFFFF.

Values in this syntax are encoded according to the following BNF:

ndsTimestamp = wholeseconds "#" replicanum "#" event

wholeseconds = uint32string ; 0 = 12:00 midnight Jan 01 1970, UTC

replicanum = uint16string

event = uint16string

The following ASN.1 data type is used to represent this syntax when transferred in binary form (see 4.1):

```
ndsTimestamp ::= SEQUENCE {
     wholeSeconds    uint32,
     replicaNum      uint16,
     event           uint16
}
```

Attributes of this syntax match for equality if the wholeSeconds matches and the event matches.

Attributes of this syntax match for ordering using first the wholeSeconds and then the event.

5.9 Counter

(2.16.840.1.113719.1.1.5.1.22 DESC 'Counter')

This syntax is the same as Integer (1.3.6.1.4.1.1466.115.121.1.27) except that it has the following special properties:

- Attributes using this syntax are implicitly single-valued.

- The LDAP modify-add operation will add the passed number to the value of the counter

- The LDAP modify-delete operation will subtract the passed number to the value of the counter.

Values in this syntax are encoded in the same manner as the INTEGER syntax. See [RFC2252] section 6.16. For example the value 11667 is represented as the character string "11667"

The following ASN.1 data type is used to represent this syntax when transferred in BER form (see 4.1):

```
counter ::=  uint32
```

Integer matching rules apply to attributes of this syntax.

5.10 Tagged Name

(2.16.840.1.113719.1.1.5.1.23 DESC 'Tagged Name')

Holds a distinguished name and a 32 bit unsigned integer.

This is the Back Link syntax in NDS.

Values in this syntax are encoded according to the following BNF:

taggedName = uint32string "#" distinguishedname

Any occurance of the # character in the distinguishedname part MUST be escaped using the rules in Section 4.3 of [RFC2252].

The following ASN.1 data type is used to represent this syntax when transferred in BER form (see 4.1):

```
taggedName ::= SEQUENCE {
     number  uint32,
     name    LDAPDN
}
```

Attributes of this syntax match for equality when the name matches using distinguishedNameMatch (2.5.13.1) and the number matches.

5.11 Typed Name

(2.16.840.1.113719.1.1.5.1.25 DESC 'Typed Name')

Used for attributes whose values represent a level and an interval associated with an object. This syntax names a directory entry and attaches two numeric values to it:

1. The level of the attribute indicates the priority.
2. The interval indicates the frequency of reference.

The objectname value identifies the directory entry referred to by the Typed Name. The values of level and interval are user-assigned and relative.

To be effective, the user must implement them. The user can use them to implement iterative intervals or to enforce priorities.

Values in this syntax are encoded according to the following BNF:

typedname = objectname "#" level "#" interval

objectname = distinguishedname. Any occurance of the # character in the objectname part MUST be escaped using the rules in Section 4.3 of [RFC2252].

level = uint32string

interval = uint32string

The following ASN.1 data type is used to represent this syntax when transferred in binary form (see 4.1):

```
typedName ::= SEQUENCE {
     objectName      LDAPDN,
     level           uint32,
     interval        uint32
}
```

Attributes of this syntax match for equality if the name matches using distinguishedNameMatch (2.5.13.1) and both values match.

8. Matching Rules

As of this printing, NDS tightly binds matching rules to syntaxes. See the syntax definitions in section 5 for matching rule explanations.

9. Security Considerations

While this document discusses the use of security related LDAP attributes and syntaxes, it does not expose or create any security problems which haven't been addressed in other documents.

10. Acknowledgements

The author would like to thank the input and review by individuals at Novell including but not limited to Ed Reed, Renea Campbell, Brian Jarvis, Mark Hinckley, Gary Anderson, Steve McLain, and Judy Wilson.

11. Author's Address

Jim Sermersheim
Novell, Inc.
1800 South Novell Place
Provo, UT 84606
USA
+1 801 861 3088
jimse@novell.com

12. Bibliography

[LDAPV3]
M. Wahl, S. Kille and T. Howes, "Lightweight Directory Access

Protocol (v3)", Internet Standard, December, 1997. Available as RFC2251.

[RFC1778]
T. Howes, S. Kille, W Yeong, C. Robbins, "The String Representation of Standard Attribute Syntaxes", Internet

Standard, March, 1995. Available as RFC1778.

[RFC2252]
M. Wahl, A. Coulbeck, T. Howes, S. Kille, "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions", Internet Standard, December, 1997. Available as RFC2252.

[RFC2253]
M. Wahl, S. Kille, T. Howes, "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names", Internet Standard, December, 1997. Available as RFC2253.

[RFC2119]
S. Bradner, "Key Words for use in RFCs to Indicate Requirement Levels", Internet Standard, March, 1997. Available as RFC2119.

[BYTEORDER]
C. Newman, "Network Byte Order" Internet Draft, February, 1999. Available as draft-newman-network-byte-order-01.txt

Full Copyright Statement