Network Working Group
Request for Comments: 2307
Category: Experimental

L. Howard
Independent Consultant

March 1998

An Approach for Using LDAP as a Network Information Service

Status of this Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This document describes an experimental mechanism for mapping entities related to TCP/IP and the UNIX system into X.500 [X500] entries so that they may be resolved with the Lightweight Directory Access Protocol [RFC2251]. A set of attribute types and object classes are proposed, along with specific guidelines for interpreting them.

The intention is to assist the deployment of LDAP as an organizational nameservice. No proposed solutions are intended as standards for the Internet. Rather, it is hoped that a general consensus will emerge as to the appropriate solution to such problems, leading eventually to the adoption of standards. The proposed mechanism has already been implemented with some success.

1. Background and Motivation

The UNIX (R) operating system, and its derivatives (specifically, those which support TCP/IP and conform to the X/Open Single UNIX specification [XOPEN]) require a means of looking up entities, by matching them against search criteria or by enumeration. (Other operating systems that support TCP/IP may provide some means of resolving some of these entities. This schema is applicable to those environments also.)

These entities include users, groups, IP services (which map names to IP ports and protocols, and vice versa), IP protocols (which map names to IP protocol numbers and vice versa), RPCs (which map names to ONC Remote Procedure Call [RFC1057] numbers and vice versa), NIS

netgroups, booting information (boot parameters and MAC address mappings), filesystem mounts, IP hosts and networks, and RFC822 mail aliases.

Resolution requests are made through a set of C functions, provided in the UNIX system's C library. For example, the UNIX system utility "ls", which enumerates the contents of a filesystem directory, uses the C library function getpwuid() in order to map user IDs to login names. Once the request is made, it is resolved using a "nameservice" which is supported by the client library. The nameservice may be, at its simplest, a collection of files in the local filesystem which are opened and searched by the C library. Other common nameservices include the Network Information Service (NIS) and the Domain Name System (DNS). (The latter is typically used for resolving hosts, services and networks.) Both these nameservices have the advantage of being distributed and thus permitting a common set of entities to be shared amongst many clients.

LDAP is a distributed, hierarchical directory service access protocol which is used to access repositories of users and other network-related entities. Because LDAP is often not tightly integrated with the host operating system, information such as users may need to be kept both in LDAP and in an operating system supported nameservice such as NIS. By using LDAP as the the primary means of resolving these entities, these redundancy issues are minimized and the scalability of LDAP can be exploited. (By comparison, NIS services based on flat files do not have the scalability or extensibility of LDAP or X.500.)

The object classes and attributes defined below are suitable for representing the aforementioned entities in a form compatible with LDAP and X.500 directory services.

2. General Issues

2.1. Terminology

The key words "MUST", "SHOULD", and "MAY" used in this document are to be interpreted as described in [RFC2119].

For the purposes of this document, the term "nameservice" refers to a service, such as NIS or flat files, that is used by the operating system to resolve entities within a single, local naming context. Contrast this with a "directory service" such as LDAP, which supports extensible schema and multiple naming contexts.

The term "NIS-related entities" broadly refers to entities which are typically resolved using the Network Information Service. (NIS was previously known as YP.) Deploying LDAP for resolving these entities does not imply that NIS be used, as a gateway or otherwise. In particular, the host and network classes are generically applicable, and may be implemented on any system that wishes to use LDAP or X.500 for host and network resolution.

The "DUA" (directory user agent) refers to the LDAP client querying these entities, such as an LDAP to NIS gateway or the C library. The "client" refers to the application which ultimately makes use of the information returned by the resolution. It is irrelevant whether the DUA and the client reside within the same address space. The act of the DUA making this information to the client is termed "republishing".

To avoid confusion, the term "login name" refers to the user's login name (being the value of the uid attribute) and the term "user ID" refers to he user's integer identification number (being the value of the uidNumber attribute).

The phrases "resolving an entity" and "resolution of entities" refer respectively to enumerating NIS-related entities of a given type, and matching them against a given search criterion. One or more entities are returned as a result of successful "resolutions" (a "match" operation will only return one entity).

The use of the term UNIX does not confer upon this schema the endorsement of owners of the UNIX trademark. Where necessary, the term "TCP/IP entity" is used to refer to protocols, services, hosts, and networks, and the term "UNIX entity" to its complement. (The former category does not mandate the host operating system supporting the interfaces required for resolving UNIX entities.)

The OIDs defined below are derived from iso(1) org(3) dod(6) internet(1) directory(1) nisSchema(1).

2.2. Attributes

The attributes and classes defined in this document are summarized below.

The following attributes are defined in this document:

- uidNumber
- gidNumber
- gecos
- homeDirectory

loginShell
shadowLastChange
shadowMin
shadowMax
shadowWarning
shadowInactive
shadowExpire
shadowFlag
memberUid
memberNisNetgroup
nisNetgroupTriple
ipServicePort
ipServiceProtocol
ipProtocolNumber
oncRpcNumber
ipHostNumber
ipNetworkNumber
ipNetmaskNumber
macAddress
bootParameter
bootFile
nisMapName
nisMapEntry

Additionally, some of the attributes defined in [RFC2256] are required.

2.3. Object classes

The following object classes are defined in this document:

posixAccount
shadowAccount
posixGroup
ipService
ipProtocol
oncRpc
ipHost
ipNetwork
nisNetgroup
nisMap
nisObject
ieee802Device
bootableDevice

Additionally, some of the classes defined in [RFC2256] are required.

2.4. Syntax definitions

The following syntax definitions [RFC2252] are used by this schema. The nisNetgroupTripleSyntax represents NIS netgroup triples:

```
        ( nisSchema.0.0 NAME 'nisNetgroupTripleSyntax'
          DESC 'NIS netgroup triple' )
```

Values in this syntax are represented by the following:

```
     nisnetgrouptriple = "(" hostname "," username "," domainname ")"
     hostname          = "" / "-" / keystring
     username          = "" / "-" / keystring
     domainname        = "" / "-" / keystring
```

X.500 servers may use the following representation of the above syntax:

```
     nisNetgroupTripleSyntax ::= SEQUENCE {
      hostname  [0] IA5String OPTIONAL,
      username  [1] IA5String OPTIONAL,
      domainname  [2] IA5String OPTIONAL
     }
```

The bootParameterSyntax syntax represents boot parameters:

```
        ( nisSchema.0.1 NAME 'bootParameterSyntax'
          DESC 'Boot parameter' )
```

where:

```
     bootparameter     = key "=" server ":" path
     key               = keystring
     server            = keystring
     path              = keystring
```

X.500 servers may use the following representation of the above syntax:

```
     bootParameterSyntax ::= SEQUENCE {
      key     IA5String,
      server  IA5String,
      path    IA5String
     }
```

Values adhering to these syntaxes are encoded as strings by LDAP servers.

3. Attribute definitions

This section contains attribute definitions to be implemented by DUAs supporting this schema.

```
( nisSchema.1.0 NAME 'uidNumber'
  DESC 'An integer uniquely identifying a user in an
        administrative domain'
  EQUALITY integerMatch SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.1 NAME 'gidNumber'
  DESC 'An integer uniquely identifying a group in an
        administrative domain'
  EQUALITY integerMatch SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.2 NAME 'gecos'
  DESC 'The GECOS field; the common name'
  EQUALITY caseIgnoreIA5Match
  SUBSTRINGS caseIgnoreIA5SubstringsMatch
  SYNTAX 'IA5String' SINGLE-VALUE )

( nisSchema.1.3 NAME 'homeDirectory'
  DESC 'The absolute path to the home directory'
  EQUALITY caseExactIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( nisSchema.1.4 NAME 'loginShell'
  DESC 'The path to the login shell'
  EQUALITY caseExactIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( nisSchema.1.5 NAME 'shadowLastChange'
  EQUALITY integerMatch
  SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.6 NAME 'shadowMin'
  EQUALITY integerMatch
  SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.7 NAME 'shadowMax'
  EQUALITY integerMatch
  SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.8 NAME 'shadowWarning'
  EQUALITY integerMatch
  SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.9 NAME 'shadowInactive'
```

```
        EQUALITY integerMatch
        SYNTAX 'INTEGER' SINGLE-VALUE )

      ( nisSchema.1.10 NAME 'shadowExpire'
        EQUALITY integerMatch
        SYNTAX 'INTEGER' SINGLE-VALUE )

      ( nisSchema.1.11 NAME 'shadowFlag'
        EQUALITY integerMatch
        SYNTAX 'INTEGER' SINGLE-VALUE )

      ( nisSchema.1.12 NAME 'memberUid'
        EQUALITY caseExactIA5Match
        SUBSTRINGS caseExactIA5SubstringsMatch
        SYNTAX 'IA5String' )

      ( nisSchema.1.13 NAME 'memberNisNetgroup'
        EQUALITY caseExactIA5Match
        SUBSTRINGS caseExactIA5SubstringsMatch
        SYNTAX 'IA5String' )

      ( nisSchema.1.14 NAME 'nisNetgroupTriple'
        DESC 'Netgroup triple'
        SYNTAX 'nisNetgroupTripleSyntax' )

      ( nisSchema.1.15 NAME 'ipServicePort'
        EQUALITY integerMatch
        SYNTAX 'INTEGER' SINGLE-VALUE )

      ( nisSchema.1.16 NAME 'ipServiceProtocol'
        SUP name )

      ( nisSchema.1.17 NAME 'ipProtocolNumber'
        EQUALITY integerMatch
        SYNTAX 'INTEGER' SINGLE-VALUE )

      ( nisSchema.1.18 NAME 'oncRpcNumber'
        EQUALITY integerMatch
        SYNTAX 'INTEGER' SINGLE-VALUE )

      ( nisSchema.1.19 NAME 'ipHostNumber'
        DESC 'IP address as a dotted decimal, eg. 192.168.1.1,
              omitting leading zeros'
        EQUALITY caseIgnoreIA5Match
        SYNTAX 'IA5String{128}' )

      ( nisSchema.1.20 NAME 'ipNetworkNumber'
        DESC 'IP network as a dotted decimal, eg. 192.168,
```

```
        omitting leading zeros'
    EQUALITY caseIgnoreIA5Match
    SYNTAX 'IA5String{128}' SINGLE-VALUE )

  ( nisSchema.1.21 NAME 'ipNetmaskNumber'
    DESC 'IP netmask as a dotted decimal, eg. 255.255.255.0,
        omitting leading zeros'
    EQUALITY caseIgnoreIA5Match
    SYNTAX 'IA5String{128}' SINGLE-VALUE )

  ( nisSchema.1.22 NAME 'macAddress'
    DESC 'MAC address in maximal, colon separated hex
        notation, eg. 00:00:92:90:ee:e2'
    EQUALITY caseIgnoreIA5Match
    SYNTAX 'IA5String{128}' )

  ( nisSchema.1.23 NAME 'bootParameter'
    DESC 'rpc.bootparamd parameter'
    SYNTAX 'bootParameterSyntax' )

  ( nisSchema.1.24 NAME 'bootFile'
    DESC 'Boot image name'
    EQUALITY caseExactIA5Match
    SYNTAX 'IA5String' )

  ( nisSchema.1.26 NAME 'nisMapName'
    SUP name )

  ( nisSchema.1.27 NAME 'nisMapEntry'
    EQUALITY caseExactIA5Match
    SUBSTRINGS caseExactIA5SubstringsMatch
    SYNTAX 'IA5String{1024}' SINGLE-VALUE )
```

4. Class definitions

This section contains class definitions to be implemented by DUAs supporting the schema.

The rfc822MailGroup object class MAY be used to represent a mail group for the purpose of alias expansion. Several alternative schemes for mail routing and delivery using LDAP directories, which are outside the scope of this document.

```
  ( nisSchema.2.0 NAME 'posixAccount' SUP top AUXILIARY
    DESC 'Abstraction of an account with POSIX attributes'
    MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
    MAY ( userPassword $ loginShell $ gecos $ description ) )
```

```
        ( nisSchema.2.1 NAME 'shadowAccount' SUP top AUXILIARY
          DESC 'Additional attributes for shadow passwords'
          MUST uid
          MAY ( userPassword $ shadowLastChange $ shadowMin
                shadowMax $ shadowWarning $ shadowInactive $
                shadowExpire $ shadowFlag $ description ) )

        ( nisSchema.2.2 NAME 'posixGroup' SUP top STRUCTURAL
          DESC 'Abstraction of a group of accounts'
          MUST ( cn $ gidNumber )
          MAY ( userPassword $ memberUid $ description ) )

        ( nisSchema.2.3 NAME 'ipService' SUP top STRUCTURAL
          DESC 'Abstraction an Internet Protocol service.
                Maps an IP port and protocol (such as tcp or udp)
                to one or more names; the distinguished value of
                the cn attribute denotes the service's canonical
                name'
          MUST ( cn $ ipServicePort $ ipServiceProtocol )
          MAY ( description ) )

        ( nisSchema.2.4 NAME 'ipProtocol' SUP top STRUCTURAL
          DESC 'Abstraction of an IP protocol. Maps a protocol number
                to one or more names. The distinguished value of the cn
                attribute denotes the protocol's canonical name'
          MUST ( cn $ ipProtocolNumber $ description )
          MAY description )

        ( nisSchema.2.5 NAME 'oncRpc' SUP top STRUCTURAL
          DESC 'Abstraction of an Open Network Computing (ONC)
               [RFC1057] Remote Procedure Call (RPC) binding.
               This class maps an ONC RPC number to a name.
               The distinguished value of the cn attribute denotes
               the RPC service's canonical name'
          MUST ( cn $ oncRpcNumber $ description )
          MAY description )

        ( nisSchema.2.6 NAME 'ipHost' SUP top AUXILIARY

          DESC 'Abstraction of a host, an IP device. The distinguished
                value of the cn attribute denotes the host's canonical
                name. Device SHOULD be used as a structural class'
          MUST ( cn $ ipHostNumber )
          MAY ( l $ description $ manager ) )

        ( nisSchema.2.7 NAME 'ipNetwork' SUP top STRUCTURAL
          DESC 'Abstraction of a network. The distinguished value of
                the cn attribute denotes the network's canonical name'
```

```
        MUST ( cn $ ipNetworkNumber )
        MAY ( ipNetmaskNumber $ l $ description $ manager ) )

      ( nisSchema.2.8 NAME 'nisNetgroup' SUP top STRUCTURAL
        DESC 'Abstraction of a netgroup. May refer to other netgroups'
        MUST cn
        MAY ( nisNetgroupTriple $ memberNisNetgroup $ description ) )

      ( nisSchema.2.09 NAME 'nisMap' SUP top STRUCTURAL
        DESC 'A generic abstraction of a NIS map'
        MUST nisMapName
        MAY description )

      ( nisSchema.2.10 NAME 'nisObject' SUP top STRUCTURAL
        DESC 'An entry in a NIS map'
        MUST ( cn $ nisMapEntry $ nisMapName )
        MAY description )

      ( nisSchema.2.11 NAME 'ieee802Device' SUP top AUXILIARY
        DESC 'A device with a MAC address; device SHOULD be
              used as a structural class'
        MAY macAddress )

      ( nisSchema.2.12 NAME 'bootableDevice' SUP top AUXILIARY
        DESC 'A device with boot parameters; device SHOULD be
              used as a structural class'
        MAY ( bootFile $ bootParameter ) )
```

5. Implementation details

5.1. Suggested resolution methods

The preferred means of directing a client application (one using the shared services of the C library) to use LDAP as its information source for the functions listed in 5.2 is to modify the source code to directly query LDAP. As the source to commercial C libraries and applications is rarely available to the end-user, one could emulate a supported nameservice (such as NIS). (This is also an appropriate opportunity to perform caching of entries across process address spaces.) In the case of NIS, reference implementations are widely available and the RPC interface is well known.

The means by which the operating system is directed to use LDAP is implementation dependent. For example, some operating systems and C libraries support end-user extensible resolvers using dynamically loadable libraries and a nameservice "switch". The means in which the DUA locates LDAP servers is also implementation dependent.

5.2. Affected library functions

The following functions are typically found in the C libraries of most UNIX and POSIX compliant systems. An LDAP search filter [RFC2254] which may be used to satisfy the function call is included alongside each function name. Parameters are denoted by %s and %d for string and integer arguments, respectively. Long lines are broken.

```
getpwnam()              (&(objectClass=posixAccount)(uid=%s))
getpwuid()              (&(objectClass=posixAccount)
                        (uidNumber=%d))
getpwent()              (objectClass=posixAccount)

getspnam()              (&(objectClass=shadowAccount)(uid=%s))
getspent()              (objectClass=shadowAccount)

getgrnam()              (&(objectClass=posixGroup)(cn=%s))
getgrgid()              (&(objectClass=posixGroup)
                        (gidNumber=%d))
getgrent()              (objectClass=posixGroup)

getservbyname()         (&(objectClass=ipService)
                        (cn=%s)(ipServiceProtocol=%s))
getservbyport()         (&(objectClass=ipService)
                        (ipServicePort=%d)
                        (ipServiceProtocol=%s))
getservent()            (objectClass=ipService)

getrpcbyname()          (&(objectClass=oncRpc)(cn=%s))
getrpcbynumber()        (&(objectClass=oncRpc)(oncRpcNumber=%d))
getrpcent()             (objectClass=oncRpc)

getprotobyname()        (&(objectClass=ipProtocol)(cn=%s))
getprotobynumber()      (&(objectClass=ipProtocol)
                        (ipProtocolNumber=%d))
getprotoent()           (objectClass=ipProtocol)

gethostbyname()         (&(objectClass=ipHost)(cn=%s))
gethostbyaddr()         (&(objectClass=ipHost)(ipHostNumber=%s))
gethostent()            (objectClass=ipHost)

getnetbyname()          (&(objectClass=ipNetwork)(cn=%s))
getnetbyaddr()          (&(objectClass=ipNetwork)
                        (ipNetworkNumber=%s))
getnetent()             (objectClass=ipNetwork)

setnetgrent()           (&(objectClass=nisNetgroup)(cn=%s))
```

5.3. Interpreting user and group entries

User and group resolution is initiated by the functions prefixed by getpw and getgr respectively. The uid attribute contains the user's login name. The cn attribute, in posixGroup entries, contains the group's name.

The account object class provides a convenient structural class for posixAccount, and SHOULD be used where additional attributes are not required.

It is suggested that uid and cn are used as the RDN attribute type for posixAccount and posixGroup entries, respectively.

An account's GECOS field is preferably determined by a value of the gecos attribute. If no gecos attribute exists, the value of the cn attribute MUST be used. (The existence of the gecos attribute allows information embedded in the GECOS field, such as a user's telephone number, to be returned to the client without overloading the cn attribute. It also accommodates directories where the common name does not contain the user's full name.)

An entry of class posixAccount, posixGroup, or shadowAccount without a userPassword attribute MUST NOT be used for authentication. The client should be returned a non-matchable password such as "x".

userPassword values MUST be represented by following syntax:

```
passwordvalue          = schemeprefix encryptedpassword
schemeprefix           = "{" scheme "}"
scheme                 = "crypt" / "md5" / "sha" / altscheme
altscheme              = "x-" keystring
encryptedpassword      = encrypted password
```

The encrypted password contains of a plaintext key hashed using the algorithm scheme.

userPassword values which do not adhere to this syntax MUST NOT be used for authentication. The DUA MUST iterate through the values of the attribute until a value matching the above syntax is found. Only if encryptedpassword is an empty string does the user have no password. DUAs are not required to consider encryption schemes which the client will not recognize; in most cases, it may be sufficient to consider only "crypt".

Below is an example of a userPassword attribute:

```
userPassword: {crypt}X5/DBrWPOQQaI
```

A future standard may specify LDAP v3 attribute descriptions to represent hashed userPasswords, as noted below. This schema MUST NOT be used with LDAP v2 DUAs and DSAs.

```
attributetype           = attributename sep attributeoption
attributename           = "userPassword"
sep                     = ";"
attributeoption         = schemeclass "-" scheme
schemeclass             = "hash" / altschemeclass
scheme                  = "crypt" / "md5" / "sha" / altscheme
altschemeclass          = "x-" keystring
altscheme               = keystring
```

Below is an example of a userPassword attribute, represented with an LDAP v3 attribute description:

```
userPassword;hash-crypt: X5/DBrWPOQQaI
```

A DUA MAY utilise the attributes in the shadowAccount class to provide shadow password service (getspnam() and getspent()). In such cases, the DUA MUST NOT make use of the userPassword attribute for getpwnam() et al, and MUST return a non-matchable password (such as "x") to the client instead.

5.4. Interpreting hosts and networks

The ipHostNumber and ipNetworkNumber attributes are defined in preference to dNSRecord (defined in [RFC1279]), in order to simplify the DUA's role in interpreting entries in the directory. A dNSRecord expresses a complete resource record, including time to live and class data, which is extraneous to this schema.

Additionally, the ipHost and ipNetwork classes permit a host or network (respectively) and all its aliases to be represented by a single entry in the directory. This is not necessarily possible if a DNS resource record is mapped directly to an LDAP entry. Implementations that wish to use LDAP to master DNS zone information are not precluded from doing so, and may simply avoid the ipHost and ipNetwork classes.

This document redefines, although not exclusively, the ipNetwork class defined in [RFC1279], in order to achieve consistent naming with ipHost. The ipNetworkNumber attribute is also used in the siteContact object class [ROSE].

The trailing zeros in a network address MUST be omitted. CIDR-style network addresses (eg. 192.168.1/24) MAY be used.

Hosts with IPv6 addresses MUST be written in their "preferred" form as defined in section 2.2.1 of [RFC1884], such that all components of the address are indicated and leading zeros are omitted. This provides a consistent means of resolving ipHosts by address.

5.5. Interpreting other entities

In general, a one-to-one mapping between entities and LDAP entries is proposed, in that each entity has exactly one representation in the DIT. In some cases this is not feasible; for example, a service which is represented in more than one protocol domain. Consider the following entry:

```
dn: cn=domain, dc=aja, dc=com
cn: domain
cn: nameserver
objectClass: top
objectClass: ipService
ipServicePort: 53
ipServiceProtocol: tcp
ipServiceProtocol: udp
```

This entry MUST map to the following two (2) services entities:

```
domain  53/tcp  nameserver
domain  53/udp  nameserver
```

While the above two entities may be represented as separate LDAP entities, with different distinguished names (such as cn=domain+ipServiceProtocol=tcp, ... and cn=domain+ipServiceProtocol=udp, ...) it is convenient to represent them as a single entry. (If a service is represented in multiple protocol domains with different ports, then multiple entries are required; multivalued RDNs may be used to distinguish them.)

With the exception of userPassword values, which are parsed according to the syntax considered in section 5.2, any empty values (consisting of a zero length string) are returned by the DUA to the client. The DUA MUST reject any entries which do not conform to the schema (missing mandatory attributes). Non-conforming entries SHOULD be ignored while enumerating entries.

The nisObject object class MAY be used as a generic means of representing NIS entities. Its use is not encouraged; where support for entities not described in this schema is desired, an appropriate

schema should be devised. Implementors are strongly advised to support end-user extensible mappings between NIS entities and object classes. (Where the nisObject class is used, the nisMapName attribute may be used as a RDN.)

5.6. Canonicalizing entries with multi-valued naming attributes

For entities such as hosts, services, networks, protocols, and RPCs, where there may be one or more aliases, the respective entry's relative distinguished name SHOULD be used to determine the canonical name. Any other values for the same attribute are used as aliases. For example, the service described in section 5.5 has the canonical name "domain" and exactly one alias, "nameserver".

The schema in this document generally only defines one attribute per class which is suitable for distinguishing an entity (excluding any attributes with integer syntax; it is assumed that entries will be distinguished on name). Usually, this is the common name (cn) attribute. This aids the DUA in determining the canonical name of an entity, as it can examine the value of the relative distinguished name. Aliases are thus any values of the distinguishing attribute (such as cn) which do not match the canonical name of the entity.

In the event that a different attribute is used to distinguish the entry, as may be the case where these object classes are used as auxiliary classes, the entry's canonical name may not be present in the RDN. In this case, the DUA MUST choose one of the non-distinguished values to represent the entity's canonical name. As the directory server guarantees no ordering of attribute values, it may not be possible to distinguish an entry deterministically. This ambiguity SHOULD NOT be resolved by mapping one directory entry into multiple entities.

6. Implementation focus

A NIS server which uses LDAP instead of local files has been developed which supports the schema defined in this document.

A reference implementation of the C library resolution code has been written for the Free Software Foundation. It may support other C libraries which support the Name Service Switch (NSS) or the Information Retrieval Service (IRS).

The author has made available a freely distributable set of scripts which parses local databases such as /etc/passwd and /etc/hosts into a form suitable for loading into an LDAP server.

7. Security Considerations

The entirety of related security considerations are outside the scope of this document. It is noted that making passwords encrypted with a widely understood hash function (such as crypt()) available to non-privileged users is dangerous because it exposes them to dictionary and brute-force attacks. This is proposed only for compatibility with existing UNIX system implementations. Sites where security is critical SHOULD consider using a strong authentication service for user authentication.

Alternatively, the encrypted password could be made available only to a subset of privileged DUAs, which would provide "shadow" password service to client applications. This may be difficult to enforce.

Because the schema represents operating system-level entities, access to these entities SHOULD be granted on a discretionary basis. (There is little point in restricting access to data which will be republished without restriction, however.) It is particularly important that only administrators can modify entries defined in this schema, with the exception of allowing a principal to change their password (which may be done on behalf of the user by a client bound as a superior principal, such that password restrictions may be enforced). For example, if a user were allowed to change the value of their uidNumber attribute, they could subvert security by equivalencing their account with the superuser account.

A subtree of the DIT which is to be republished by a DUA (such as a NIS gateway) SHOULD be within the same administrative domain that the republishing DUA represents. (For example, principals outside an organization, while conceivably part of the DIT, should not be considered with the same degree of authority as those within the organization.)

Finally, care should be exercised with integer attributes of a sensitive nature (particularly the uidNumber and gidNumber attributes) which contain zero-length values. DUAs MAY treat such values as corresponding to the "nobody" or "nogroup" user and group, respectively.

8. Acknowledgements

Thanks to Leif Hedstrom of Netscape Communications Corporation, Michael Grant and Rosanna Lee of Sun Microsystems Inc., Ed Reed of Novell Inc., and Mark Wahl of Critical Angle Inc. for their valuable contributions to the development of this schema. Thanks to Andrew Josey of The Open Group for clarifying the use of the UNIX trademark, and to Tim Howes and Peter J. Cherny for their support.

UNIX is a registered trademark of The Open Group.

9. References

[RFC1057]
Sun Microsystems, Inc., "RPC: Remote Procedure Call: Protocol Specification Version 2", RFC 1057, June 1988.

[RFC1279]
Kille, S., "X.500 and Domains", RFC 1279, November 1991.

[RFC1884]
Hinden, R., and S. Deering, "IP Version 6 Addressing Architecture", RFC 1884, December 1995.

[RFC2119]
Bradner, S., "Key Words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2251]
Wahl, M., Howes, T., and S. Kille, "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997.

[RFC2252]
Wahl, M., Coulbeck, A., Howes, T., and S. Kille, "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions", RFC 2252, December 1997.

[RFC2254]
Howes, T., "The String Representation of LDAP Search Filters", RFC 2254, December 1997.

[RFC2256]
Wahl, M., "A Summary of the X.500(96) User Schema for use with LDAPv3", RFC 2256, December 1997.

[ROSE]
M. T. Rose, "The Little Black Book: Mail Bonding with OSI Directory Services", ISBN 0-13-683210-5, Prentice-Hall, Inc., 1992.

[X500]
"Information Processing Systems - Open Systems Interconnection - The Directory: Overview of Concepts, Models and Service", ISO/IEC JTC 1/SC21, International Standard 9594-1, 1988.

[XOPEN]
ISO/IEC 9945-1:1990, Information Technology - Portable Operating Systems Interface (POSIX) - Part 1: Systems Application Programming Interface (API) [C Language]

10. Author's Address

Luke Howard
PO Box 59
Central Park Vic 3145
Australia

EMail: lukeh@xedoc.com

A. Example entries

The examples described in this section are provided to illustrate the schema described in this memo. They are not meant to be exhaustive.

The following entry is an example of the posixAccount class:

```
dn: uid=lester, dc=aja, dc=com
objectClass: top
objectClass: account
objectClass: posixAccount
uid: lester
cn: Lester the Nightfly
userPassword: {crypt}X5/DBrWPOQQaI
gecos: Lester
loginShell: /bin/csh
uidNumber: 10
gidNumber: 10
homeDirectory: /home/lester
```

This corresponds the UNIX system password file entry:

```
lester:X5/DBrWPOQQaI:10:10:Lester:/home/lester:/bin/sh
```

The following entry is an example of the ipHost class:

```
dn: cn=peg.aja.com, dc=aja, dc=com
objectClass: top
objectClass: device
objectClass: ipHost
objectClass: bootableDevice
objectClass: ieee802Device
cn: peg.aja.com
cn: www.aja.com
ipHostNumber: 10.0.0.1
macAddress: 00:00:92:90:ee:e2
bootFile: mach
bootParameter: root=fs:/nfsroot/peg
bootParameter: swap=fs:/nfsswap/peg
bootParameter: dump=fs:/nfsdump/peg
```

This entry represents the host canonically peg.aja.com, also known as www.aja.com. The Ethernet address and four boot parameters are also specified.

An example of the nisNetgroup class:

```
dn: cn=nightfly, dc=aja, dc=com
objectClass: top
objectClass: nisNetgroup
cn: nightfly
nisNetgroupTriple: (charlemagne,peg,dunes.aja.com)
nisNetgroupTriple: (lester,-,)
memberNisNetgroup: kamakiriad
```

This entry represents the netgroup nightfly, which contains two triples (the user charlemagne, the host peg, and the domain dunes.aja.com; and, the user lester, no host, and any domain) and one netgroup (kamakiriad).

Finally, an example of the nisObject class:

```
dn: nisMapName=tracks, dc=dunes, dc=aja, dc=com
objectClass: top
objectClass: nisMap
nisMapName: tracks

dn: cn=Maxine, nisMapName=tracks, dc=dunes, dc=aja, dc=com
objectClass: top
objectClass: nisObject
cn: Maxine
nisMapName: tracks
nisMapEntry: Nightfly$4
```

This entry represents the NIS map tracks, and a single map entry.

Full Copyright Statement