Network Working Group
Internet-Draft
Obsoletes: 2307 (if approved)
Intended status: Informational
Expires: February 10, 2010

L. Howard
PADL Software
H. Chu, Ed.
Symas Corp.

August 9, 2009

An Approach for Using LDAP as a Network Information Service
draft-howard-rfc2307bis-02.txt

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on February 10, 2010.

Copyright Notice

Abstract

This document describes a mechanism for mapping entities related to TCP/IP and the UNIX system [UNIX] into [X.500] entries so that they may be resolved with the Lightweight Directory Access Protocol [RFC4511]. 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 [UNIX]) 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.

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) [RFC1034]. (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 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 the 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. Schema

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

2.2.1. Attributes

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
- nisPublicKey
- nisSecretKey
- nisDomain
- automountMapName
- automountKey
- automountInformation

Additionally, some of the attributes defined in [RFC4519] and [RFC3112] are required.

2.2.2. Attribute Option

Centralizing a name service in LDAP allows heterogeneous systems to obtain their information from a single place. However in some cases, this information cannot be used uniformly on all of the client systems. These attribute options are defined to allow system-specific values to be used where necessary. The options are defined as the following:

```
host-<hostname>
hostos-<ostype>
```

where hostname is a string representing the name of a specific machine, and ostype is a string representing a particular operating system.

For example, a user named "Babs Jensen" may have a different home directory on different machines. This could be represented as:

```
homeDirectory: /home/babsj
homeDirectory;hostos-sunos: /export/home/bjensen
homeDirectory;host-babshost: /home/root
```

These attribute options follow sub-typing semantics. If a DUA requests an attribute to be returned in a search operation, and does not specify an option, all subtypes of that attribute are returned.

2.2.3. Object Classes

The following object classes are defined in this document:

```
posixAccount
shadowAccount
posixGroup
ipService
ipProtocol
oncRpc
ipHost
ipNetwork
nisNetgroup
nisMap
nisObject
ieee802Device
bootableDevice
nisKeyObject
nisDomainObject
automountMap
automount
```

Additionally, some of the attributes defined in [RFC4519] are required.

3. Attribute Definitions

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

```
( 1.3.6.1.1.1.1.0 NAME 'uidNumber'
    DESC 'An integer uniquely identifying a user in an
          administrative domain'
    EQUALITY integerMatch
    ORDERING integerOrderingMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
    SINGLE-VALUE )


( 1.3.6.1.1.1.1.1 NAME 'gidNumber'
    DESC 'An integer uniquely identifying a group in an
          administrative domain'
    EQUALITY integerMatch
    ORDERING integerOrderingMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
    SINGLE-VALUE )


( 1.3.6.1.1.1.1.2 NAME 'gecos'
    DESC 'The GECOS field; the common name'
    EQUALITY caseIgnoreMatch
    SUBSTRINGS caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
    SINGLE-VALUE )


( 1.3.6.1.1.1.1.3 NAME 'homeDirectory'
    DESC 'The absolute path to the home directory'
    EQUALITY caseExactIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
    SINGLE-VALUE )


( 1.3.6.1.1.1.1.4 NAME 'loginShell'
    DESC 'The path to the login shell'
    EQUALITY caseExactIA5Match
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
    SINGLE-VALUE )
```

```
    ( 1.3.6.1.1.1.1.5 NAME 'shadowLastChange'
        EQUALITY integerMatch
        ORDERING integerOrderingMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        SINGLE-VALUE )

    ( 1.3.6.1.1.1.1.6 NAME 'shadowMin'
        EQUALITY integerMatch
        ORDERING integerOrderingMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        SINGLE-VALUE )

    ( 1.3.6.1.1.1.1.7 NAME 'shadowMax'
        EQUALITY integerMatch
        ORDERING integerOrderingMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        SINGLE-VALUE )

    ( 1.3.6.1.1.1.1.8 NAME 'shadowWarning'
        EQUALITY integerMatch
        ORDERING integerOrderingMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        SINGLE-VALUE )

    ( 1.3.6.1.1.1.1.9 NAME 'shadowInactive'
        EQUALITY integerMatch
        ORDERING integerOrderingMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        SINGLE-VALUE )

    ( 1.3.6.1.1.1.1.10 NAME 'shadowExpire'
        EQUALITY integerMatch
        ORDERING integerOrderingMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        SINGLE-VALUE )

    ( 1.3.6.1.1.1.1.11 NAME 'shadowFlag'
        EQUALITY integerMatch
        ORDERING integerOrderingMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        SINGLE-VALUE )
```

```
( 1.3.6.1.1.1.1.12 NAME 'memberUid'
    EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )


( 1.3.6.1.1.1.1.13 NAME 'memberNisNetgroup'
    EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )


( 1.3.6.1.1.1.1.14 NAME 'nisNetgroupTriple'
    DESC 'Netgroup triple'
    EQUALITY caseIgnoreMatch
    SUBSTRINGS caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )


( 1.3.6.1.1.1.1.15 NAME 'ipServicePort'
    DESC 'Service port number'
    EQUALITY integerMatch
    ORDERING integerOrderingMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
    SINGLE-VALUE )


( 1.3.6.1.1.1.1.16 NAME 'ipServiceProtocol'
    DESC 'Service protocol name'
    EQUALITY caseIgnoreMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )


( 1.3.6.1.1.1.1.17 NAME 'ipProtocolNumber'
    DESC 'IP protocol number'
    EQUALITY integerMatch
    ORDERING integerOrderingMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
    SINGLE-VALUE )


( 1.3.6.1.1.1.1.18 NAME 'oncRpcNumber'
    DESC 'ONC RPC number'
    EQUALITY integerMatch
    ORDERING integerOrderingMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
    SINGLE-VALUE )
```

```
    ( 1.3.6.1.1.1.1.19 NAME 'ipHostNumber'
        DESC 'IPv4 addresses as a dotted decimal omitting leading
              zeros or IPv6 addresses as defined in RFC2373'
        EQUALITY caseIgnoreIA5Match
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )


    ( 1.3.6.1.1.1.1.20 NAME 'ipNetworkNumber'
        DESC 'IP network omitting leading zeros, eg. 192.168'
        EQUALITY caseIgnoreIA5Match
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
        SINGLE-VALUE )


    ( 1.3.6.1.1.1.1.21 NAME 'ipNetmaskNumber'
        DESC 'IP netmask omitting leading zeros, eg. 255.255.255.0'
        EQUALITY caseIgnoreIA5Match
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
        SINGLE-VALUE )


    ( 1.3.6.1.1.1.1.22 NAME 'macAddress'
        DESC 'MAC address in maximal, colon separated hex
              notation, eg. 00:00:92:90:ee:e2'
        EQUALITY caseIgnoreIA5Match
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )


    ( 1.3.6.1.1.1.1.23 NAME 'bootParameter'
        DESC 'rpc.bootparamd parameter'
        EQUALITY caseExactIA5Match
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )


    ( 1.3.6.1.1.1.1.24 NAME 'bootFile'
        DESC 'Boot image name'
        EQUALITY caseExactIA5Match
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )


    ( 1.3.6.1.1.1.1.26 NAME 'nisMapName'
        DESC 'Name of a generic NIS map'
        EQUALITY caseIgnoreMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} )
```

```
    ( 1.3.6.1.1.1.1.27 NAME 'nisMapEntry'
        DESC 'A generic NIS entry'
        EQUALITY caseExactMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{1024}
        SINGLE-VALUE )


    ( 1.3.6.1.1.1.1.28 NAME 'nisPublicKey'
        DESC 'NIS public key'
        EQUALITY octetStringMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
        SINGLE-VALUE )


    ( 1.3.6.1.1.1.1.29 NAME 'nisSecretKey'
        DESC 'NIS secret key'
        EQUALITY octetStringMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
        SINGLE-VALUE )


    ( 1.3.6.1.1.1.1.30 NAME 'nisDomain'
        DESC 'NIS domain'
        EQUALITY caseIgnoreIA5Match
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )


    ( 1.3.6.1.1.1.1.31 NAME 'automountMapName'
        DESC 'automount Map Name'
        EQUALITY caseExactMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE )


    ( 1.3.6.1.1.1.1.32 NAME 'automountKey'
        DESC 'Automount Key value'
        EQUALITY caseExactMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE )


    ( 1.3.6.1.1.1.1.33 NAME 'automountInformation'
        DESC 'Automount information'
        EQUALITY caseExactMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE )
```

4. Class Definitions

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

Various schema for mail routing and delivery using LDAP directories have been proposed, and as such this document does not consider schema for representing mail aliases or distribution lists.

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


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


( 1.3.6.1.1.1.2.2 NAME 'posixGroup' SUP top AUXILIARY
    DESC 'Abstraction of a group of accounts'
    MUST gidNumber
    MAY ( authPassword $ userPassword $ memberUid $
          description ) )


( 1.3.6.1.1.1.2.3 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 )


( 1.3.6.1.1.1.2.4 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 canonical name'
    MUST ( cn $ ipProtocolNumber )
    MAY description )
```

```
( 1.3.6.1.1.1.2.5 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 canonical name'
    MUST ( cn $ oncRpcNumber )
    MAY description )


( 1.3.6.1.1.1.2.6 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 ( authPassword $ userPassword $ l $ description $
          manager ) )


( 1.3.6.1.1.1.2.7 NAME 'ipNetwork' SUP top STRUCTURAL
    DESC 'Abstraction of a network. The distinguished value of
          the cn attribute denotes the network canonical name'
    MUST ipNetworkNumber
    MAY ( cn $ ipNetmaskNumber $ l $ description $ manager ) )


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


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


( 1.3.6.1.1.1.2.10 NAME 'nisObject' SUP top STRUCTURAL
    DESC 'An entry in a NIS map'
    MUST ( cn $ nisMapEntry $ nisMapName )


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

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


( 1.3.6.1.1.1.2.14 NAME 'nisKeyObject' SUP top AUXILIARY
    DESC 'An object with a public and secret key'
    MUST ( cn $ nisPublicKey $ nisSecretKey )
    MAY ( uidNumber $ description ) )


( 1.3.6.1.1.1.2.15 NAME 'nisDomainObject' SUP top AUXILIARY
    DESC 'Associates a NIS domain with a naming context'
    MUST nisDomain )


( 1.3.6.1.1.1.2.16 NAME 'automountMap' SUP top STRUCTURAL
    MUST ( automountMapName )
    MAY description )


( 1.3.6.1.1.1.2.17 NAME 'automount' SUP top STRUCTURAL
    DESC 'Automount information'
    MUST ( automountKey $ automountInformation )
    MAY description )


( 1.3.6.1.1.1.2.18 NAME 'groupOfMembers' SUP top STRUCTURAL
    DESC 'A group with members (DNs)'
    MUST cn
    MAY ( businessCategory $ seeAlso $ owner $ ou $ o $
          description $ member ) )
```

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 Appendix B 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" (NSS). The means in which the DUA locates LDAP servers is also implementation dependent.

5.2. 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. This document preserves the use of the uid attribute even though, being a naming attribute, its syntax is case insensitive. This may cause a problem with existing deployments where there exist login names differing only in case. If DUAs support attribute mapping, a different attribute MAY be used to represent users' login names.

The account object class provides a convenient structural class for posixAccount, and SHOULD be used where additional attributes are not required. Likewise, the groupOfMembers object class SHOULD be used for groups. (The groupOfUniqueNames object class is deprecated and SHOULD NOT be used.)

It is suggested that uid and cn are used as the naming attribute for posixAccount and posixGroup entries, respectively. Group members may either be login names (values of memberUid) or distinguished names (values of member). In the latter case, the distinguished name must be mapped to one or more login names by examining the name's RDN or, if it is not distinguished by uid, performing a base search on the DN with a filter of "(objectclass=*)". DUAs MAY wish to cache DN to login name mappings. The DUA MAY traverse nested groups.

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.)

5.2.1. Using Attribute Options

Some posixAccount attributes may be accompanied by options (Section 2.2.2) identifying particular hosts or operating system types. The attribute with a hostos option matching the operating system of the DUA SHOULD be used in preference to an attribute without any associated options. The attribute with a host option matching the hostname of the DUA SHOULD be used in preference to any other attribute.

5.2.2. Authentication Considerations

5.2.2.1. Using Password Values

When authenticating using a NIS to LDAP gateway or using NSS, a lookup is performed to retrieve the password attribute and the value is used in the DUA to authenticate a user. This approach to authentication is deprecated, as it requires that read access to the password attribute be granted across a network.

An entry of class posixAccount, posixGroup, or shadowAccount without an authPassword or userPassword attribute MUST NOT be used for authentication. In this case the client SHOULD be returned a non-matchable password such as "x".

If userPassword is used, its values MUST be represented by the following syntax:

```
passwordvalue   = schemeprefix hashedpasswd
schemeprefix    = "{" scheme "}"
scheme          = "crypt" / "md5" / "sha" / "ssha" / altscheme
altscheme       = "x-" keystring
hashedpasswd    = hashed password
```

The hashed password contains a plaintext key hashed using the algorithm scheme. If the schema is "sha", the hashed password is the base64 encoding of the SHA-1 digest of the plaintext password.

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 hashedpassword is an empty string does the user have no password. DUAs are not required to consider hashing schemes which the client will not recognize; in most cases, it may be sufficient to consider only "crypt".

DUA MAY use the authPassword attribute instead of userPassword, defined in [RFC3112]. The DUA MUST iterate the values of the authPassword attribute until a value whose scheme is CRYPT is found. The DUA MAY iterate through the values of the userPassword attribute, using the syntax defined here, until a value whose scheme is CRYPT is found. If no conforming value is found, the client MUST be returned a non-matchable password such as "x". Authentication using schemes other than CRYPT is, although advisable, beyond the scope of this document

Below is an example of an authPassword attribute:

```
authPassword: CRYPT$X5/DBrWPOQQaI
```

Below is an example of a (deprecated) userPassword attribute:

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

A DUA MAY utilize 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.

5.2.2.2. Using LDAP Bind

The preferred method for authenticating users with LDAP is to perform an LDAP Bind operation with the user's name and password. In this case the method the DSA uses to store and verify the password is completely internal to the DSA and not of any concern to the DUA.

Likewise, using the shadowAccount attributes requires the DUA to handle password policy enforcement. However the policies expressed in the shadowAccount are limited in scope, and not uniformly applicable to all the systems that will be using LDAP.

The preferred method is to leave password policy enforcement to the DSA, so that it will be uniformly enforced across all of the various systems that rely on the directory. This enforcement occurs implicitly when authenticating using LDAP Bind if the DSA supports the LDAP password policy [I-D.behera-ldap-password-policy] mechanisms.

The means in which authentication in the DUA is configured is implementation dependent. Typically it is accomplished using [PAM]. Further details of authentication are beyond the scope of this document.

5.2.3. Naming Considerations

On UNIX systems, users and groups reside in separate name spaces and it is common for the same name to be used by both a user and a group. Since they are in separate name spaces there is no ambiguity and no conflict. However, when integrating a name service into LDAP the directory may be used with other systems besides UNIX and its derivatives. In particular, the Microsoft Windows operating system may also use LDAP for its own name service. In Windows, users and groups reside in a single name space and so one cannot use the same name for both a user and a group.

Conflicts in naming conventions may arise in other deployments as well, and should be carefully taken into account when migrating from other naming services into LDAP.

5.3. 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 authPassword and userPassword attributes are included in ipHost such that hosts MAY be treated as authentication principals. The treatment of these attributes and inherent caveats considered in Section 5.2 apply here also.

The trailing zeros in a network address MUST be omitted. CIDR-style

network addresses (eg. 192.168.1/24) MAY be used. Leading zeros MUST be removed from all components of an IPv6 address string as defined by [RFC2373], section 2.2, item 1. The IPv6 address string MUST be further normalized by following the "::" syntax as defined in [RFC2373], section 2.2, item 2. In addition, "::" MUST be used to replace the longest string of zero bits. If there are two or more longest strings of zero bits, then the first string MUST be replaced. In addition, the syntax defined by [RFC2373], section 2.2, item 3 MUST NOT be used. IPv4 addresses MUST be represented by the IPv4 dotted decimal string syntax.

For example the following address:

```
1080:0000:0:0:08:800:200C:417A
FF01:0:0:0:0:0:01
0:0:0:0:0:0:0:0001
0:0:0:0:0:0:0:0
```

MUST be normalized as:

```
1080::8:800:200C:417A
FF01::101
0::1
::
```

5.4. 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,ou=services,dc=aja,dc=com
objectClass: top
objectClass: ipService
cn: domain
cn: nameserver
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; multi-valued RDNs MAY be used to distinguish them.)

With the exception of authPassword and userPassword values, 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 nisDomainObject object class is provided to associate a NIS domain with a naming context. A DUA would retrieve the NIS domain name from a configuration file and enumerate the naming contexts served by an LDAP server, searching each naming context for (nisDomain=%s). The first matching entry that is found MAY be used as a search base for configuration profile information or for entries themselves. For example, the following example shows an association between the NIS domain "nis.aja.com" and the naming context "dc=aja,dc=com":

```
dn: dc=aja,dc=com
objectClass: top
objectClass: domain
objectClass: nisDomainObject
dc: aja
nisDomain: nis.aja.com
```

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. Implementers 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.) The nisObject class might be used to represent automount information.

5.5. 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 5.4 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

Gateways between NIS and LDAP have been developed by PADL Software and Sun Microsystems. They both support this schema.

An open source implementation of the C library resolution code has been written and is available from PADL Software. It supports C libraries on GNU, BSD, AIX, and Solaris operating systems. PADL have also made available a set of scripts for migrating flat files into a form suitable for loading into an LDAP server. Another open source implementation of the C library code is available from the OpenLDAP Project.

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 Bob Joslin of the Hewlett Packard Company, and to all those that helped with this document's predecessor, RFC 2307.

UNIX is a registered trademark of The Open Group.

9. References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

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

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

[RFC2373] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 2373, July 1998.

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

[RFC4511] Sermersheim, J., "Lightweight Directory Access Protocol (LDAP): The Protocol", RFC 4511, June 2006.

[RFC4515] Smith, M. and T. Howes, "Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters", RFC 4515, June 2006.

[RFC4519] Sciberras, A., "Lightweight Directory Access Protocol (LDAP): Schema for User Applications", RFC 4519, June 2006.

[RFC3112] Zeilenga, K., "LDAP Authentication Password Schema", RFC 3112, May 2001.

[I-D.behera-ldap-password-policy] Sermersheim, J., Poitou, L., and H. Chu, "Password Policy for LDAP Directories", draft-behera-ldap-password-policy-10 (work in progress), August 2009.

[ROSE] Rose, M., "The Little Black Book: Mail Bonding with OSI Directory Services", ISBN 0-13-683210-5, 2001.

[X.500] ISO/IEC JTC 1/SC21, "Information Processing Systems - Open Systems Interconnection - The Directory: Overview of Concepts, Models and Service", 1988.

[UNIX] Institute of Electrical and Electronics Engineers and The Open Group, "IEEE Std 1003.1, 2004 Edition, Single UNIX Specification Version 3", IEEE Standard 1003.1, 2004.

[PAM] Samar, V. and R. Schemers, "Unified Login with Pluggable Authentication Modules (PAM)", OSF RFC 86.0, October 1995.

Appendix A. Example Entries

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

The following entry is an example of the posixAccount class:

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

This corresponds to 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=josie.aja.com,ou=hosts,dc=aja,dc=com
objectClass: top
objectClass: device
objectClass: ipHost
objectClass: bootableDevice
objectClass: ieee802Device
cn: josie.aja.com
cn: www.aja.com
ipHostNumber: 10.0.0.1
macAddress: 00:00:92:90:ee:e2
bootFile: mach
bootParameter: root=dan.aja.com:/nfsroot/peg
bootParameter: swap=dan.aja.com:/nfsswap/peg
bootParameter: dump=dan.aja.com:/nfsdump/peg
```

This entry represents the host canonically josie.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,ou=netgroup,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 represents the NIS map tracks, and a single map entry.

Appendix B. Affected Library Functions

The following functions are typically found in the C libraries of most UNIX and POSIX compliant systems [UNIX]. An LDAP search filter string [RFC4515] 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))
    getpublickey()      (&(objectClass=nisKeyObject)(...))
```

Appendix C. Suggested DIT structure

The cn attribute is typically used to name entities. The ipHostNumber, ipNetworkNumber, and ipServiceProtocol attributes are also naming attributes, such that multi-valued RDNs may be used to distinguish between, for example, different interfaces of a multihomed host.

The following DIT structure MAY be used for deploying this schema. It is not required that DC-naming be used, but it is encouraged:

Naming context	ObjectClass
ou=people,dc=...	posixAccount
	shadowAcount
ou=group,dc=...	posixGroup
ou=services,dc=...	ipService
ou=protocols,dc=...	ipProtocol
ou=rpc,dc=...	oncRpc
ou=hosts,dc=...	ipHost
ou=ethers,dc=...	ieee802Device
	bootableDevice
ou=networks,dc=...	ipNetwork
ou=netgroup,dc=...	nisNetgroup
nisMapName=...,dc=...	nisObject
automountMapName=...,dc=...	automountMap

Authors' Addresses

Luke Howard
PADL Software Pty. Ltd.
PO Box 59
Central Park, Vic 3145
AU

Email: lukeh@padl.com

Howard Chu (editor)
Symas Corp.
18740 Oxnard Street, Suite 313A
Tarzana, California 91356
US

Phone: +1 818 757-7087
Email: hyc@symas.com