Defining and Locating IPv6 Address Blocks using the Internet Resource Query Service

1. Abstract

This document defines LDAP schema and searching rules for IPv6 addresses and address blocks, in support of the Internet Resource Query Service described in [ldap-whois].

2. Definitions and Terminology

This document unites, enhances and clarifies several pre-existing technologies. Readers are expected to be familiar with the following specifications:

- RFC 2247 - Using Domains in LDAP/X.500 DNs
- RFC 2254 - The String Representation of LDAP Search Filters
- RFC 2256 - A Summary of the X.500(96) User Schema for use
with LDAPv3

RFC 2798 - Definition of the inetOrgPerson LDAP Object Class

[namedref] - <draft-zeilenga-ldap-namedref-04.txt> - Named Subordinate References in LDAP Directories

[ir-dir-req] - <draft-newton-ir-dir-requirements-00.txt> - Internet Registry Directory Requirements

The following abbreviations are used throughout this document:

DIT (Directory Information Tree) - A DIT is a contained branch of the LDAP namespace, having a root of a particular distinguished name. "dc=example,dc=com" is used throughout this document as one DIT, with many example entries being stored in this DIT.

DN (Distinguished Name) - A distinguished name provides a unique identifier for an entry, through the use of a multi-level naming syntax. Entries are named according to their location relevant to the root of their containing DIT. For example, "cn/inetResources,dc=example,dc=com" is a DN which uniquely identifies the "inetResources" entry within the "dc=example,dc=com" DIT.

RDN (Relative DN) - An RDN provides a locally-scoped unique identifier for an entry. A complete, globally-unique DN is
formed by concatenating the RDNs of an entry together. For example, "cn=admins,cn/inetResources,dc=example,dc=com" consists of two RDNs ("cn=admins" and "cn/inetResources") within the "dc=example,dc=com" DIT. RDNs are typically only referenced within their local scope.

OID (Object Identifier) - An OID is a globally-unique, concatenated set of integers which provide a kind of "serial number" to attributes, object classes, syntaxes and other schema elements.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

3. The inetIpv6Network Object Class

The inetIpv6Network object class is a structural object class which provides administrative information about a specific IPv6 address and an associated subnet prefix (this pairing is most often used to represent the starting address of an IPv6 network, but can also be used to identify a specific host).

3.1. Naming syntax

The naming syntax for IPv6 network entries MUST follow the form of "cn=<inetIpv6NetworkSyntax>,cn/inetResources,<dc-DIT>". Each IPv6 network address which is managed as a discrete LDAP-WHOIS network resource MUST have a dedicated entry in each of the DITs which provide public LDAP-WHOIS data regarding that network address.

The inetIpv6NetworkSyntax component of an entry is subject to DN rules, although the inetIpv6NetworkSyntax is also used for extended search operations, and is therefore subject to specific syntax rules. This syntax specifically requires the use of the starting address from a range of inclusive addresses, and specifically requires the use of the common IPv6 prefix annotation. In this manner, it is possible to create an inetIpv6Network entry for a range of addresses (by specifying the starting address and the network prefix size), or a single host (by specifying the host-specific address and a /128 prefix).

In this definition, the inetIpv6NetworkSyntax uses the uncompressed, 32-nibble IPv6 addressing syntax, where the network
address consists of eight sub-components, with each sub-component consisting of four hexadecimal values that represent one nibble, with each sub-component being separated by a colon character, and with the entire sequence being followed by a "/" character and a three-digit decimal "prefix" value. An augmented BNF for this syntax is as follows:

\[
\]

\[
vSixPart = 4*4nibblePart
\]

\[
nibblePart = \text{hexadecimal digit between "0" and "F" inclusive}
\]

\[
vSixPrefix = \text{decimal value between "1" and "128" inclusive, with the non-affective leading zeroes removed}
\]

For example, an IPv6 network with a range of addresses between "3ffe:ffff::" and "3ffe:ffff:ffff:ffff:ffff:ffff:ffff:ffff" inclusive would have a RDN of "cn=3ffe:ffff:0000:0000:0000:0000:0000:0000/32". Similarly, a host address of "3ffe:ffff::1:2:3:4" would have the RDN of "cn=3ffe:ffff:0000:0000:0001:0002:0003:0004/128".

Each of the 16-bit colon-separated values MUST be written in the uncompressed form. Nibbles with a value of zero MUST be represented by the hexadecimal sequence of "0000".

Note that the use of "/" is illegal in LDAP URLs when it is provided as data (in particular, URLs use this character as a part delimiter). This character MUST be escaped as "%2F" when it is provided as part of an inetIpv6Network entry in a ref attribute.

3.2. Schema Definition

IPv6 network entries MUST exist with the top, inetResources and inetIpv6Network object classes defined. If an entry exists as a referral, the entry MUST also be defined with the referral object class, in addition to the above requirements.

The inetIpv6Network object class is a structural object class which is subordinate to the inetResources object class, and which MUST be treated as a container class capable of holding additional subordinate entries. The inetIpv6Network object class has no
mandatory attributes, although it does have several optional attributes.

The inetIpv6Network object class defines attributes which are specific to IPv6 networks, such as the delegation date and the status of the delegation. The inetIpv6Network object class is subordinate to the inetResources object class, so it inherits those attributes as well.

Some of the inetIpv6Network object class attributes define contact-related referrals which provide LDAP URLs that refer to inetOrgPerson entries, and these entries will need to be queried separately if detailed information about a particular contact is required. The contact attribute values follow the same rules as the labeledURI attribute defined in RFC 2079, with additional restrictions described in [ldap-whois].

The various ModifiedBy and ModifiedDate attributes SHOULD be treated as operational attributes. Their values SHOULD be filled in automatically by the database management application, and SHOULD NOT be returned except when explicitly requested.

The schema definition for the inetIpv6Network object class is as follows:

```
inetIpv6Network
( 1.3.6.1.4.1.7161.1.3.0 NAME 'inetIpv6Network' DESC 'IPv6 network attributes.' SUP inetResources STRUCTURAL MAY ( inetIpv6DelegationStatus $ inetIpv6DelegationDate $ inetIpv6DelegationModifiedDate $ inetIpv6DelegationModifiedBy $ inetIpv6Contacts $ inetIpv6ContactsModifiedBy $ inetIpv6ContactsModifiedDate $ inetIpv6RoutingContacts $ inetIpv6RoutingContactsModifiedBy $ inetIpv6RoutingContactsModifiedDate ) )
```

The attributes from the inetIpv6Network object class are described below:

```
inetIpv6Contacts
( 1.3.6.1.4.1.7161.1.3.2 NAME 'inetIpv6Contacts' DESC 'Contacts for reporting problems with this network.' EQUALITY caseExactMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```
inetIpv6ContactsModifiedBy
( 1.3.6.1.4.1.7161.1.3.3 NAME 'inetIpv6ContactsModifiedBy'
  DESC 'Person who last modified the inetIpv6Contacts
  attribute.' EQUALITY distinguishedNameMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE USAGE
  distributedOperation )

inetIpv6ContactsModifiedDate
( 1.3.6.1.4.1.7161.1.3.4 NAME 'inetIpv6ContactsModifiedDate'
  DESC 'Last modification date of the inetIpv6Contacts
  attribute.' EQUALITY generalizedTimeMatch ORDERING
  generalizedTimeOrderingMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE USAGE
  distributedOperation )

inetIpv6DelegationDate
( 1.3.6.1.4.1.7161.1.3.5 NAME 'inetIpv6DelegationDate' DESC
  'Date of original delegation.' EQUALITY
  generalizedTimeMatch ORDERING generalizedTimeOrderingMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE )

inetIpv6DelegationModifiedBy
( 1.3.6.1.4.1.7161.1.3.6 NAME 'inetIpv6DelegationModifiedBy'
  DESC 'Person who last modified the inetIpv6DelegationStatus
  attribute.' EQUALITY distinguishedNameMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE USAGE
  distributedOperation )

inetIpv6DelegationModifiedDate
( 1.3.6.1.4.1.7161.1.3.7 NAME
  'inetIpv6DelegationModifiedDate' DESC 'Last modification
  date of the inetIpv6DelegationStatus attribute.' EQUALITY
  generalizedTimeMatch ORDERING generalizedTimeOrderingMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE USAGE
  distributedOperation )

inetIpv6DelegationStatus
( 1.3.6.1.4.1.7161.1.3.8 NAME 'inetIpv6DelegationStatus' DESC
  'Current delegation status code for this network.' EQUALITY
  numericStringMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.27(2)
  SINGLE-VALUE )

NOTE: In an effort to facilitate internationalization and
programmatic processing, the current status of a delegation
is identified by a 16-bit integer. The values and status
mapping is as follows:
0 Reserved delegation (permanently inactive)
1 Assigned and active (normal state)
2 Assigned but not yet active (new delegation)
3 Assigned but on hold (disputed)
4 Assignment revoked (database purge pending)

Additional values for the inetIpv6DelegationStatus attribute are reserved for future use, and are to be administered by IANA. Note that there is no status code for "unassigned"; unassigned entries SHOULD NOT exist, and SHOULD NOT be returned as answers.

inetIpv6RoutingContacts
( 1.3.6.1.4.1.7161.1.3.9 NAME 'inetIpv6RoutingContacts' DESC 'Contacts for routing issues with this network.' EQUALITY caseExactMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

inetIpv6RoutingContactsModifiedBy
( 1.3.6.1.4.1.7161.1.3.10 NAME 'inetIpv6RoutingContactsModifiedBy' DESC 'Person who last modified the inetIpv6RoutingContacts attribute.' EQUALITY distinguishedNameMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE USAGE distributedOperation )

inetIpv6RoutingContactsModifiedDate
( 1.3.6.1.4.1.7161.1.3.11 NAME 'inetIpv6RoutingContactsModifiedDate' DESC 'Last modification date of the inetIpv6RoutingContacts attribute.' EQUALITY generalizedTimeMatch ORDERING generalizedTimeOrderingMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE USAGE distributedOperation )

The inetIpv6NetworkSyntax syntax is as follows:

inetIpv6NetworkSyntax
( 1.3.6.1.4.1.7161.1.3.1 NAME 'inetIpv6NetworkSyntax' DESC 'An IPv6 address and prefix.' )

3.3. Example

An example of the inetIpv6Network object class is shown in Figure 1 below, with attributes from the inetResources object class also being used to provide administrative contacts. This data is a
result of a query which was sent to the LDAP servers responsible for operating the ip6.arpa delegation domain.

```
cn=3ffe:ffff:0000:0000:0000:0000:0000:0000/32,
  cn/inetResources,dc=ip6,dc=arpa
[top object class]
/inetResources object class]
/inetIpv6Network object class]
|  +attribute: description
|   value: "The example.net top-level network"
|  +attribute: inetIpv6Contacts
|   value: "ldap://ldap.example.com/cn=hostmaster,ou=admins,
|          dc=example,dc=net"
|  +attribute: inetGeneralContacts
|   value: "ldap://ldap.example.com/cn=admins,ou=admins,
|          dc=example,dc=net"
```

Figure 1: The 3ffe:ffff:0000:0000:0000:0000:0000:0000/32 inetIpv6Network delegation entry.

Reverse-lookup DNS domains for IPv6 address blocks are managed as inetDnsDomain object class entries which are entirely different network resources, and which should not be confused with the inetIpv6Network object class entries. Note that due to the 128-bit address size and the structuring rules defined in RFC 1886, a fully-formed IPv6 reverse-lookup domain name will have 34 labels, which can result in very large distinguished names.

4. The inetIpv6NetworkMatch Filter

The inetIpv6NetworkMatch filter provides an identifier and search string format which collectively inform a queried server that a specific IPv6 address should be searched for, and that any matching inetIpv6network object class entries should be returned.

NOTE: IPv6 addresses are also stored in DNS for reverse-lookups, and those entries are treated as inetDnsDomain object class entries rather than being treated as inetIpv6Network object class entries (they are treated as DNS zones with their own operational administrators). As such, those entries use the inetDnsDomainMatch query described in [ldap-whois-dns].
The inetIpv6NetworkMatch extensibleMatch filter is defined as follows:

inetIpv6NetworkMatch
( 1.3.6.1.4.1.7161.1.4.19 NAME 'inetIpv6NetworkMatch' SYNTAX inetIpv6NetworkSyntax )

The assertion value MUST be an IPv6 address, using the inetIpv6NetworkSyntax defined in section 3. Clients MUST provide assertion values in this syntax. If an input string does not match this syntax, the client MAY manipulate the input string to form a valid assertion value. For example, if a user provides a zero-compressed IPv6 address such as 3ffe:ffff::, the client MAY convert the input value to the inetIpv6NetworkSyntax form of "3ffe:ffff:0000:0000:0000:0000:0000:0000/32".

The server MUST compare the assertion value against the RDN of all entries in the inetResources container which have an object class of inetIpv6Network. Any entry for an IPv6 network resource which is clearly superior to the IPv6 address provided in the input string MUST be returned to the client. Entries which do not encompass the queried address MUST NOT be returned. Entries which do not have an object class of inetIpv6Network MUST NOT be returned.

Using the notation format described in RFC 2254, the search filter expression for the inetDnsDomainMatch query above would be written as "(1.3.6.1.4.1.7161.1.4.19:=
3ffe:ffff:0000:0000:0000:0000:0000:0000/32)".

Response entries MAY be fully-developed inetIpv6Network entries, or MAY be referrals generated from entries which have the inetIpv6Network and referral object classes defined. Any attribute values which are received MUST be displayed by the client. If a subordinate reference referral is received, the client MUST restart the query, using the provided data as the new search base. If any continuation reference referrals are received, the client SHOULD start new queries for each reference, and append the output of those queries to the original query’s output.

5. Security Considerations
This document describes an application of the LDAPv3 protocol, and as such it inherits the security considerations associated with LDAPv3, as described in section 7 of RFC 2251.

By nature, LDAP is a read-write protocol, while the legacy WHOIS service has always been a read-only service. As such, there are significant risks associated with allowing unintended updates by unauthorized third-parties. Moreover, allowing the LDAP-WHOIS service to update the underlying delegation databases could result in network resources being stolen from their lawful operators. For example, if the LDAP front-end had update access to a domain delegation database, a malicious third-party could theoretically take ownership of that domain by exploiting an authentication weakness, thereby causing ownership of the domain to be changed to another party. For this reason, it is imperative that the LDAP-WHOIS service not be allowed to make critical modifications to delegated resources without ensuring that all possible precautions have been taken.

The query processing models described in this document make use of DNS lookups in order to locate the LDAP servers associated with a particular resource. DNS is susceptible to certain attacks and forgeries which may be used to redirect clients to LDAP servers which are not authoritative for the resource in question.

Some operators may choose to purposefully provide misleading or erroneous information in an effort to avoid responsibility for bad behavior. In addition, there are likely to be sporadic operator errors which will result in confusing or erroneous answers.

This document provides multiple query models which will cause the same query to be answered by different servers (one would be processed by a delegation entity, while another would be processed by an operational entity). As a result, each of the servers may provide different information, depending upon the query type that was originally selected.

For all of the reasons listed above, it is essential that applications and end-users not make critical decisions based on the information provided by the LDAP-WHOIS service without having reason to believe the veracity of the information. Users should limit unknown or untrusted information to routine purposes.

Finally, there are physical security issues associated with any service which provides physical addressing and delivery information. Although organizations are generally encouraged to...
provide as much information as they feel comfortable with, no
information is required.

6. IANA Considerations

This document defines an application of the LDAPv3 protocol rather
than a new Internet application protocol. As such, there are no
protocol-related IANA considerations.

However, this document does define several LDAP schema elements,
including object classes, attributes, syntaxes and extensibleMatch
filters, and these elements should be assigned OID values from the
IANA branch, rather than being assigned from a particular
enterprise branch.

Finally, this document also describes several instances where
public DNS and LDAP servers are queried. It is expected that IANA
will establish and maintain these LDAP servers (and the necessary
DNS SRV domain names and resource records) required for this
service to operate. This includes providing SRV resource records
in the generic TLDs and the root domain, and also includes
administering the referenced LDAP servers.

7. Author’s Addresses

Eric A. Hall
ehall@ehsco.com

8. References

RFC 2247 - Using Domains in LDAP/X.500 DNs
RFC 2251 - Lightweight Directory Access Protocol (v3)
Attribute Syntax Definitions.
RFC 2254 - The String Representation of LDAP Search Filters
[ir-dir-req] - <draft-newton-ir-dir-requirements-00.txt> -
Internet Registry Directory Requirements
9. Acknowledgments

Portions of this work were funded by Network Solutions, Inc.