INTERNET-DRAFT
Document: draft-ietf-crisp-lw-dns-00.txt
Expires: January, 2003
Category: Standards-Track

Eric A. Hall

July 2002

Defining and Locating DNS Domains using the Internet Resource Query Service

1. Abstract

This document defines LDAP schema and searching rules for DNS domains, 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 2251 - Lightweight Directory Access Protocol (v3)

RFC 2252 - 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

[ldap-whois] - <draft-ietf-crisp-lw-core-00.txt> - The Internet Resource Query Service and the Internet Resource Schema

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 inetDnsDomain Object Class

The inetDnsDomain object class is a structural object class which provides administrative information about a specific DNS domain name resource, such as a zone, a well-known host, or some other network resource which is primarily identified by a domain name.

3.1. Naming syntax

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

The inetDnsDomainSyntax component of an entry is subject to DN rules, although the inetDnsDomainSyntax is also used for extended search operations, and is therefore subject to specific syntax rules. The basic rules for this format are that domain names MUST be stored as sequences of labels, where each label consists of a maximum of 63 characters, with each label being separated by a full-stop (period) character, and with the entire domain name sequence being a maximum of 255 characters.

For example, the "www.example.com" DNS domain name resource stored in the "dc=example,dc=com" DIT would be represented as an entry named "cn=www.example.com,cn=inetResources,dc=example,dc=com", while the "2.0.192.in-addr.arpa" reverse-lookup domain which was stored in the "dc=example,dc=com" DIT would be named "cn=2.0.192.in-addr.arpa,cn=inetResources,dc=example,dc=com".

Note that the domain name syntax rules defined by STD 13 allow any eight-bit character code to be used within any domain name, although the host naming rules defined by RFC 952, STD 13 and STD

3 only allow a subset of the printable characters from US-ASCII to be used for domain names which specify connection targets. However, many domain names will need to be queried which will not conform to the host naming rules ("_ldap._tcp.example.com" might be specified in a search, for example), so any eight-bit character MUST be considered valid for this service.

RFC 2253 defines several escaping mechanisms for use when handling certain "special" characters, and these mechanisms MUST be used whenever a character in a domain name needs to be escaped in order for an assertion value to be parsed. However, STD 13 also allows the use of special characters, and also provides several mechanisms for escaping special characters in DNS domain names, and these rules MUST also be accommodated if valid DNS names are to be supported.

In order to facilitate this potential overlap while minimizing confusion during handling, LDAP-WHOIS clients MUST allow DNS domain name query strings to be entered as raw eight-bit data, but if any of the characters need to be escaped for the assertion value to be properly built, then the client MUST escape these characters before the search is submitted.

Secondarily, if the user needs to search for a DNS domain name which would normally require escaping or other special handling in order for the domain name to be processed, then the user MUST provide the domain name in its escaped form. By extension, this also means that these DNS domain names MUST be stored as RDNs in their escaped form.

STD 13 and RFC 2253 both use a common method of escaping special characters with a reverse solidus (backslash) character, with either the special character or a two-digit decimal code for that character immediately following the reverse solidus.

For example, if a user needs to specify the domain name of "weird name.example.com" (where "weird name" is a valid domain name label with an embedded space), then the domain name would have an RDN of "cn=weird\32name.example.com" in the directory, and would have to be entered into the client as a search for "weird\32name.example.com". The client would then perform a secondary escape to form "weird\\32name.example.com" as the assertion value, and this secondary escape would be removed by the LDAP-WHOIS server upon receipt. Thus a match would be found.

> NOTE: Remember that IPv4 addresses are also stored in DNS for reverse-lookup purposes, and the associated zones and PTR domain names may also require escaping, particularly when used with site-specific CIDR notation.

The common reference to the root domain is a single full-stop (".") character, and this usage is also endorsed by this document when the root domain name needs to be explicitly queried. For any domain name which contains a non-root label, the trailing period which normally signifies the root domain MUST be omitted. The maximum size of a valid DNS domain name is 255 characters (this limit applies to the unescaped assertion value). Clients MUST restrict input to this range, prior to submitting the LDAP query.

The domain name component of the DN MUST match the domain name of the managed resource exactly as the domain name exists in the DNS namespace. For example, if an organization wishes to provide information about "www.example.com", then a RDN entry would need to exist for "cn=www.example.com". If an organization wishes to provide information about the "www.example.com" canonical target "server1.example.net", then a RDN for "cn=server1.example.net" would need to exist. If an organization wishes to provide information about "server1.example.net" whenever a query is received for "www.example.com", then the "cn=www.example.com" entry would need to be defined as a subordinate reference referral, with the ref attribute pointing to the "cn=server1.example.net" entry.

This usage model also applies to reverse-lookup domains. If an organization is the authority for the "2.0.192.in-addr.arpa" reverse-lookup domain associated with an IPv4 network (this is different from providing information about the network block in particular, as is discussed separately in [ldap-whois-ipv4] and [ldap-whois-ipv6], then that syntax would also be used to form the RDN for the associated inetDnsDomain entry.

Note that reverse-lookup domain names are mapped directly as they exist in the public DNS namespace. If a /24 IPv4 network block such as 192.0.2.0 has been delegated to an organization, the default controlling domain name of the reverse-lookup zone will be 2.0.192.in-addr.arpa, and the name of the associated LDAP-WHOIS entry would be "cn=2.0.192.in-addr.arpa". However, if that network had been delegated to an ISP who had in turn delegated the 192.0.2.0/29 address block and an associated reverse-lookup zone of 29-0.2.0.192.in-addr.arpa to a user, then the associated LDAP-WHOIS entry for that zone would be "cn=29-0.2.0.192.in-addr.arpa".

3.2. Schema definition

DNS domain name entries MUST exist with the top, inetResources and inetDnsDomain 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 inetDnsDomain 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 inetDnsDomain object class has no mandatory attributes, although it does have several optional attributes.

The inetDnsDomain object class defines attributes which are specific to DNS domains, particularly as this relates to domain delegation (DNS operational information is available through DNS itself). This includes information such as the delegation date and the status of the delegation. The inetDnsDomain object class is subordinate to the inetResources object class, so it inherits those attributes as well.

Some of the inetDnsDomain 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 inetDnsDomain object class is as follows:

```
inetDnsDomain
( 1.3.6.1.4.1.7161.1.1.0 NAME 'inetDnsDomain' DESC 'DNS
  domain attributes.' SUP inetResources STRUCTURAL MAY (
  inetDnsDelegationStatus $ inetDnsDelegationDate $
  inetDnsDelegationModifiedDate $ inetDnsDelegationModifiedBy
  $ inetDnsContacts $ inetDnsContactsModifiedBy $
  inetDnsContactsModifiedDate $ inetDnsAuthServers ) )
```

The attributes from the inetDnsDomain object class are described below:

```
inetDnsAuthServers
( 1.3.6.1.4.1.7161.1.1.2 NAME 'inetDnsAuthServers' DESC
  'Authoritative DNS servers for this domain.' EQUALITY
  caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

The inetDnsAuthServers attribute provides a read-only summary of the authoritative servers associated with the zone. The attribute is defined as multi-valued, with each attribute value currently (tentatively) being defined as:

```
domain.dom [address/prefix]
```

where "domain.dom" is the domain name of the authoritative server, written as an inetDnsDomainSyntax string, and where "address/prefix" is an IPv4 or IPv6 host-specific network address, written as either an inetIpv4NetworkSyntax or inetIpv6NetworkSyntax string. Clients that wish to obtain additional information about the listed servers can issue new queries for either the domain name or address syntax.

NOTE: THIS IS A TEMPORARY ATTRIBUTE WHICH WILL EVENTUALLY BE REPLACED WITH GENERALIZED RESOURCE-RECORD ENTRIES AND ATTRIBUTES.

```
inetDnsContacts
( 1.3.6.1.4.1.7161.1.1.3 NAME 'inetDnsContacts' DESC
  'Contacts for reporting problems with this domain name.'
  EQUALITY caseExactMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.15 )
```

```
inetDnsContactsModifiedBy
( 1.3.6.1.4.1.7161.1.1.4 NAME 'inetDnsContactsModifiedBy'
  DESC 'Person who last modified the inetDnsContacts
  attribute.' EQUALITY distinguishedNameMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE USAGE
  distributedOperation )

inetDnsContactsModifiedDate
( 1.3.6.1.4.1.7161.1.1.5 NAME 'inetDnsContactsModifiedDate'
  DESC 'Last modification date of the inetDnsContacts
  attribute.' EQUALITY generalizedTimeMatch ORDERING
  generalizedTimeOrderingMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE USAGE
  distributedOperation )

inetDnsDelegationDate
( 1.3.6.1.4.1.7161.1.1.6 NAME 'inetDnsDelegationDate' DESC
  'Date of original delegation.' EQUALITY
  GeneralizedTimeMatch ORDERING generalizedTimeOrderingMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE )

inetDnsDelegationModifiedBy
( 1.3.6.1.4.1.7161.1.1.7 NAME 'inetDnsDelegationModifiedBy'
  DESC 'Person who last modified the  inetDnsDelegationStatus
  attribute.' EQUALITY distinguishedNameMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE USAGE
  distributedOperation )

inetDnsDelegationModifiedDate
( 1.3.6.1.4.1.7161.1.1.8 NAME 'inetDnsDelegationModifiedDate'
  DESC 'Last modification date of the inetDnsDelegationStatus
  attribute.' EQUALITY generalizedTimeMatch ORDERING
  generalizedTimeOrderingMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE USAGE
  distributedOperation )

inetDnsDelegationStatus
( 1.3.6.1.4.1.7161.1.1.9 NAME 'inetDnsDelegationStatus' DESC
  'Current delegation status code for this domain.' 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 inetDnsDelegationStatus 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.

The inetDnsDomainSyntax syntax is as follows:

```
inetDnsDomainSyntax
( 1.3.6.1.4.1.7161.1.1.1 NAME 'inetDnsDomainSyntax' DESC 'A
  fully-qualified DNS domain name.' )
```

3.3. Example

An example of the inetDnsDomain object class in use is shown in Figure 1 below, with some additional attributes inherited from the parent inetResources entry. This query is most likely being sent to the LDAP servers responsible for operating the "example.com" DNS domain.

```
cn=example.com,cn=inetResources,dc=example,dc=com
[top object class]
[inetResources object class]
[inetDnsDomain object class]
|
+-attribute: description
| value: "The example.com DNS domain"
|
+-attribute: inetDnsContacts
| value: "ldap://ldap.example.com/cn=hostmaster,ou=admins,
|           dc=example,dc=com"
|
+-attribute: inetGeneralContacts
  value: "ldap://ldap.example.com/cn=admins,ou=admins,
            dc=example,dc=com"
```

Figure 1: The example.com inetDnsDomain entry.

4. The inetDnsDomainMatch Filter

The inetDnsDomainMatch filter provides an identifier and search string format which collectively inform a queried server that a specific DNS domain name should be searched for, and that any matching inetDnsDomain object class entries should be returned.

The inetDnsDomainMatch extensibleMatch filter is defined as follows:

```
inetDnsDomainMatch
( 1.3.6.1.4.1.7161.1.1.9 NAME 'inetDnsDomainMatch' SYNTAX
  inetDnsDomainSyntax )
```

The assertion value MUST be a valid DNS domain name, using the inetDnsDomainSyntax syntax rules defined in section 3.

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

For example, assume that the client has issued a query with the assertion value of "www.example.com". If the queried server has an inetDnsDomain object class entry with a DN of "cn=example.com,cn=inetResources,dc=com", then that entry would be returned to the client. Similarly, a continuation reference referral of "cn=cref1,cn=example.com,cn=inetResources,dc=com" would also be returned, since it has a "cn" component that is superior to the queried domain name, and has the inetDnsDomain object class.

Domain names MUST be compared on label boundaries, and MUST NOT be qualified through simple character matching. Given two entries of "cn=example.com" and "cn=an-example.com", only the first would match an assertion value of "example.com".

Using the notation format described in RFC 2254, the search filter expression for the inetDnsDomainMatch query above would be written as "(1.3.6.1.4.1.7161.1.1.9:=www.example.com)".

Response entries MAY be fully-developed inetDnsDomain entries, or MAY be referrals generated from entries which have the inetDnsDomain 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)

RFC 2252 - 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

[ldap-whois] - <draft-ietf-crisp-lw-core-00.txt> - The Internet Resource Query Service and the Internet Resource Schema

[ldap-whois-ipv4] - <draft-ietf-crisp-lw-ipv4-00.txt> - Defining and Locating IPv4 Address Blocks using the Internet Resource Query Service

[ldap-whois-ipv6] - <draft-ietf-crisp-lw-ipv6-00.txt> - Defining and Locating IPv6 Address Blocks using the Internet Resource Query Service

9. Acknowledgments

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