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

Eric A. Hall

July 2002

The Internet Resource Query Service and the Internet Resource Schema

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC 2026.

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.

1. Abstract

This document describes an architectural framework for locating and retrieving information about network resources, using LDAPv3 for the data-formatting and query-processing services. This document also defines LDAP schema and searching rules for four Internet resource types: DNS domains, IPv4 addresses, IPv6 address, and AS numbers. The framework specified in this document also allows additional documents to define additional Internet resource types and their handling rules.

Table of Contents

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

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. Background, Objectives and Overview

3.1. Background

The WHOIS service was originally provided as a front-end to a centralized repository of ARPANET resources and users. Over time, multiple WHOIS information servers have been deployed which provide other kinds of information for various types of Internet resources.

For example, there are scores of WHOIS servers which serve one or more of the top-level domains ("com", "jp", etc.), with each server providing information about the sub-domains that have been delegated beneath each of the managed TLDs, and which also provide information about the human operators of those domains, among other details. Similarly, there are WHOIS servers which provide information about different portions of the IPv4 address space. Similarly, there are WHOIS servers which are operated by service providers which provide information about the resources in use by that organization and its customers. All told, there are hundreds of WHOIS servers available on the public Internet, with each server providing general information about the particular network resources under the control of each organization.

Unfortunately, the WHOIS specification does not define a strict set of data-typing or formatting requirements, and as a result, each of the different implementations provide information in slightly different ways. Some servers provide limited amounts of unstructured information, while others provide information in a highly-detailed and highly-structured form. Similarly, some

servers provide information in only one language and charset, while others support multiple languages and charsets, and use input switches to control the output format. Essentially, every WHOIS server has its own data formats and syntaxes, with little consistency between them, which has made programmatic processing of the data difficult.

Furthermore, each WHOIS server operates as a self-contained entity, with no knowledge or linkage between the different servers, meaning that WHOIS servers cannot redirect clients to other servers for additional information.

Another concern is that the WHOIS services which are being operated today offer no means of client authentication, requiring that server operators essentially publish all data with a single "world-readable" permission. However, this single permission conflicts with the privacy and security policies of specific jurisdictions. A more flexible mechanism for controlling the release of physical and personal information is required in order to meet the requirements of the varying constituencies.

There are many other secondary issues with the WHOIS service as it exists in current form. However, the largest problems are a lack of standardized data formats, a lack of widely-supported referral mechanisms, and lack of privacy and security controls, as described in the preceding text.

This document attempts to address these issues by defining operational and protocol guidelines for a distributed and highly-structured WHOIS-like service, using the LDAP protocol for the query/response transfer service, and using LDAP schema for the search inputs, answer data, and redirection mechanisms. In short, the intention of this approach is to provide an extensible and scalable WHOIS service, leveraging the capabilities of LDAP.

3.2. Overview

This document defines four basic service components and their interaction as part of a distributed resource-locator service. Each of these components work together to provide a structured and distributed resource-locator service.

Specifically, this document only defines the elements which govern the core service. Separate documents define the individual resource types, their schema and matching filters, and so forth.

As such, the architecture and protocols defined in this specification are purposefully designed to be capable of accommodating a variety of different data-types and usage models, including future uses which are not defined here.

The four components of the core service model are:

* Structured Namespace. This document makes use of an LDAP namespace which is built upon the existing DNS delegation hierarchy, and which is supplemented by a layered namespace consisting of service-specific containers and resource-specific entries. This namespace and the associated naming rules facilitate the programmatic formation of queries, structured data, and referrals.

* Schema Definitions. This document reuses many existing LDAP schema definitions, but also introduces several new object classes, attributes, syntaxes and matching filters. Some of these definitions apply to the overall architecture, while others are concerned with specific resource types.

* Searching Rules. This document defines several rules for forming queries which are designed to facilitate consistent answer sets, and to improve interoperability between compliant clients and servers.

* Query Processing Models. This document defines three distinct query-processing models which may be used for locating the authoritative servers associated with a named resource. These include a "top-down" model which is designed for querying centrally-managed Internet resources, a "bottom-up" model which is designed for querying user-managed resources, and a "targeted search" model which is designed for querying known servers for information about known resources. This document also specifies protocol behavior for following subordinate reference referrals, continuation reference referrals, and attribute references.

As stated above, this document defines core schema and matching rules, while external (related) documents define schema and matching rules for specific resource types. Among the resource types already defined (and partially reused herein) are:

* [ldap-whois-dns] - <draft-ietf-crisp-lw-dns-00.txt> - Defining and Locating DNS Domains using the Internet Resource Query Service

* [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

* [ldap-whois-asn] - <draft-ietf-crisp-lw-asn-00.txt> - Defining and Locating Autonomous System Numbers using the Internet Resource Query Service

* [ldap-whois-user] - <draft-ietf-crisp-lw-user-00.txt> - Defining and Locating Contact Persons using the Internet Resource Query Service

It is the intention of the author that additional resource types will be added to this framework over time.

4. The LDAP-WHOIS Namespace

A critical aspect of this service is the use of a predictable naming syntax, both for the automatic creation of programmatic searches for data, and for publishing structured data and referrals. In order to ensure this predictability, this document defines a multi-layered syntax which MUST be used by all compliant implementations.

The LDAP-WHOIS service also makes provisions for the use of multiple referral services for the purpose of redirecting LDAP clients to foreign directory information trees (DITs). This allows organizations to redirect queries to external service providers, consolidate DITs within a single server, maintain foreign objects within a private DIT (such as allowing a third-party router to exist as a separately managed resource within an end-user DIT), and allows answer sets to contain responses from multiple servers.

4.1. Namespace Example

Figure 1 below shows a subset example of the LDAP-WHOIS namespace. This namespace will be used throughout this document to illustrate many of the concepts from this specification.

```
DIT: dc=example,dc=com
|
+-cn=inetResources,dc=example,dc=com
  [top object class]
  [inetResources object class]
  |
  +-attribute: o
  | value: "Example Widgets, Inc. public network resources"
  |
  +-cn=example.com,cn=inetResources,dc=example,dc=com
  | [top object class]
  | [inetResources object class]
  | [inetDnsDomain object class]
  | |
  | +-attribute: inetDnsContacts
  |   value: "ldap://ldap.example.com/cn=hostmaster,
  |             ou=admins,dc=example,dc=com"
  |
  +-cn=2.0.192.in-addr.arpa,
      cn=inetResources,dc=example,dc=com
  | [top object class]
  | [inetResources object class]
  | [inetDnsDomain object class]
  | |
  | +-attribute: description
  | | value: "Example Widgets' reverse-lookup domain"
  | |
```

```
          | +-cn=cref1,cn=2.0.192.in-addr.arpa,
          |        cn=inetResources,dc=example,dc=com
          |   [top object class]
          |   [inetResources object class]
          |   [inetDnsDomain object class]
          |   [referral object class]
          |   |
          |   +-attribute: ref
          |     value: "ldap://ldap.example.com/cn=example.com,
          |               cn=inetResources,dc=example,dc=com"
          |
          +-cn=192.0.2.0/24,cn=inetResources,dc=example,dc=com
            [inetResources object class]
            [inetIpv4Network object class]
            |
            +-attribute: inetIpv4Contacts
              value: "ldap://ldap.example.com/cn=hostmaster,
                        ou=admins,dc=example,dc=com"

      DIT: dc=2,dc=0,dc=192,dc=in-addr,dc=arpa
      |
      +-cn=inetResources
        [top object class]
        [inetResources object class]
        [referral object class]
        |
        +-attribute: ref
          value: "ldap://ldap.example.com/cn=inetResources,
                    dc=example,dc=com"
```

Figure 1: Namespace for Example Widgets' domain and network.

Figure 1 shows different DITs, both of which are managed by the Example Widgets company. The "dc=example,dc=com" DIT is authoritative for the DNS domain of "example.com", while the "dc=2,dc=0,dc=192,dc=in-addr,dc=arpa" DIT is authoritative for the reverse-lookup DNS domain of 2.0.192.in-addr.arpa and the IPv4 network of "192.0.2.0/24".

Both DITs have container entries called "cn=inetResources". This container entry is responsible for holding all of the entries which are associated with the Internet resources that are being managed by the LDAP-WHOIS service. For example, the "cn=inetResources,dc=example,dc=com" entry contains a subordinate entry for "cn=example.com", which is a DNS domain that is being managed through the LDAP-WHOIS service, and also contains entries

for the 2.0.192.in-addr.arpa reverse-lookup DNS domain and the 192.0.2.0/24 IPv4 network.

The "cn=inetResources,dc=2,dc=0,dc=192,dc=in-addr,dc=arpa" entry only exists as a referral which will cause queries to be redirected to the "cn=inetResources,dc=example,dc=com" hierarchy.

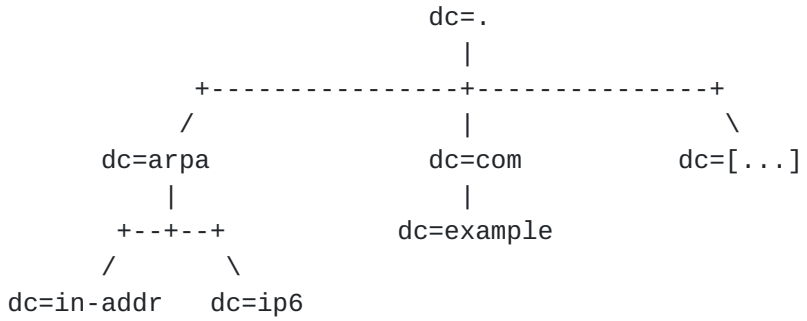
The naming syntax and rules are described throughout the remainder of this section 4.1. Figure 1 is only provided as an example a relatively complete namespace, for illustration and subsequent discussion purposes.

4.2. The domainComponent LDAP Hierarchy

The top-level of the namespace defined for use with this service uses the domainComponent naming syntax specified in RFC 2247, which maps DNS domain names to domainComponent ("dc=") labels to form a DIT. Each DIT represents a primary identifier for the management body that is offering an LDAP server, and as such, provides a primary identifier for the Internet resources under the control of that organization. The DITs will be used to build LDAP queries for specific resources, and will also be used to locate the LDAP servers associated with the controlling organization.

Examples of the RFC 2247 syntax are shown in Figure 2 below.

Figure 2: The LDAP-WHOIS domainComponent Namespace.



A complete sequence of domainComponent DNs represents the scope of the DIT. For example, a DIT with the distinguished name (DN) of "dc=com" is authoritative for all of the LDAP resources within the "com" DNS domain (for many LDAP-WHOIS queries, this will also include any sub-domains under the "com" domain). Meanwhile, a DIT with the DN of "dc=2,dc=0,dc=192,dc=in-addr,dc=arpa" DIT is authoritative for domain name resources within the reverse-lookup

"2.0.192.in-addr.arpa" DNS domain, as well as the IPv4 network addresses within the 192.0.2.0/24 network. At the other extreme, the dc="." DIT is responsible for all Internet resources (although this DIT is rarely used).

Since the DIT determines the scope of control over a set of resources, DITs that overlap also have overlapping scopes of control. For example, the "dc=com" and "dc=example,dc=com" DITs can both provide information about the "www.example.com" domain name resource. In order to allow end-users to specify which scope they wish to work with for any given query, this document defines three different query models (these are described in section 7).

When the LDAP servers associated with the chosen DIT need to be located, the domainComponent DNs from the DIT are mapped to a DNS domain name, and a query is issued for the LDAP servers associated with that domain name (this process is also described in section 7). This means that the authority to process LDAP searches for a DIT comes directly from the portion of the DNS namespace already under the control of that management body. For example, the LDAP servers which are used to process queries for the "dc=com" DIT will be located by querying the DNS zone responsible for the "com" portion of the DNS namespace, and so forth.

4.3. The inetResources Container

This specification requires the use of a mandatory LDAP container entry with the well-known relative distinguished name (RDN) of "cn=inetResources", which MUST exist in the root of every DIT that provides LDAP-WHOIS services. All resource-specific entries which are provided on public LDAP-WHOIS servers MUST be stored in the cn=inetResources container entry.

The primary motivation for this naming is for predictability, in that it allows searches to be formed programmatically (a search base for resources in the "dc=example,dc=com" DIT can be programmatically formed as "cn=inetResources,dc=example,dc=com", for example). However, there are several other motivating factors for this naming syntax.

For example, it is easier to apply a single anonymous read-only permission to the inetResources container than it is to apply the same permission to multiple discrete entries, which in turn means that it is more likely that the appropriate restrictions will be defined. Furthermore, the use of a single container entry for all

of an organization's Internet resources allows that branch of the DIT to be redirected to another DIT through the use of a single referral operation (this will be particularly important when the LDAP servers that are located by DNS lookups are not the same servers that provide LDAP-WHOIS services). Another reason to use this naming syntax is that it shelters clients from server-side vagaries with DIT entries (where different vendors use different object classes to define the DITs).

All told, the use of the "cn=inetResources" RDN facilitates smooth operations, and is important enough to justify the MANDATORY usage of this naming syntax.

4.4. Resource-Specific Entries

This document defines four Internet resource types, each of which have their own naming rules. However, each resource type has a consistent naming principle, in that the specific managed resource has an RDN which uniquely identifies that resource, with the RDN residing within the inetResources container entry.

For example, an entry for the "www.example.com" domain name resource stored in the "dc=example,dc=com" DIT would have a DN of "cn=www.example.com,cn=inetResources,dc=example,dc=com", while an entry for the "192.0.2.0/24" IPv4 network resource would have a DN of "cn=192.0.2.0/24,cn=inetResources,dc=example,dc=com". Although the relative naming syntax is different for each resource type, the resource naming is consistent for each type, and the position of the RDN within the DN is also predictable.

Most resource types cannot be located through simple LDAP browsing and equality matches. Instead, resource-specific entries use structured naming rules in order to facilitate the extensible match search operations which are specific to each of the defined resource types. For example, there is not likely to be a specific entry for every possible IPv4 address. In order to allow the appropriate entry to be located, however, the client can use the inetIpv4NetworkMatch extensible matching search operation, which locates the appropriate entry based on the search input.

The naming rules associated with each resource type are provided in section 5, along with the schema definitions for each of the resource types. The extensible matching filters associated with each resource type are described in section 6.

4.5. Redirects and Referrals

A critical objective behind this service is for servers to be able to redirect clients to other servers, entries, or DITs, when this is necessary or desirable. Towards this end, this document specifies three methods for generating and processing redirects and referrals: subordinate reference referrals, continuation reference referrals, and attribute references.

Subordinate reference referrals indicate that the queried entry is an alias for some other entry, and that the query has to be restarted in order for the current operation to be completed. Meanwhile, continuation references indicate that the search was successfully initiated, but that additional queries are required for the original query to be completely exhausted. Finally, attribute references simply indicate that supplemental data is available at some other location, but that no additional queries are required to satisfy the current operation.

> NOTE: RFC 2251 defines a superior reference referral which is used as a "default referral" for out-of-scope searches. However, this application specifically excludes support for superior reference referrals. Any superior reference referrals which are encountered as a part of this service are to be treated as errors.

Subordinate references and continuation references use the ref attribute and referral object class defined in [namedref]. Attribute references use a superset of the formatting rules associated with the labeledURI attribute, as defined in RFC 2079. All of these mechanisms use LDAP URLs as their input data, although these URLs have service-specific restrictions that are somewhat tighter than the source specifications allow.

Among these rules:

* All referenced entries MUST comply with the naming syntax rules specified in this document. This means that all entries MUST use the domainComponent ("dc=") naming syntax for their DITs, resource-specific entries MUST reside in the inetResources container entry, and resource-specific entries MUST comply with the naming rules for the resource type in question.

* Referral sources and targets MUST have the same resource-specific object classes defined (for example, the referral source and target for a DNS domain resource would both have the inetDnsDomain object class defined). This is a prerequisite for the proper handling of the search filters specified in this document. Attribute references are not referrals, so they are exempt from this requirement.

* Referenced entries MAY exist as referrals to other entries, but recursive referrals are discouraged.

* Except where otherwise noted, the protocol identifier of a URL MUST specify the LDAP service type. Although general-purpose LDAP referrals are allowed to specify any URL, LDAP-WHOIS referrals and references are intended to be used for automated queries, so the use of other protocols or services is expressly forbidden.

* The host identifier of a URL MUST specify either an IP address or a domain name. URLs which do not provide host identifiers are invalid in all cases.

* URLs MUST be provided and stored in a URL-safe format. For example, the IPv4 and IPv6 network address syntaxes defined in this document make use of the forward-slash ("/") character to indicate a subnet prefix, and if this character needs to be provided in a URL, it must be provided in the escaped form ("%2F" in this example). Furthermore, some of the matching rules described in this document require that the URLs also be stored in this format in order for the searches to succeed.

* Implementations MUST NOT restrict URL values to verifiable entries from local partitions. Implementations MAY validate targets when the partition is known and accessible, but a lack of knowledge regarding a target MUST NOT be cause to prevent the entry from being specified.

Clients MAY implement support for additional protocol identifiers if they wish to act upon URLs which are provided in conflict with the requirements above. However, clients MUST NOT violate any other mandates in this document while doing so (in particular, clients MUST NOT break the query-processing procedures defined in section 7 of this document).

Each of the supported redirection mechanisms are discussed in more detail below.

4.5.1. Subordinate reference referrals

Subordinate reference referrals are returned when the search base specified in a query names an entry which exists as a referral object class that points to some other entry.

Any of the named entries specified in section 4 of this document MAY be defined as subordinate reference referrals which point to other entries. However, almost all of the search functions defined for use by this service use the inetResources container entry as the search base (the exceptions to this rule are targeted searches for explicit entries), so subordinate reference referrals will most commonly be seen when an inetResources container entry has been redirected to an inetResources container in another DIT.

For example, the namespace shown in Figure 1 has an entry of "cn=inetResources,dc=2,dc=0,dc=192,dc=in-addr,dc=arpa" defined with the referral object class, with the ref attribute value pointing to the LDAP server of "ldap.example.com" and the DN of "cn=inetResources,dc=example,dc=com". Any queries for resources within "cn=inetResources,dc=2,dc=0,dc=192,dc=in-addr,dc=arpa" would be answered with that subordinate reference referral, and these queries would have to be restarted using the specified search base and server before they would be processed.

Servers MUST support the use of subordinate reference referrals for this purpose, and clients MUST be prepared to accept and process any subordinate reference referrals in answer sets.

When subordinate reference referrals are used for this purpose, the referral source MUST be defined with the referral object class, and MUST also be defined with the appropriate object class for that resource type. For example, a "cn=inetResources" entry which provided a subordinate reference referral would need to have both the referral and inetResources object classes defined, while a DNS domain resource such as "dc=example.com" would need to have both the referral and inetDnsDomain object classes defined (among the other object class definitions which were required for that entry). Referral targets need to use whatever object classes are appropriate for the resource in question, and MAY also be referrals to other entries.

Client rules for processing subordinate reference referrals are given in section 7.4.1.

4.5.2. Continuation reference referrals

Continuation reference referrals are returned when a search operation has been successfully processed by the queried server, but the answer data also includes referrals to other entries. These referrals are often provided as supplemental data to an answer set, although this is not required (a continuation reference referral can be the only response, but it won't be the only response in the common case).

For example, the namespace shown in Figure 1 has an entry of "cn=cref1,cn=2.0.192.in-addr.arpa,cn=inetResources,dc=example, dc=com" defined with the referral object class, with the ref attribute value pointing to the LDAP server of "ldap.example.com" and the DN of "cn=example.com,cn=inetResources,dc=example,dc=com". Any answers to any queries about "cn=2.0.192.in-addr.arpa" would also include the continuation reference referral, and new queries for the referral target would have to be issued before the original queries were completely processed.

Servers MUST support the use of continuation reference referrals for this purpose, and clients MUST be prepared to accept and process any subordinate reference referrals in answer sets.

When continuation reference referrals are used for this purpose, entries MAY exist for the queried resource, but one or more entries MUST exist with the referral object class defined, and which provide LDAP URLs that point to other entries which have additional information about the resource in question.

Continuation reference referrals are returned in response to specific extensible match queries, and have specific naming requirements which are necessary for the matching functions to work properly. These considerations are described in 7.4.3.

Client rules for processing continuation reference referrals are also given in section 7.4.3.

4.5.3. Attribute references

This document defines attribute references as attribute values which provide LDAP URLs, for the purpose of providing pointers to contextually-related information regarding the entry currently being viewed. Attribute references use the same basic syntax as the labeledURI attribute defined in RFC 2079, although there are additional restrictions, as described above.

The contact attributes defined in this document use the attribute reference rules and formats to provide role-specific pointers to inetOrgPerson entries. Whenever one of these attributes is encountered, it is up to the client to deconstruct the provided URLs in order to locate and read the inetOrgPerson entries, although such actions are secondary to the original query process, and will typically only be performed at the user's request.

For example, the namespace shown in Figure 1 has an entry of "cn=192.0.2.0/24,cn=inetResources,dc=example,dc=com" defined with the inetIpv4Network object class, with the inetIpv4Contacts attribute value pointing to the LDAP server of "ldap.example.com" and the DN of "cn=hostmaster,ou=admins,dc=example,dc=com". When this attribute is provided as part of an answer to a query, a client MAY choose to follow this URL for information about that contact, although this would be considered a new and separate query, and would not be required to satisfy the original query.

Note that the attribute reference URLs are similar to the URLs defined in RFC 2079, and use a two-part notation of "url://any.host:port/any/path description", with the "description" string providing a free-text description of the target specified by the URL. When the descriptive text is provided in an attribute reference, it SHOULD be displayed to the user as potentially informative data.

Client rules for processing attribute references are given in section 7.4.4.

4.5.4. labeledURI references

Some of the object classes defined in this document make use of the labeledURI attribute, as defined by RFC 2079. These attributes (and their values) are not governed by this document, but instead are governed by RFC 2079. As such, the rules set forth in RFC 2079

always apply to those attributes. In particular, this means that those attribute values may reference any protocol (such as http://) and are not restricted to LDAP protocol targets.

5. The LDAP-WHOIS Object Classes and Attributes

This document defines a general framework for locating information about public network resources in a distributed environment, and a critical element of this definition are schema definitions for the object classes and attributes that provide this information.

Towards that end, this document defines a schema for the global inetResources object class which is inherited by all of the resource-specific types, defines four resource-specific object classes, and defines a generalized object class for cross-referencing resources. This document also takes advantage of some pre-existing schema definitions (in particular, the inetOrgPerson object class), where suitable schema were available. Each of the schema definitions provided in this document include attribute definitions, naming rules, and other definitions which are designed to facilitate searching and browsing Internet resources.

The following resource definitions are provided in this section:

* Organizational and summary data. The inetResources object class defines a variety of general-purpose attributes for describing an organization and its resources. For example, there is a free-text attribute which may be used to provide general comments about the organization or the resources under its control, attributes for providing general-use postal and email addresses, and so forth. The inetResources object class also defines several attributes which may be used to provide attribute references for a variety of administrative roles.

* Associated objects. Internet resources are typically assigned by independent entities, although there is often an extensive amount of cross-pollination. For example, AS Numbers are typically associated with IPv4 and IPv6 address blocks, with this association being considered as a mandatory linkage. However, less-formal associations also often exist, such as a private organization associating an IP address block with a specific DNS domain, or where a regional authority is responsible for all domain name and IP address delegations. Due to this flexibility, the LDAP-

WHOIS service provides an auxiliary object class for associated objects which may be used with any of the resource-specific object classes defined in this document.

Each of the attributes and object classes listed above represent the Internet-wide network resources which MAY be offered by an LDAP-WHOIS server.

It is expected that additional network resource definitions will be provided by other documents.

5.1. The inetResources Object Class

The inetResources object class is a structural object class which defines the attributes associated with a "cn=inetResources" container entry, and which provides general information about the network resources associated with the current DIT.

5.1.1. Naming syntax

This document requires the presence of an entry named "cn=inetResources" in the root of every DIT which provides LDAP-WHOIS services.

5.1.2. Schema definition

Every DIT which provides public LDAP-WHOIS data MUST have a "cn=inetResources" entry in the root of the DIT. The inetResources entry MUST exist with the top and inetResources object classes defined. If the entry exists as a referral, the entry MUST also be defined with the referral object class, in addition to the above requirements.

The inetResources object class is a structural object class which is subordinate to the top abstract class, and which MUST be treated as a container class capable of holding additional subordinate entries. The inetResources object class has one mandatory attribute which is "cn" (the naming attribute), and also has several optional attributes. Each of the other object classes defined by this document are subordinate to the inetResources object class and inherit the attributes defined for the inetResources object class.

The inetResources object class is intended to provide summary information about a collection of resources under the control of a single organization or management body. For example, the mail attribute is intended to be used as a general-purpose email address for the organization as a whole (such as "info@example.com"), rather than being used as an email address for a particular administrative role. Because this object class is also inherited by the resource-specific object classes, these attributes can be defined at each of the subordinate entries if a global set of values is undesirable or unfeasible.

The inetResources object class provides several multi-valued contact-related attributes for a variety of well-known administrative roles. This model allows the inetResources entry and each of the subordinate managed resources to share a common set of administrative roles, or to have unique roles for each resource, as seen fit by the managing entity. The contact attribute values follow the same rules as the labeledURI attribute defined in RFC 2079, with additional restrictions as described in section 4.5 of this document.

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.

Since multiple directory trees can share a single inetResources entry (through the use of subordinate reference referrals), it is important for the associated data to be applicable for all of the objects which refer to it. For example, it would be effective for a small private company to use a shared set of inetResources attributes for their DNS domain names and IP network blocks, but it would probably be counter-productive for a global ISP to share contact data across all of their hosted domains and routed networks. If separate contacts are required for each resource, the contact data should be specified within each entry, rather than being linked to the inetResources entry.

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

```
inetResources
( 1.3.6.1.4.1.7161.1.0.0 NAME 'inetResources' DESC 'The
  inetResources container for the LDAP-WHOIS service' SUP top
  STRUCTURAL MUST cn MAY ( o $ ou $ description $
  inetResourceComments $ businessCategory $ telephoneNumber $
  facsimileTelephoneNumber $ mail $ labeledURI $
  preferredDeliveryMethod $ physicalDeliveryOfficeName $
  postOfficeBox $ postalAddress $ postalCode $ street $ l $
  st $ c $ inetAbuseContacts $ inetAbuseContactsModifiedBy $
  inetAbuseContactsModifiedDate $ inetGeneralContacts $
  inetGeneralContactsModifiedBy $
  inetGeneralContactsModifiedDate $ inetSecurityContacts $
  inetSecurityContactsModifiedBy $
  inetSecurityContactsModifiedDate $ inetTechContacts $
  inetTechContactsModifiedBy $ inetTechContactsModifiedDate $
  inetGeneralDisclaimer ) )
```

The attributes from the inetResources object class are described below:

businessCategory, see RFC 2256, section 5.16

c (country), see RFC 2256, section 5.7

cn (commonName), see RFC 2256, section 5.4

description, see RFC 2256, section 5.14

facsimileTelephoneNumber, see RFC 2256, section 5.24

```
inetAbuseContacts
( 1.3.6.1.4.1.7161.1.0.1 NAME 'inetAbuseContacts' DESC
  'Contacts for reporting abusive behavior or acceptable-use
  policy violations.' EQUALITY caseExactMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.15 )

inetAbuseContactsModifiedBy
( 1.3.6.1.4.1.7161.1.0.2 NAME 'inetAbuseContactsModifiedBy'
  DESC 'Person who last modified the inetAbuseContacts
  attribute.' EQUALITY distinguishedNameMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE USAGE
  distributedOperation )
```

```
inetAbuseContactsModifiedDate
( 1.3.6.1.4.1.7161.1.0.3 NAME 'inetAbuseContactsModifiedDate'
  DESC 'Last modification date of the inetAbuseContacts
  attribute.' EQUALITY generalizedTimeMatch ORDERING
  generalizedTimeOrderingMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE USAGE
  distributedOperation )

inetGeneralContacts
( 1.3.6.1.4.1.7161.1.0.4 NAME 'inetGeneralContacts' DESC
  'Contacts for general administrative issues.' EQUALITY
  caseExactMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

inetGeneralContactsModifiedBy
( 1.3.6.1.4.1.7161.1.0.5 NAME 'inetGeneralContactsModifiedBy'
  DESC 'Person who last modified the inetGeneralContacts
  attribute.' EQUALITY distinguishedNameMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE USAGE
  distributedOperation )

inetGeneralContactsModifiedDate
( 1.3.6.1.4.1.7161.1.0.6 NAME
  'inetGeneralContactsModifiedDate' DESC 'Last modification
  date of the inetGeneralContacts attribute.' EQUALITY
  generalizedTimeMatch ORDERING generalizedTimeOrderingMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE USAGE
  distributedOperation )

inetGeneralDisclaimer
( 1.3.6.1.4.1.7161.1.0.7 NAME 'inetResourceComments' DESC
  'General disclaimer text regarding this data' EQUALITY
  caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{1024}
  )

inetResourceComments
( 1.3.6.1.4.1.7161.1.0.8 NAME 'inetResourceComments' DESC
  'General comments about this entry' EQUALITY
  caseIgnoreMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{1024}
  )

inetSecurityContacts
( 1.3.6.1.4.1.7161.1.0.9 NAME 'inetSecurityContacts' DESC
  'Contacts for general security issues.' EQUALITY
  caseExactMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

```
inetSecurityContactsModifiedBy
( 1.3.6.1.4.1.7161.1.0.10 NAME
  'inetSecurityContactsModifiedBy' DESC 'Person who last
  modified the inetSecurityContacts attribute.' EQUALITY
  distinguishedNameMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  SINGLE-VALUE USAGE distributedOperation )

inetSecurityContactsModifiedDate
( 1.3.6.1.4.1.7161.1.0.11 NAME
  'inetSecurityContactsModifiedDate' DESC 'Last modification
  date of the inetSecurityContacts attribute.' EQUALITY
  generalizedTimeMatch ORDERING generalizedTimeOrderingMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE USAGE
  distributedOperation )

inetTechContacts
( 1.3.6.1.4.1.7161.1.0.12 NAME 'inetTechContacts' DESC
  'Contacts for general technical issues.' EQUALITY
  caseExactMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

inetTechContactsModifiedBy
( 1.3.6.1.4.1.7161.1.0.13 NAME 'inetTechContactsModifiedBy'
  DESC 'Person who last modified the inetTechContacts
  attribute.' EQUALITY distinguishedNameMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE USAGE
  distributedOperation )

inetTechContactsModifiedDate
( 1.3.6.1.4.1.7161.1.0.14 NAME 'inetTechContactsModifiedDate'
  DESC 'Last modification date of the inetTechContacts
  attribute.' EQUALITY generalizedTimeMatch ORDERING
  generalizedTimeOrderingMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE USAGE
  distributedOperation )
```

l (locality), see RFC 2256, section 5.8

labeledURI, see RFC 2079

mail, see RFC 2798, section 9.1.3

o (organization), see RFC 2256, section 5.11

ou (organizational unit), see RFC 2256, section 5.12

physicalDeliveryOfficeName, see RFC 2256, section 5.20

postalAddress, see RFC 2256, section 5.17

postalCode, see RFC 2256, section 5.18

postOfficeBox, see RFC 2256, section 5.19

preferredDeliveryMethod, see RFC 2256, section 5.29

st (stateOrProvinceName), see RFC 2256, section 5.9

street (streetAddress), see RFC 2256, section 5.10

telephoneNumber, see RFC 2256, section 5.21

5.1.3. Example

An example of the inetResources object class in use is shown in Figure 3 below.

```
cn=inetResources,dc=example,dc=com
[top object class]
[inetResources object class]
|
+-attribute: o
| value: "Example Widgets' network resources"
|
+-attribute: inetGeneralContacts
| value: "ldap://ldap.example.com/cn=admins,ou=admins,
|           dc=example,dc=com"
|
+-attribute: telephoneNumber
| value: "1-800-555-1212"
|
+-attribute: mail
| value: "info@example.com"
|
+-attribute: inetResourceComments
  value: "Please don't send complaints to the
          postmaster@example.com mailbox."
```

Figure 3: The Example Widgets inetResources container entry.

5.2. The inetAssociatedResources Object Class

The inetAssociatedResources object class defines cross-reference attributes which may be used with any of the object classes defined in this document. For example, it allows inetDnsDomain object class entries to be associated with IPv4 networks, or even to other DNS domains, if that information is known (this information may be useful if a single organization has multiple DNS domains registered). Furthermore, it allows inetOrgPerson object classes to be associated with managed resources such as IP networks or DNS domains. In short, any resource can be associated with any other resource through the use of this object class.

The inetAssociatedResources object class MUST NOT be associated with an entry that only exists as a referral source.

5.2.1. Naming syntax

The inetAssociatedResources object class is an auxiliary object class, and not a structural object class. Entries which use this object class definition are primarily defined under the rules associated with the structural object class that defines the Internet resource in question. As such, the naming rules associated with the structural object class in use with that entry take precedence. Therefore, the inetAssociatedResources object class does not define a naming syntax.

5.2.2. Schema definition

The inetAssociatedResources object class is an auxiliary object class which is subordinate to the top object class. The inetAssociatedResources object class has no mandatory attributes, although it does have several optional attributes.

Although the inetAssociatedResources object class is subordinate to the top object class, it is intended to only be associated with the resource-specific structural object classes defined in this document. For example, the inetAssociatedResources object class is not likely to provide much value when it is associated with the inetResources object class, since the inetResources object class does not specifically define any resources (and since it does not define resources, it cannot define any associated resources). On the other hand, it is reasonable for the inetAssociatedResources object class to be associated with an inetOrgPerson object class

entry, particularly if the referenced person (or role) is responsible for the management of multiple resources.

Each of the associated resource attributes provide multi-valued data, using the syntax notations which are specific to the resource in question. For example, the inetAssociatedDnsDomain attribute provides associated DNS domain name resources using a multi-valued array, with each DNS domain name using the inetDnsDomainSyntax naming rules.

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 inetAssociatedResources object class is as follows:

```
inetAssociatedResources
( 1.3.6.1.4.1.7161.1.5.0 NAME 'inetAssociatedResources' DESC
  'Network resources associated with this entry.' SUP top
  AUXILIARY MAY ( inetAssociatedDnsDomains $
  inetAssociatedDnsDomainsModifiedBy $
  inetAssociatedDnsDomainsModifiedDate $
  inetAssociatedIpv4Networks $
  inetAssociatedIpv4NetworksModifiedBy $
  inetAssociatedIpv4NetworksModifiedDate $
  inetAssociatedIpv6Networks $
  inetAssociatedIpv6NetworksModifiedBy $
  inetAssociatedIpv6NetworksModifiedDate $
  inetAssociatedAsNumbers $
  inetAssociatedAsNumbersModifiedBy $
  inetAssociatedAsNumbersModifiedDate ) )
```

The attributes from the inetAssociatedResources object class are described below:

```
inetAssociatedAsNumbers
( 1.3.6.1.4.1.7161.1.5.2 NAME 'inetAssociatedAsNumbers' DESC
  'The autonomous system numbers associated with this entry.'
  EQUALITY caseIgnoreMatch SYNTAX inetAsNumberSyntax )
```

```
inetAssociatedAsNumbersModifiedBy
( 1.3.6.1.4.1.7161.1.5.3 NAME
  'inetAssociatedAsNumbersModifiedBy' DESC 'Person who last
  modified the inetAssociatedAsNumbers attribute.' EQUALITY
  distinguishedNameMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  SINGLE-VALUE USAGE distributedOperation )

inetAssociatedAsNumbersModifiedDate
( 1.3.6.1.4.1.7161.1.5.4 NAME
  'inetAssociatedAsNumbersModifiedBy' DESC 'Last modification
  date of the inetAssociatedAsNumbers attribute.' EQUALITY
  generalizedTimeMatch ORDERING generalizedTimeOrderingMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE USAGE
  distributedOperation )

inetAssociatedDnsDomains
( 1.3.6.1.4.1.7161.1.5.5 NAME 'inetAssociatedDnsDomains' DESC
  'The DNS domains associated with this entry.' EQUALITY
  caseIgnoreMatch SYNTAX inetDnsDomainSyntax )

inetAssociatedDnsDomainsModifiedBy
( 1.3.6.1.4.1.7161.1.5.6 NAME
  'inetAssociatedDnsDomainsModifiedBy' DESC 'Person who last
  modified the inetAssociatedDnsDomains attribute.' EQUALITY
  distinguishedNameMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  SINGLE-VALUE USAGE distributedOperation )

inetAssociatedDnsDomainsModifiedDate
( 1.3.6.1.4.1.7161.1.5.7 NAME
  'inetAssociatedDnsDomainsModifiedBy' DESC 'Last
  modification date of the inetAssociatedDnsDomains
  attribute.' EQUALITY generalizedTimeMatch ORDERING
  generalizedTimeOrderingMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE USAGE
  distributedOperation )

inetAssociatedIpv4Networks
( 1.3.6.1.4.1.7161.1.5.8 NAME 'inetAssociatedIpv4Networks'
  DESC 'The IPv4 networks associated with this entry.'
  EQUALITY caseIgnoreMatch SYNTAX inetIpv4NetworkSyntax )
```

```
inetAssociatedIpv4NetworksModifiedBy
( 1.3.6.1.4.1.7161.1.5.9 NAME
  'inetAssociatedIpv4NetworksModifiedBy' DESC 'Person who
  last modified the inetAssociatedIpv4Networks attribute.'
  EQUALITY distinguishedNameMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE USAGE
  distributedOperation )

inetAssociatedIpv4NetworksModifiedDate
( 1.3.6.1.4.1.7161.1.5.10 NAME
  'inetAssociatedIpv4NetworksModifiedDate' DESC 'Last
  modification date of the inetAssociatedIpv4Networks
  attribute.' EQUALITY generalizedTimeMatch ORDERING
  generalizedTimeOrderingMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE USAGE
  distributedOperation )

inetAssociatedIpv6Networks
( 1.3.6.1.4.1.7161.1.5.11 NAME 'inetAssociatedIpv6Networks'
  DESC 'The IPv6 networks associated with this entry.'
  EQUALITY caseIgnoreMatch SYNTAX inetIpv6NetworkSyntax )

inetAssociatedIpv6NetworksModifiedBy
( 1.3.6.1.4.1.7161.1.5.12 NAME
  'inetAssociatedIpv6NetworksModifiedBy' DESC 'Person who
  last modified the inetAssociatedIpv6Networks attribute.'
  EQUALITY distinguishedNameMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE USAGE
  distributedOperation )

inetAssociatedIpv6NetworksModifiedDate
( 1.3.6.1.4.1.7161.1.5.13 NAME
  'inetAssociatedIpv6NetworksModifiedDate' DESC 'Last
  modification date of the inetAssociatedIpv6Networks
  attribute.' EQUALITY generalizedTimeMatch ORDERING
  generalizedTimeOrderingMatch SYNTAX
  1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE USAGE
  distributedOperation )
```

5.2.3. Example

An example of the inetAssociatedResources object class is shown in Figure 4 below.

```
cn=192.0.2.0/24,cn=inetResources,dc=example,dc=com
[top object class]
[inetResources object class]
[inetIpv4Network object class]
[inetAssociatedResources object class]
|
+-attribute: description
| value: "The example.com network"
|
+-attribute: inetAssociatedAsNumbers
| value: "65535"
|
+-attribute: inetAssociatedDnsDomains
  value: "example.com"
```

Figure 4: The inetAssociatedResources attributes associated with the 192.0.2.0/24 IPv4 network entry.

5.3. The referral Object Class

This document allows the use of the referral object class to define subordinate reference referrals and continuation reference referrals for inetResources container entries and all of the resource-specific entries.

Referral entries MUST conform to the schema specification defined in [namedref]. In particular, referral entries MUST NOT contain any user-definable attributes other than the mandatory "cn" naming attribute and the mandatory "ref" operational attribute. By extension, referral entries MUST be leaf nodes, and MUST NOT have any subordinate entries defined at the referral source.

Furthermore, in order to facilitate programmatic access to this data, LDAP URLs provided in ref attributes MUST refer to entries which use the same object classes as the source entry, MUST refer to an entry in a DIT which uses the domainComponent object class syntax ("dc="), and MUST specify the LDAP protocol-type.

5.4. Object Class and Attribute Permissions

The information presented through the LDAP-WHOIS service will be used for many operational and problem-resolution purposes. In order for this information to be suitable for this purpose, it must be accurate. In order to ensure the veracity of the information, a minimal set of operational guidelines are provided in this section. For the most part, these rules are designed to prevent unauthorized modifications to the data.

Note that these rules only apply to data which is willingly provided; no data is required to be entered, but where the data is provided, it MUST be accurate, and it MUST be secured against unauthorized modifications.

* The inetResources container entry and all of the resource-specific subordinate entries within every public DIT that provides LDAP-WHOIS resources SHOULD have anonymous read-only access permissions, and SHOULD NOT have anonymous add, delete or modify permissions.

* With the exception of contact-related attributes from the inetOrgPerson object class, each attribute MAY have whatever restrictions are necessary in order to suit local security policies, government regulations or personal privacy concerns. When the inetOrgPerson object class is used to provide contact details, the mail attribute MUST be defined, SHOULD be valid, SHOULD have read-only anonymous access, and SHOULD NOT have anonymous add, delete or modify permissions.

  By using the inetOrgPerson object class, it is expected that existing contact-related entries can be reused. If reusing these entries is undesirable or unfeasible, entries with the necessary access SHOULD be made available.

  Note that contact pointers are entirely optional and are not required to exist. However, where they exist, they MUST comply with the above requirements.

* End-users and implementers SHOULD provide anonymous access to the creatorsName, createTimestamp, modifiersName and modifyTimestamp operational attributes associated with each entry in the inetResources branch, since this information is useful for determining the age of the information.

* Server administrators MAY define additional add, delete or modify permissions for authenticated users, using any LDAPv3 authentication mechanisms they wish. In particular, delegation entities MAY provide for the remote management of delegated resources (such as assigning modification privileges to the managers of a particular delegated domain or address block), although this is entirely optional, and is within the sole discretion of the delegation body.

External applications SHOULD NOT make critical decisions based on the information provided through this service without having reason to trust the veracity of the information. Clients and users SHOULD limit the use of unknown or untrusted information to routine purposes.

6. Search and Match Filters

LDAP search filters are fairly flexible, in that they allow for a wide variety of configurable elements, such as the maximum number of entries which are returned, the type of comparison operation that needs to be performed, and other details. In order to promote interoperability, default values are defined here for many of these elements, although these defaults are only applicable when they are used with the LDAP-WHOIS service.

In particular, this document defines several suggested and mandatory search filter qualifiers, which are described in detail in section 6.1. This document also defines extensibleMatch filter definitions which MUST be implemented whenever the associated resource types defined in this document are implemented by an LDAP-WHOIS client or server. These filter definitions are provided in section 6.2 below.

6.1. Search Filter Expressions

Section 4.5.1 of RFC 2251 defines the LDAP search request specification, although it does not provide guidelines or recommended values for the filter settings. In an effort to promote interoperability among LDAP-WHOIS clients and servers, this document defines some recommended and mandatory values for searches within the LDAP-WHOIS service.

NOTE: These rules ONLY apply to the LDAP-WHOIS search operations in particular. Any queries for other resources

(such as requests for inetOrgPerson contact entries) MUST NOT impose these restrictions. Also note that other documents which define additional resource types can also define different restrictions, and those definitions will take preference over these guidelines.

Generic LDAP clients may be used to browse and search for data, and in those cases, these rules are not likely to be followed. As such, servers MUST be prepared to enforce these rules independently of the client settings.

The values of an LDAP search filter should be as follows:

* Search base. The DIT to be used in a search will vary for each query operation. The methodology for determining the current search base for a query is defined by the query-processing protocols described in section 7, although LDAP-WHOIS searches are normally constrained to the "cn=inetResources" container of a particular DIT.

* Scope. In order to support continuation reference referrals (which are defined as referral entries beneath a resource-specific entry), clients MUST use a sub-tree scope for LDAP-WHOIS searches. Servers MUST NOT arbitrarily limit the scope of search operations.

* Dereference aliases. Although the LDAP-WHOIS service does not make direct use of alias entries, they are not prohibited. Clients SHOULD set the Dereference Aliases option to "Always" for LDAP-WHOIS searches. Servers SHOULD dereference any aliases which are encountered, where this is feasible (in particular, where the alias refers to another DIT on the same server).

* Size limit. The size limit field specifies the maximum number of entries that a server should return. For the LDAP-WHOIS service, this setting SHOULD be set to a value between 25 and 100. This range ensures that the client is capable of receiving a sufficient number of entries and continuation references in a single response, but also works to prevent runaway queries that match everything (such as searches for "com", which can match every inetDnsDomain entry in the "cn=inetResources,dc=com" container). Servers MAY truncate answer sets to 100 responses if the client specifies a larger value.

* Time limit. The time limit field specifies the maximum number of seconds that a server should process the search. For the LDAP-WHOIS service, this setting SHOULD be set to a value between 10 and 60 seconds. This range ensures that the server is able to process a sufficient number of entries, but also works to prevent runaway queries that match everything. Servers MAY stop processing queries after 60 seconds if the client specifies a larger value.

* Types-only. The types-only setting is a Boolean flag which controls whether or not attribute values are returned in the answer sets. Since excessive queries are likely to be more burdensome than larger answer sets, this setting SHOULD be set to FALSE. Resource-constrained clients (such as PDAs) MAY set this value to TRUE, but these clients MUST be prepared to issue the necessary subsequent queries.

* Filter. The search operation will depend on the type of data being queried. For LDAP-WHOIS queries, the filter MUST use the matching rules defined in section 6.2 for the relevant resource type. Other resource-specific documents may define their own handling rules.

  Note that the extensible match filters defined in this document MUST be supported by LDAP-WHOIS clients and servers. LDAP-WHOIS servers MAY also support additional sub-string filters, soundex filters, or any other filters they wish (these may be required for generic LDAP clients), although LDAP-WHOIS clients MUST NOT expect any additional filters to be available.

* Attribute list. Clients MAY restrict the list of attributes which are returned in searches, but are discouraged from doing so without cause.

6.2. Matching Filter Definitions

Each of the object classes defined in this document have their own search criteria which MUST be used whenever a collection of resource pools need to be searched. In this model, resource types are specified during the search operation, and most of the resource types have extensibleMatch definition which are used whenever the available resources need to be searched.

For example, if a user wishes to find the inetIPv4network object class entry associated with a specific IPv4 address, then the inetIpv4NetworkMatch extensibleMatch filter MUST be specified by the client, and MUST be used by the server when attempting to locate the relevant inetIpv4Network entry.

The inetResources object class can be searched with simple equalityMatch filters, and do not require dedicated extensibleMatch filters, although they do have specific handling rules.

In order to ensure that all of the relevant entries (including any referrals) are found, the search filters for these resources MUST specify two distinct elements: the object class of the resource being queried, and the naming element of the resource specified as a distinguished name attribute.

For example, using the notation format described in RFC 2254, the search filter expression for the inetOrgPerson entry associated with "cn=admins,ou=admins,dc=example,dc=com" would be structured as "(&(objectclass=inetOrgPerson)(cn:dn:=admins))", using "ou=admins,dc=example,dc=com" as the search base. This would find all entries with the object class of inetOrgPerson (including all of the referral entries for inetOrgPerson entries) where the distinguished name contained the "cn" attribute of "admins".

The input source and search base for these matches will vary according to the query being processed, but whenever an equalityMatch is called for during query processing, the above methods MUST be used in order to ensure that all of the related entries are located.

Response entries MAY be fully-developed entries, or MAY be referrals generated from entries which have the referral object class 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.

7. Query Processing Models

The LDAP-WHOIS service uses three different query-processing models. These are the "top-down" model which initiates the query process at the top-level of a DNS delegation hierarchy, a "bottom-up" model which directs queries to user-managed servers, and a "targeted" search model which is functionally identical to traditional LDAP searches. Furthermore, any of these mechanisms may be redirected to other servers, either through simple DNS query processing, or by way of LDAP redirections (including subordinate reference referrals, continuation reference referrals, attribute references, or labeledURI attributes).

Each of the three query models are appropriate to different usage environments. For example, the top-down model is best suited for searches about global resources which are centrally managed and delegated (such as IP addresses and DNS domains), and where delegation information is a critical element of the resource data. Meanwhile, the bottom-up model is most appropriate for those resources which are managed by the end-users directly, and which are not managed from a centralized delegation authority (this includes information such as private keys, mail servers, and other leaf-node resources). Finally, the targeted model is best suited for explicit queries where a particular resource is supposed to exist with a known DN (such as with contact pointers).

LDAP-WHOIS clients and servers MUST implement all three models. Clients MUST default to using the top-down model, but clients MUST also provide a user-selectable option for the disposition of individual queries.

7.1. Top-Down Processing

The top-down model is primarily suited for locating Internet resources which are centrally managed and delegated. The top-down model is similar to other distributed WHOIS protocols in this regard, with the principle difference being the use of LDAP for standardized syntaxes, data and referrals, rather than using a specialized protocol specifically for this application.

The top-down model uses an input string to construct an LDAP assertion value and search base, with DNS queries being used to locate the LDAP servers associated with the appropriate top-level delegation entity. Once this process completes, an extensible

match query is issued to the specified servers. The query may also be redirected through the use of LDAP referrals, if additional data is known to exist elsewhere.

For example, a top-down search for the domain name of "www.example.com" would result in the client building an inetDnsDomainMatch extensible match query with the search base of "cn=inetResources,dc=com", and with the client issuing a DNS query for the LDAP servers associated with "com" domain. If the queried server had information about the "www.example.com" resource, it would be returned as answer data. If the server knew of other sources of information about the resource (such as the registrar for the domain, or the entity operating the domain, or both), continuation reference referrals could be returned. Any of the subsequent queries could return additional answers and/or referrals, according to the data they had.

IP address blocks and AS numbers are processed in a similar fashion. If a client needed to locate information about the "192.0.2.14/32" IPv4 address, it would begin the process by building a reverse-lookup DNS domain name from the input string, and then issuing a DNS query for the LDAP servers associated with the "arpa" top-level domain. Once a server had been located, an LDAP query with the assertion value of "192.0.2.14/32" would be submitted with a search base of "cn=inetResources,dc=arpa". The server would return data and/or referrals, with this process repeating until the query string had been completely processed.

Note that entries for the inetResources and inetOrgPerson object classes are not searchable with this model, since they do not have centralized delegation authorities. One of the other search models MUST be used for those resource types.

7.1.1. Processing steps

The steps for processing top-down queries are described below:

a. Determine the input type (DNS Domain, IPv4 Address, etc.)

b. Determine the authoritative domain name for the query.

   1. Separate the input string into discrete elements where this is possible. For a DNS domain name of "www.example.com", this would be "www", "example" and "com". For the IPv4 network number of "192.0.2.14",

this would be "192", "0", "2" and "14". AS numbers only have a single value and require no separation. Do not discard the original query string.

2. IP addresses and AS numbers require additional conversion. For IPv4 addresses, strip off the prefix and convert the input string into a reverse-lookup DNS domain name by reversing the order of the octets and appending "in-addr.arpa" to the right of the domain name. For IPv6 addresses, strip off the prefix and reverse the nibble order of the address (where each nibble is represented by a single hexadecimal character), and append "ip6.arpa". For AS numbers, append only the "arpa" domain name.

c. Form the LDAP search base for the query.

1. Convert the right-most element from the domain name formed in step 7.1.1.b above into a domainComponent DN (such as "dc=com" or "dc=arpa"). This represents the DIT for the current query.

2. Append the "cn=inetResources" RDN to the front of the domainComponent syntax ("cn=inetResources,dc=com"). This will form the fully-qualified search base for the LDAP query.

d. Locate the LDAP servers associated with the resource by processing the domain name formed in step 7.1.1.b above through the SRV query steps provided in section 7.4.5.

e. If the SRV lookup succeeds:

1. Choose the best LDAP server, using the weighting formula described in RFC 2782.

2. Construct the LDAP search filter according to the rules specified in section 6.1, using the appropriate matching rule from section 6.2.

3. Formulate the LDAP search using the search base and search filter constructed above. For example, if the input query string was for "www.example.com", then the client would begin the process by submitting an inetDnsDomainMatch extensibleMatch search with the assertion value of "www.example.com", and with a

search base of "dc=inetResources,dc=com". Similarly, if the input query string was "192.0.2.14", then the client would begin the process by submitting an inetIpv4NetworkMatch extensibleMatch search with the assertion value of "192.0.2.14/32", and with the search base of "cn=inetResources,dc=arpa".

4. Submit the search operation to the chosen server and port number. If the operation fails, report the failure to the user and exit. Otherwise, display any answer data which is returned.

5. If the answer data contains a subordinate reference referral or a continuation reference referral, new query processes MUST be spawned.

   For subordinate reference referrals, process the URLs according to the rules described in section 7.4.1 and restart the query process at step 7.1.1.e.2. For each continuation reference referral, display the answer data received so far, process the LDAP URLs according to the rules described in section 7.4.3 and start new query processes for each referral at step 7.1.1.e.2, appending the output from these searches to the current output.

   Any additional subordinate reference referrals or continuation reference referrals which are encountered from any subsequent searches will need to be processed in the same manner as specified above, until no additional referrals are received.

f. If the SRV lookup fails (where failure is defined as any DNS response message other than an answer), report the failure to the user and exit the current search operation.

7.1.2. Top-Down example

In the example below, the user has entered a search string of "www.example.com" and has indicated that the query is for a DNS domain name.

a. The input string is broken into the discrete label components ("www", "example" and "com").

b. The right-most label ("com") is used to form the DNS SRV lookup ("_ldap._tcp.com"), in order to find the LDAP servers authoritative for the delegation hierarchy.

c. One of the LDAP servers is contacted, and an inetDnsDomainMatch search filter is submitted with the assertion value of "www.example.com" and a search base of "cn=inetResources,dc=com".

d. The server responds with a continuation reference referral URL of "ldap://ldap.netsol.com/cn=example.com, cn=inetResources,dc=netsol,dc=com", indicating that the domain delegation is managed under the "dc=netsol,dc=com" DIT, and is hosted at the "ldap.netsol.com" server. The client uses this information to start a new query. No additional data was provided for the client to display.

e. An inetDnsDomainMatch extensibleMatch search is submitted to "ldap.netsol.com", using the search base of "cn=example.com,cn=inetResources,dc=netsol,dc=com".

f. The queried server returns the information that it has. No additional referrals are provided. The client displays the data and exits the query.

7.2. Bottom-Up Processing

The bottom-up model is best used when a leaf-node resource needs to be queried, and where an LDAP-WHOIS server is expected to be able to answer the query. In this case, navigating down through a delegation hierarchy would be either fruitless or inefficient. For example, information about a mail domain would be more efficient in the bottom-up model, since there is no global delegation body for Internet mail (the DNS domains are delegated, but the message routing is specific to the operational entities responsible for the domain name). The bottom-up model can also be used for DNS domain names, IPv4 addresses, and IPv6 addresses, although this will generally prove to be less useful than top-down queries, given the limited number of user-managed servers deployed.

The bottom-up model uses an input string to construct an LDAP assertion value and search base, with DNS queries being used to locate the LDAP servers which are associated with the management entity that is directly responsible for the resource in question. Once this process completes, an extensible match query is issued

to the specified servers. The query may also be redirected through the use of LDAP referrals, if additional data is known to exist elsewhere.

For example, a bottom-up search for the domain name of "www.example.com" would result in the client building an inetDnsDomainMatch extensible match query with the search base of "cn=inetResources,dc=www,dc=example,dc=com", and with the client issuing a DNS query for the LDAP servers associated with "www.example.com" domain. If the DNS lookup failed, the client would issue a subsequent query for the LDAP servers associated with the "example.com" domain, and so forth, until a server had been located. If the queried server had information about the "www.example.com" resource, it would be returned as answer data. If the server knew of other sources of information about the resource (such as the registrar for the domain, or the entity operating the domain, or both), continuation reference referrals could be returned. Any of the subsequent queries could return additional answers and/or referrals, according to the data they had.

IP address blocks are processed in a similar fashion. If a client needed to locate information about the "192.0.2.14" IPv4 address, it would begin by issuing a DNS query for the LDAP servers responsible for the "14.2.0.192.in-addr.arpa" domain name, with the left-most labels being truncated as the search for an authoritative server was broadened. Once a server had been located, an inetIpv4NetworkMatch extensibleMatch search with the assertion value of "192.0.2.14/32" would be submitted. If the server knew of any information about that resource, it would return data or a referral, with this process repeating until the query string had been processed as completely as possible.

Note that entries for inetAsNumber and inetOrgPerson object classes are not searchable with this model, since they are not represented in the DNS delegation hierarchy. One of the other search models MUST be used for those resource types.

7.2.1. Processing steps

The steps for processing bottom-up queries are described below:

a. Determine the input type (DNS Domain, IPv4 Address, etc.)

b. Determine the authoritative DNS domain for the resource.

1. Separate the input string into discrete elements where this is possible. For a DNS domain name of "www.example.com", this would be "www", "example" and "com". For the IPv4 network number of "192.0.2.14", this would be "192", "0", "2" and "14". Do not discard the original query string.

2. IP addresses require additional conversion. For IPv4 addresses, strip off the prefix and convert the input string into a reverse-lookup DNS domain name by reversing the order of the octets and appending "in-addr.arpa" to the right of the resulting sequence. For IPv6 addresses, strip off the prefix and reverse the nibble order of the address (where each nibble is represented by a single hexadecimal character), and append "ip6.arpa" to the right of the resulting sequence.

c. Form the LDAP search base for the query.

1. Convert the domain name formed in step 7.2.1.b above into a domainComponent DN (such as "dc=www,dc=example,dc=com" or "dc=0,dc=2,dc=0,dc=192, dc=in-addr,dc=arpa"). This represents the DIT for the current query.

2. Append the "cn=inetResources" RDN to the left of the domainComponent syntax (perhaps resulting in "cn=inetResources,dc=www,dc=example,dc=com"). This will become the search base for the LDAP query.

d. Locate the LDAP servers associated with the resource by processing the domain name formed in step 7.2.1.b above through the SRV query steps provided in section 7.4.5.

e. If the SRV lookup fails with an NXDOMAIN response code (as described in RFC 2308), then the domain name used for the SRV lookup does not exist, and a substitute LDAP server and search base must be identified. This process involves determining the parent zone for the domain name in question, issuing an SRV lookup for that zone, and using the domain name of the zone as the new LDAP search base, with this process repeating until a search base can be located, or until a critical failure forces an exit.

1. Remove the left-most label from the domain name formed in step 7.2.1.b.

2. If this process has already resulted in a query domain name at a top-level domain such as "com" or "arpa", convert the query domain name to "." (to signify the root domain).

3. If the queried domain name is already set to ".", the query can go no higher (this most likely indicates a malformed DNS configuration, a connectivity problem, or a typo in the query). Exit and report the failure to the user.

4. Restart the process at step 7.1.1.c, using the domain name formed above. Repeat until a server is located or a critical failure forces an exit.

   For example, if the original input string of "www.example.com" resulted in a failed SRV lookup for "_ldap._tcp.www.example.com", then the first fallback SRV query would be for "_ldap._tcp.example.com", and the next fallback query would be for "_ldap._tcp.com", possibly being followed by "_ldap._tcp.", and possibly resulting in failure after that.

f. If the SRV lookup succeeds:

1. Choose the best LDAP server, using the weighting formula described in RFC 2782.

2. Construct the LDAP search filter according to the rules specified in section 6.1, and choose the appropriate matching rule from section 6.2.

3. Formulate the LDAP search using the search base and search filter constructed above. For example, if the input query string was for "www.example.com", then the client would begin the process by submitting an inetDnsDomainMatch extensibleMatch search with the assertion value of "www.example.com", with the search base of "cn=inetResources,dc=www,dc=example,dc=com".

4. Submit the search operation to the chosen server and port number. If the operation fails, report the

failure to the user and exit. Otherwise, display any answer data which is returned.

5. If the answer data contains a subordinate reference referral or a continuation reference referral, new query processes MUST be spawned.

   For subordinate reference referrals, process the URLs according to the rules described in section 7.4.1 and restart the query process at step 7.2.1.f.2. For each continuation reference referral, display the answer data received so far, process the LDAP URLs according to the rules described in section 7.4.3 and start new query processes for each referral at step 7.2.1.f.2, appending the output from these searches to the current output.

   Any additional subordinate reference referrals or continuation reference referrals which are encountered from any subsequent queries will need to be processed in the same manner as specified above, until no additional referrals are received.

g. If a fatal DNS error condition occurs, report the error to the user and stop processing the current query. A fatal DNS error is any response message with an RCODE of FORMERR, SERVFAIL, NOTIMPL, or REFUSED, or where a query results in NODATA (implying that an "_ldap._tcp" domain name exists but it doesn't have an SRV resource record associated with it, which is most likely a configuration error).

7.2.2. Bottom-Up example

In the example below, the user has entered a search string of "www.example.com" and has indicated that the query is for a DNS Domain Name.

a. The query string is used to form the DNS SRV lookup ("_ldap._tcp.www.example.com"), in order to find the LDAP servers authoritative for that domain name.

b. The SRV lookup fails with NXDOMAIN, indicating that the queried domain name does not exist.

c. The client creates a new query for the parent domain ("_ldap._tcp.example.com"), which succeeds.

d. The client contacts one of the servers, and issues an inetDnsDomainMatch extensibleMatch search with the assertion value of "www.example.com", and with the search base of "cn=inetResources,dc=example,dc=com".

e. The server returns a continuation reference referral of "ldap://ldap.example.net/cn=server1.example.net, cn=inetResources,dc=example,dc=net", indicating that the queried resource is a referral for a web hosting server at Example Networks. The client uses this information to start a new query. No additional data was provided for the client to display.

f. An inetDnsDomainMatch extensibleMatch search is submitted to the "ldap.example.net" server, using the search base of "cn=server1.example.net,cn=inetResources,dc=example,dc=net"

g. The queried server returns the information that it has. No additional referrals are provided. The client displays the data and exits the query.

7.3. Targeted Search Processing

The targeted search model is similar to the bottom-up query model described in the preceding section, except that it does not provide fallback processing of DNS domain names. In this regard, the targeted search model is closely similar to the traditional LDAP searching model, in that a client queries a specified LDAP server for a specific entry, under the assumption that the resource exists at that location. If the server or resource does not exist, the entire query fails.

For this reason, the targeted search model is not suitable for search operations against generic Internet resources, but instead is mostly suitable for searches against known entries which are presumed to exist at a known location. In terms of the LDAP-WHOIS service in particular, this includes inetOrgPerson entries which are provided in contact-related attributes. However, the targeted search model can be used for any resource type, and it can be useful for diagnosing problems with resource types. For this reason, clients SHOULD support this model for use with all known resource types.

The targeted search takes an LDAP URL as the query input (along with the resource-type identifier), and uses the URL to determine the query server, the search base, and the assertion value.

7.3.1. Processing steps

The steps for processing targeted search queries are described below:

a. Process the LDAP URLs according to the continuation reference referral handling rules described in section 7.4.3. This process will determine the servers, search base and assertion value of the query.

b. If this process succeeds:

1. Construct the LDAP search filter according to the rules specified in section 6.1, and choose the appropriate matching rule from section 6.2.

2. Submit the search operation to the chosen server and port number. If the operation fails, report the failure to the user and exit. Otherwise, display any answer data which is returned.

3. If the answer data contains a subordinate reference referral or a continuation reference referral, new query processes MUST be spawned.

   For subordinate reference referrals, process the URLs according to the rules described in section 7.4.1 and restart the query process at step 7.3.1.b. For each continuation reference referral, display the answer data received so far, process the LDAP URLs according to the rules described in section 7.4.3 and start new query processes for each referral at step 7.3.1.b.

   Any additional subordinate reference referrals or continuation reference referrals which are encountered from any subsequent queries will need to be processed in the same manner as specified above, until no additional referrals are received.

c. If this process fails, report the failure to the user and exit the current search operation.

7.3.2. Targeted search example

In the example below, the user has provided an LDAP URL of "ldap://ldap.example.com/cn=admins,ou=admins,dc=example,dc=com", and has indicated that the query is for an inetOrgPerson entry.

a. The query string is used to form a DNS lookup of the specified server ("ldap.example.com").

b. The client contacts the servers, and issues a search for "(&(objectclass=inetOrgPerson)(cn:dn:=admins))", with a search base of "ou=admins,dc=example,dc=com".

c. The queried server returns the information that it has. No additional referrals are provided. The client displays the data and exits the query.

7.4. Supplemental Query Processing Mechanisms

During the course of normal query processing, an LDAP-WHOIS client may need to use additional mechanisms to complete an operation, such as processing a URL received from a redirect operation, or issuing DNS SRV lookups against a provided domain name.

7.4.1. URL processing

URL processing in this specification is a function of both content and context. Different attributes and result codes provide different types of URLs, and the disposition of these URLs will depend on the query-resolution process currently being executed.

On the content front, this specification allows three different forms of URLs to appear throughout this service: labeledURI attribute values, attribute references, and referral messages. Each of these usage scenarios have slightly different restrictions and formats.

* The labeledURI attribute is included with the inetResources object class for the purpose of informing end-users of a generic resource associated with an entry (such as an

organization's home page). The labeledURI attribute is defined in RFC 2079 for the purpose of storing generic URLs as attribute values, and uses a two-part syntax of "url://any.host:port/any/path description", with the "description" string providing a free-text description of the target specified by the URL.

* Attribute references also use the two-part format of the labeledURI attribute, but with some additional restrictions as described in section 4.5 of this document.

* Subordinate and continuation reference referrals use URLs for the purpose of providing referral targets. The URL format specified in [namedref] is also an explicit subset of the labeledURI format, but without the "description" free-text block. When used with the LDAP-WHOIS service, subordinate and continuation referrals are subject to some additional rules as described in section 4.5 of this document.

Non-compliance with the requirements provided in section 4.5 amounts to an error, and is sufficient cause for a client to stop processing a query.

7.4.2. Subordinate reference referrals

Subordinate reference referrals and their schema are defined in [namedref]. Subordinate reference referrals use the SearchResultDone response with a Referral result code, which is defined and described in section 4.1.11 of RFC 2251. Subordinate reference referrals use a subset of the labeledURI syntax as defined in RFC 2079, and use the syntax definitions from RFC 2255 when LDAP URLs in particular are provided, although section 4.5 of this document also defines additional restrictions on the allowable URL syntax.

In the context of the LDAP-WHOIS service, subordinate reference referrals are returned when the search base specified in a search operation exists as a referral object class with the ref attribute pointing to some other entry, resulting in queries with that search base being answered with a SearchResultDone referral response. This condition means that the current search operation cannot proceed past this point, and the search MUST be restarted. This will most often occur when the inetResources entry for a DIT has been redirected to another DIT, but it can also happen after

continuation reference referrals have been followed or after targeted searches have been issued, and where the queried entry exists as a referral to some other entry.

The procedure for processing URLs returned in a subordinate reference referral is as follows:

a. RFC 2251 allows multiple URLs to be provided, although the URLs are not provided with any "preference" or "weighting" values. If a set of URLs are provided, only one of the URLs need to be tried (implementations MAY perform additional queries in an attempt to recover from temporary failures, although this is not required). Select one of the URLs at random ("round-robin"), and continue to the next step in the process.

b. Extract and discard any description text which may have been provided with the URL.

c. Validate the protocol label. This specification only supports the use of the LDAP service type. URLs with other protocol identifiers are to be treated as malformed.

d. Extract the host identifier element and perform any DNS lookups which may be required. URLs without host identifiers are to be treated as malformed.

e. Extract the port number provided with the URL, and set it aside for use with the subsequent connection attempt. If no port number has been provided in the URL, use the default port numbers associated with the protocol, as discovered in step 7.4.2.c.

f. Extract the path element from the URL for use as the search base of the subsequent search operation. URLs without path elements are to be treated as malformed.

g. Restart the current search operation, using the LDAP server from step 7.4.2.d, the port number from step 7.4.2.e, and the search base formed in step 7.4.2.f.

7.4.3. Continuation reference referrals

Continuation reference referrals and their schema are defined in [namedref]. Continuation reference referrals use the

SearchResultReference response, which is defined and described in section 4.5.3 of RFC 2251. Continuation reference referrals use a subset of the labeledURI syntax as defined in RFC 2079, and use the syntax definitions from RFC 2255 when LDAP URLs in particular are to be provided, although section 4.5 of this document also defines additional restrictions on the allowable URL syntax.

For this service, continuation reference referrals are returned when the search base specified in a search operation exists, but one or more of the answer elements exist as referral object classes, resulting in one or more SearchResultReference responses. This condition means that the current search operation has partially succeeded, but that additional searches SHOULD be started in order for all of the answer data to be retrieved (in many cases, no answer data will be provided, and in those situations, new queries will be required for any data to be retrieved). This will occur whenever the assertion value of a search has matched a resource entry which is being managed by another DIT, and can occur with any of the search operations described in this document.

Multiple continuation reference referrals MAY be returned in response to a search, and each of them MUST be processed in order for all of the answer data to be retrieved.

The procedure for processing the URLs returned in a continuation reference referral is as follows:

a. RFC 2251 allows multiple URLs to be provided, although the URLs are not provided with any "preference" or "weighting" values. If a set of URLs are provided, only one of the URLs need to be tried (implementations MAY perform additional queries in an attempt to recover from temporary failures, although this is not required). Select one of the URLs at random ("round-robin"), and continue to the next step in the process.

b. Extract and discard any description text which may have been provided with the URL.

c. Validate the protocol label. This specification only supports the use of the LDAP service types. URLs with other protocol identifiers are to be treated as malformed.

d. Extract the host identifier element and perform any DNS lookups which may be required. URLs without host identifiers are to be treated as malformed.

e. Extract the port number provided with the URL, and set it aside for use with the subsequent connection attempt. If no port number has been provided in the URL, use the default port numbers associated with the protocol, as discovered in step 7.4.3.c.

f. Extract the path element from the URL for use as the search base of the subsequent search operation. URLs without path elements are to be treated as malformed.

g. Extract the left-most RDN from the search base constructed in step 7.4.3.e, and delete the naming attribute label. The resulting string will be used as the assertion value for the subsequent search operation. For example, if the path element from a URL provided a distinguished name of "cn=example.com,cn=inetResources,dc=example,dc=com", then the "cn=example.com" RDN would be used to form an assertion value of "example.com".

h. Start a new search operation, using the LDAP server from step 7.4.3.d, the port number from step 7.4.3.e, the search base formed in step 7.4.3.f, and the assertion value formed in step 7.4.3.g.

7.4.4. Attribute references

Attribute references are defined in this document as attributes which provide URLs as pointers to contextually related information. These are not referrals, but instead are simple URLs returned as attribute values. In particular, this document defines multiple contact-related attributes which provide these URLs. Other documents may also define attributes which reuse the URL format defined here, or may define their own URL rules, as needed.

For this service, attribute reference URLs are returned when an entry has an attribute defined which uses them. Attribute references are not referrals, and do not require additional processing. Clients MAY automatically start new search operations when an attribute reference is encountered, or they MAY delay processing until a user requests the action.

The procedure for processing the URLs returned in an attribute reference is as follows:

a. RFC 2251 allows multiple URLs to be provided, although the URLs are not provided with any "preference" or "weighting" values. If a set of URLs are provided, only one of the URLs need to be tried (implementations MAY perform additional queries in an attempt to recover from temporary failures, although this is not required). Select one of the URLs at random ("round-robin"), and continue to the next step in the process.

b. Extract and discard any description text which may have been provided with the URL.

c. Validate the protocol label. This specification only supports the use of LDAP service types. URLs with other protocol identifiers are to be treated as malformed.

d. Extract the host identifier element and perform any DNS lookups which may be required. URLs without host identifiers are to be treated as malformed.

e. Extract the port number provided with the URL, and set it aside for use with the subsequent connection attempt. If no port number has been provided in the URL, use the default port numbers associated with the protocol, as discovered in step 7.4.4.c.

f. Extract the path element from the URL for use as the search base of the subsequent search operation. URLs without path elements are to be treated as malformed.

g. Extract the left-most RDN from the search base constructed in step 7.4.4.e, and delete the naming attribute label. The resulting string will be used as the assertion value for the subsequent search operation. For example, if the path element from a URL provided a distinguished name of "cn=example.com,cn=inetResources,dc=example,dc=com", then the "cn=example.com" RDN would be used to form an assertion value of "example.com".

h. Determine the object class filter to be used with the assertion value. This will depend on the attribute which provided the attribute reference. The contact-related

attributes defined in this document refer to inetOrgPerson object class entries.

i. Start a new search operation, using the LDAP server from step 7.4.4.d, the port number from step 7.4.4.e, the search base formed in step 7.4.4.f, the assertion value formed in step 7.4.4.g, and the new object class filter formed in step 7.4.4.h.

7.4.5. SRV processing

The query models described in this document make use of DNS SRV resource records whenever a new query process is started, as a way to locate the LDAP servers associated with a DIT.

The procedure for constructing this SRV lookup is as follows:

a. Construct an SRV-specific label pair for the service type. For LDAP queries, this will be "_ldap._tcp".

b. Append the SRV label pair to the left of the input domain name. In the case of an LDAP query for "example.com", this would result in an SRV-specific domain name of "_ldap._tcp.example.com".

c. Issue a DNS query for the SRV resource records associated with the domain name formed in step 7.4.5.b.

Multiple SRV resource records may be returned in response to a query. Each resource record identifies a different connection target, including the domain name of a server, and a port number for that server. The port number specified in a SRV resource record MUST be used for any subsequent bind and search operations.

SRV resource records provide "priority" and "weight" values which MUST be used to determine the preferred server. If a server is unavailable or unreachable, a connection attempt must be made to the next-best server in the answer set.

Refer to RFC 2782 for a detailed explanation of SRV resource records and their handling.

8. Internationalization and Localization

The LDAP-WHOIS model uses the internationalization and localization services provided by LDAPv3. In this regard, LDAP-WHOIS clients do not need to implement any special services in order to process and display internationalized attribute data, since the attribute types already provide direct support for internationalized data.

LDAP-WHOIS clients may have some localization or language-specific presentation issues with regards to attribute names, in that the names of the attributes may need to be localized for specific markets. However, these services are outside the scope of the protocol operations. Any such requirements must be dealt with according to the services available on the client platform.

In the case of legacy WHOIS servers which gateway requests between TCP port 43 and the LDAP-WHOIS service, the input and output language and/or locale codes MAY be specified by server-specific options, although these mechanisms must be defined as part of the WHOIS protocol for any widespread consistency to be possible, and are therefore beyond the scope of this document.

9. DIT Replication

All DITs which provide data for global Internet resources SHOULD be replicated across two or more servers. Each of the authoritative LDAP servers for the managed resource MUST be specified with a unique DNS SRV resource record for the domain name associated with the top-level resource assignment space.

For example, the top-level "com" delegation space SHOULD have two or more SRV resource records associated with the "_ldap._tcp.com" domain name, with each entry referring to separate LDAP servers, and with each of those servers maintaining accurate copies of the "dc=com" DIT (within reasonable timeliness). Similarly, the top-level " arpa" domain which is used by the IPv4 and IPv6 delegation trees SHOULD provide two or more SRV resource records for the "_ldap._tcp.arpa" domain name, as should the "in-addr.arpa" and "ip6.arpa" domain hierarchies.

DITs which serve multiple organizations SHOULD also be replicated. For example, an ISP which provides LDAP-WHOIS services for their customers SHOULD also follow these same rules, since outages of

those servers will affect multiple parties. Leaf-node DITs associated with an user-managed resource MAY be replicated, and are encouraged to do so.

Similarly, any referrals which present URLs as answer data SHOULD provide multiple URLs, each of which reference different hosts on different networks. For leaf-node referrals, attribute references, and labeledURI references, this behavior MAY be relaxed, although it is still encouraged.

Note that the most effective replication strategy will be for entities to replicate their DITs with the delegation parents, as this will allow queries for those resources to be processed by the parent servers (thereby eliminating the need for referral queries). In many cases, this will not be feasible (the servers for the "dc=com" DIT cannot be expected to host replicas of every subordinate DIT), but it is encouraged where practical.

10. Transition Issues

There are a handful of areas where the proposed service does not fully match with all of the existing WHOIS service offerings. These areas are discussed in more detail below.

10.1. NIC Handles

NIC handles represent a historical method of WHOIS lookups, tying unique identifiers to a specific record in a specific database. Given that the model proposed in this document uses a distributed lookup system rather than isolated databases, the NIC handle model is no longer necessary. Furthermore, given the limited global usability of NIC handles, they should be deprecated.

However, NIC handles are an important part of the legacy service, and their continued usage is likely to be desired in at least some instances. There are two possible workarounds for this problem:

* NIC handle output in legacy WHOIS systems SHOULD be replaced with an LDAP URL for the contact entries. This option facilitates faster coalescence around the LDAP-WHOIS system.

* Referral entries MAY be defined for each existing NIC handle if the explicit NIC handle is still required for an application or usage, and queries for NIC handles MAY be

processed through these referral entries. For example, the NIC handle of EH26 on Network Solutions' WHOIS server can be represented as "cn=EH26,cn=inetResources,dc=netsol,dc=com", with the inetOrgPerson and referral object classes defined, and with the ref attribute value pointing to an entry named "cn=Eric A. Hall,cn=inetResources,dc=ntrg,dc=com".

Of the two mechanisms described above, the former is preferred.

10.2. Change-Logs

Several WHOIS services provide pseudo change-logs in their response data, listing each unique modification event which has occurred for a particular resource. For example, RIPE and some of its member ccTLDs provide WHOIS output which includes a series of "changed" fields that itemize every modification event ("updated", "added", etc.), the modifier, and the modification date, which cumulatively act as a change-log for the resource in question.

While this service is useful and informative to the delegating bodies, this information is not as useful to external entities. Furthermore, the principle use of this information is for the purpose of internal audits, rather than external information. Finally, a subset of this kind of information is already provided in the *modified* operational attributes, which are always available for public review.

Organizations are certainly free to maintain this information on their internal systems (and are even encouraged to do so). However, this information is not necessary for public view of the data in the LDAP-WHOIS service. Where the auditing information will be required, a format which is more suitable to legal review will be required and more appropriate.

For these reasons, this service is not supported in the LDAP-WHOIS service. However, if this information is absolutely required, implementers MAY provide it as additional unstructured data via the inetGeneralComments attribute (perhaps using an "event:modifier:date" format).

10.3. Open Issues

The following issues require additional analysis:

* inetIpv6Network and other entries will likely benefit from certificate-related data, although the extent and nature of this information (minimum requirements, preferred attributes, pre-existing schema, etcetera) is currently unknown.

* The RIPE database v3 has several additional attributes:

```
domain:     [mandatory]  [single]     [primary/look-up key]
descr:      [mandatory]  [multiple]   [ ]
admin-c:    [mandatory]  [multiple]   [inverse key]
tech-c:     [mandatory]  [multiple]   [inverse key]
zone-c:     [mandatory]  [multiple]   [inverse key]
nserver:    [optional]   [multiple]   [inverse key]
sub-dom:    [optional]   [multiple]   [inverse key]
dom-net:    [optional]   [multiple]   [ ]
remarks:    [optional]   [multiple]   [ ]
notify:     [optional]   [multiple]   [inverse key]
mnt-by:     [optional]   [multiple]   [inverse key]
mnt-lower:  [optional]   [multiple]   [inverse key]
refer:      [optional]   [single]     [ ]
changed:    [mandatory]  [multiple]   [ ]
source:     [mandatory]  [single]     [ ]
```

see http://www.ripe.net/ripe/docs/databaseref-manual.html

11. 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.

12. 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.

Furthermore, this document defines delegation status codes for four of the resource types described herein, and IANA is expected to maintain the code-point mapping values associated with these attribute values. Each resource type may develop its own peculiar status codes, so each of the mapping tables will need to be maintained independently.

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.

13. Author's Addresses

Eric A. Hall
ehall@ehsco.com

14. References

RFC 1274 - The COSINE and Internet X.500 Schema

RFC 2079 - Definition of an X.500 Attribute Type and an Object Class to Hold Uniform Resource Identifiers (URIs)

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 2253 - Lightweight Directory Access Protocol (v3): UTF-8 String Representation of DNs

RFC 2254 - The String Representation of LDAP Search Filters

RFC 2255 - The LDAP URL Format

RFC 2256 - A Summary of the X.500(96) User Schema for use with LDAPv3

RFC 2308 - Negative Caching of DNS Queries (DNS NCACHE)

RFC 2782 - A DNS RR for specifying the location of services (DNS SRV)

RFC 2798 - Definition of the inetOrgPerson LDAP Object Class

RFC 2849 - The LDAP Data Interchange Format (LDIF) - Technical Specification

[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

[ldap-whois-dns] - <draft-ietf-crisp-lw-dns-00.txt> - Defining and Locating DNS Domains using the Internet Resource Query Service

[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

[ldap-whois-asn] - <draft-ietf-crisp-lw-asn-00.txt> - Defining and Locating Autonomous System Numbers using the Internet Resource Query Service

[ldap-whois-user] - <draft-ietf-crisp-lw-user-00.txt> - Defining and Locating Contact Persons using the Internet Resource Query Service

On a related note, VeriSign has been working on an RLDAP project [described in draft-newton-ldap-whois-00.txt (Whois Domain Data in LDAP)] that uses a query model very similar to the one described in this document, and which illustrates many of the points described in this document. The current RLDAP implementation has three client implementations, multiple distributed servers, and

contains more than 32 million DNS domain entries, and 115 million resource-specific entries. In many regards, this document is an extension of RLDAP.

15. Acknowledgments

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

16. Changes from Previous Versions

draft-ietf-crisp-lw-core-00:

* As a result of the formation of the CRISP working group, the original monolithic document has been broken into multiple documents, with draft-ietf-crisp-lw-core describing the core service, while related documents describe the per-resource schema and access mechanisms.

* References to the ldaps: URL scheme have been removed, since there is no standards-track specification for the ldaps: scheme.

* An acknowledgements section was added.

draft-hall-ldap-whois-01:

* The ôObjectivesö section has been removed. [ir-dir-req] is now being used as the guiding document for this service.

* Several typographical errors have been fixed.

* Some unnecessary text has been removed.

* Figures changed to show complete sets of object classes, to improve inheritance visibility.

* Clarified the handling of reverse-lookup domains (zones within the in-addr.arpa portion of the DNS hierarchy) in the inetDnsDomain object class reference text.

* Referrals now use regular LDAP URLs (multiple responses with explicit hostnames and port numbers). Prior editions

of this specification used LDAP SRV resource records for all referrals.

* The delegation status codes used by the inetDnsDelegationStatus, inetIpv4DelegationStatus, inetIpv6DelegationStatus and inetAsnDelegationStatus attributes have been condensed to a more logical set.

* Added an inetDnsAuthServers attribute for publishing the authoritative DNS servers associated with a domain. NOTE THAT THIS IS A TEMPORARY ATTRIBUTE THAT WILL EVENTUALLY BE REPLACED BY GENERALIZED RESOURCE-RECORD ENTRIES AND ATTRIBUTES.

* Added an inetGeneralDisclaimer attribute for publishing generalized disclaimers.

* Added the inetAssociatedResources auxiliary object class for defining associated resources, and moved some of the IP addressing and ASN attributes to the new object class.

* Several attributes had their OIDs changed. NOTE THAT THIS IS AN INTERNET DRAFT, AND THAT THE OIDS ARE SUBJECT TO ADDITIONAL CHANGES AS THIS DOCUMENT IS EDITED.