INTERNET-DRAFT
Document: draft-ietf-crisp-firs-core-03.txt
Expires: March, 2004
Category: Standards-Track

Eric A. Hall

August 2003

The Federated Internet Registry Service: Core Elements

Status of this Memo

Copyright Notice

Abstract

This document describes the core technical elements of the Federated Internet Registry Service (FIRS), a distributed service for storing, locating and transferring information about Internet resources using LDAPv3.

Table of Contents

1. Introduction

This specification defines the core object classes, attributes, syntax rules, matching filters, and operational behaviors for the FIRS service as a whole. Refer to [FIRS-ARCH] for information on the FIRS architecture, and the resource-specific specifications for definitions and rules which govern each of the different resource-types.

The definitions in this specification are intended to be used with FIRS. Their usage outside of FIRS is not prohibited, but any such usage is beyond this specification's scope of authority.

2. Prerequisites and Terminology

The complete set of specifications in the FIRS collection cumulative define a structured and distributed information service using LDAPv3 [RFC3377] for the data-formatting and transport functions. This specification should be read in the context of that set, which currently includes [FIRS-ARCH], [FIRS-DNS], [FIRS-DNSRR], [FIRS-CONTCT], [FIRS-ASN], [FIRS-IPV4] and [FIRS-IPV6].

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

3. The FIRS Namespace

The FIRS namespace acts as an index to the federated partition structure of the globally-distributed FIRS database. There are three major components to this namespace, which are:

* The domainComponent entries. Each partition of the globally-distributed FIRS database is uniquely represented by a sequence of domainComponent relative distinguished names. These sequences effectively identify the root scope of authority for each partition in the global directory database. Partitions MAY be replicated across one or more servers, but every instance of a specific partition MUST use the same sequence of domainComponent relative distinguished names.

* An inetResources entry. All of the FIRS-related resource-specific entries in the global database are required to be stored within a well-known "cn=inetResources" container

entry at the root of each partition. These well-known entries act as application-specific access points within the globally distributed directory database, and also provide some information about the partition and the organization which manages that partition.

* The resource-specific entries. Each of the resource-specific entries within the inetResources container entries have their own unique naming rules, as defined in the governing specifications for those resources.

Note that an inetResources container or any of the resource-specific entries MAY exist as referral stub entries that redirect clients to other entries in the FIRS database.

The naming rules associated with the different portions of the FIRS namespace are discussed in more detail below.

3.1. The domainComponent (dc=) Namespace Component

The global FIRS directory database is divided into administrative partitions, each of which represent a scope-of-authority for a certain portion of the global database. The root of each partition is represented by a sequence of domainComponent relative distinguished names (RDNs), as defined in RFC 2247 [RFC2247]. In this model, the scope-of-authority for a FIRS partition is derived from a domain name in the global DNS directory, meaning that whoever has authority over any particular domain name effectively has authority over the related FIRS partition.

Note that the domainComponent attribute is restricted to seven-bit character codes, and is therefore effectively limited to using character codes from US-ASCII [US-ASCII]. Due to this limitation, internationalized domain names MUST be converted into their ASCII-compatible forms using the "ToASCII" process defined in RFC 3490 [RFC3490] before the domainComponent RDNs are used in the directory database or LDAP messages.

3.2. The inetResources Namespace Component

The FIRS-specific directory entries are segregated from other application-specific entries by the use of a container entry with the MANDATORY name of "cn=inetResources".

Every FIRS-specific resource that is to be located via the FIRS service MUST be stored within the inetResources container entry.

However, the entries themselves MAY exist as referrals which point to entries in other LDAP partitions or namespace branches if this is necessary or desirable (see section 3.5).

3.3. The Resource-Specific Namespace Component

Every resource-specific entry has a relative distinguished name which identifies that resource within the context of the inetResources container of a FIRS partition. Examples of these entries can be seen in Figure 1 of [FIRS-ARCH], and include "cn=example.com" which refers to the "example.com" DNS domain resource, and "cn=admins@example.com" which refers to the "admins@example.com" contact resource.

Each of the FIRS resource-types have their own specific naming rules which govern those resources. Refer to the resource-specific specifications for information on those rules.

3.4. Attribute References

Many of the core attributes provide pointers to other resources, thereby allowing the client to initiate follow-up queries for related data. For example, an entry for a domain name resource has attributes for a variety of contacts for the domain name, and FIRS clients are able to extract and use that data to generate additional queries for any of those contact resources.

Pointers to resources that are stored within the global FIRS database are generally provided as the identifier for the target resource, using the syntax rules associated with the resource in question. For example, a pointer to the contact entry of "admins@example.com" will be provided as that resource name, using the syntax rules associated with contact names. Examples of this usage form can be seen in Figure 1 of [FIRS-ARCH].

Note that traditional LDAP models often use URIs or distinguished names to provide fully-qualified pointers to entries, although these syntaxes usually require detailed knowledge about the target resources or the servers for the target partition. However, FIRS is much more distributed than traditional LDAP usages, and may require pointing to data in an opaque partition, or where the topology of the target partition is unknown. For these reasons, attribute references in FIRS typically use the short name of the target resource, with the expectation that the client will use the attribute value as the query seed for new FIRS queries.

However, pointers to resource data that is likely to be stored outside the global FIRS database (such as a web page) is generally provided as a URI so that any necessary application and/or protocol transfer services can be specified. For example, the inetResources object class provides an attribute for the organization which manages the resource, with a secondary attribute for a URL associated with the target organization, thus allowing additional (extra-directory) information to be specified and retrieved where this would be useful.

In those cases where a URI provides an LDAP URL that references a resource in the global FIRS directory, the URL data SHOULD use the referral URL rules described in section 3.5.

3.5. Referrals

Entries in the namespace can refer to other entries, as necessary or desirable. Specifically, FIRS allows certain entries to be created as "placeholders" for other entries which contain the canonical data, and also allows "stub" child entries to provide reference pointers to additional data.

LDAP provides several methods for conveying and processing these kinds of referrals, although FIRS only makes specific use of subordinate reference referrals and continuation reference referrals. Subordinate reference referrals indicate that the search base in the original query is an alias for some other entry, and that the query has to be restarted with a new search base in order for the search operation to be processed. Meanwhile, continuation reference referrals indicate that the search was successfully initiated and that some data has been found, but that additional queries for additional resources are required for the query to be completely exhausted.

3.5.1. Referral source entries

Referral entries can be used in FIRS in two distinct scenarios, with each scenario having different naming requirements for the referral sources.

In the first instance, specific entries can be defined as referrals so that queries for the entry always generate a single referral. This may be necessary when an entire inetResources container entry needs to be redirected to another inetResources container entry in another tree, but can also be useful when entries only exist as placeholders for other entries.

For example, a registrar can create referral entries for variant domain names whenever a canonical domain name is registered, with the variant entries only providing referrals to the canonical entry. Similarly, an entry for the host named "www.example.com" could exist as a referral which pointed to the domain name entry for "www-1.example.net". In these scenarios, users who generate queries for the alias domain name entries would always get referred to the canonical domain name entries.

Most entries are expected to provide some kind of information, and in this kind of situation, the canonical entry will have data, but will need to be able to generate referrals to one or more other entries where additional data about the resource can be retrieved. For example, the entries in the partition for the "com" domain registry can provide basic information about a domain, but can also provide a referral to the domain registrar, while the registrar can provide another referral to the domain operator.

In these cases, each partition would need to have an entry for the resource in question, while child entries underneath each of those entries would be used to generate the necessary referrals. For example, the "example.com" domain would likely exist as a canonical entry within a registry's partition, with that entry providing information that the registry had about that domain name. Meanwhile, the registry could have a child entry underneath the canonical entry which provided a referral to the registrar's partition, and so forth.

The relative distinguished name of a referral child entry is usually irrelevant, and can therefore be defined according to local policy rather than fixed rules. However, operators SHOULD NOT use names that are likely to match against searches for other resources. In the general case, using generalized relative distinguished names such as "cn=ref1" or the like would be the safest practice.

3.5.2. Referral target data

Referral entries MUST use the ref attribute and referral object class, as defined in RFC 3296 [RFC3296].

The referral target is provided to the client with the ref attribute, which provides a URI as the destination pointer, although there are some additional FIRS-specific restrictions which are as follows:

* At least one of the URLs in a referral MUST be an LDAP URL as specified in RFC 2255 [RFC2255], and FIRS clients MUST ignore all non-LDAP URIs. Note that general-purpose LDAP referrals are allowed to use any protocol, but FIRS clients have a requirement to automatically process referrals, and this requirement precludes the use of ambiguous services and their data formats. As such, every FIRS referral MUST specify at least one LDAP URL, and FIRS clients MUST only use the LDAP URLs.

* Referral targets MUST use domainComponent ("dc=") naming syntax for target partitions. If a referral needs to specify an exact partition or container for the referral target, the path to the referral target MUST be provided as the search base of the LDAP URLs, and this data MUST be used by FIRS clients when the subsequent query is built.

* The LDAP URL data MUST be escaped prior to being sent. For example, domain names and contact names can contain UTF-8 character data, and some of those character codes will need to be escaped in order to be passed as URL data. Similarly, 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).

In the general sense, referrals SHOULD NOT provide any more information about the referral target than absolutely necessary. For example, if a referral source for a domain name resource needs to reference a referral target for another domain name resource, then only the resource type and identifier SHOULD be provided (this will give the client enough information to begin a new query), while data such as the target partition or LDAP server SHOULD NOT be provided since the authoritative forms of this information will be detected as part of the subsequent query's bootstrapping process. With this in mind, the following recommendations apply to referral targets in the general sense, and SHOULD be followed:

* In those cases where a referral points to a FIRS resource of a known type and name (E.G., a domain name, or an IPv4 address), the referral URL MUST specify the matching filter and assertion value of the referral target. For example, a referral that points to a DNS domain resource MUST provide

the inetDnsDomainMatch matching filter and value in the filter element of the LDAP URL (such as providing "(1.3.6.1.4.1.7161.1.3.0.1:=example.com)" for a referral to the "example.com" DNS domain). Clients MUST use this data to seed the resource type and assertion value of the subsequent query if it is provided.

* In those cases where a referral points to a FIRS resource in a particular partition, the referral URL MUST specify the search base element. For example, if an entry for the "example.com" domain name resource in the "com" partition needs to specify the "example.com" domain name resource in the "registrar.com" partition, then the referral MUST specify "dc=registrar,dc=com" in the search base element of the LDAP URL. Clients MUST use this data to seed the boot partition of the subsequent query if it is provided.

* In those cases where a referral points to some other kind of entry, the referral target SHOULD specify as little information as possible, while still providing an adequate reference. For example, if a referral needs to point to a contact in an alternate container of a specific partition, the full path to the referral target SHOULD be specified in the search base element of the URL. Clients MUST use the additional information if it is provided.

* In the general case, referral sources and targets SHOULD have the same resource-specific object classes defined, although referral targets MAY specify other resource types if needed. For example, the referral source and target for a DNS domain resource should both have the inetDnsDomain object class defined, although a referral may point to an IPv4 host address if this is necessary. If a referral target is known to have a different object class than the referral source, a matching filter for the referral target MUST be provided in the filter element of the LDAP URLs, and this data MUST be used by FIRS clients when the subsequent query is built.

* LDAP URLs SHOULD NOT provide host identifiers or port numbers unless this is absolutely necessary, since the client will usually discover this information during the bootstrap process. If a referral provides this information, it MUST be used by FIRS clients when the subsequent query is built.

* Attribute lists, scope filters, and URL extensions SHOULD NOT be provided, and these elements MUST be ignored by FIRS clients unless an applicable specification details explicit behavior for these elements.

* The operators of a partition MUST NOT restrict referral data to verifiable referral targets. Providers MAY validate the referral targets in URLs, but a lack of knowledge regarding a target MUST NOT be treated as sufficient cause to prevent the referral target from being specified.

* Referral targets MAY themselves be referrals to other entries, but recursive referrals are discouraged. Clients MAY discontinue referral processing after a reasonable amount of effort (eight referrals is a reasonable threshold, but the actual amount of processing is left to the discretion of the clients).

An example referral is illustrated in Figure 1 of [FIRS-ARCH]. In that example, the referral data provides an inetDnsDomainMatch matching filter with the explicit assertion value of "www-1.example.net". This data would inform the client of the resource-type to be queried and the assertion value to use, which collectively would give the client everything needed to begin bootstrapping a new query.

As another example, a referral to a specific entry could look like the following:

```
ldap://cn=admins@example.com,cn=inetResources,
      dc=example,dc=com
```

In that example, the referral is pointing to an explicit entry in an explicit container in an explicit partition. Although the client would not be able to tell what kind of resource was being queried for, it would be able to determine the resource-type once an answer was received, based on the object class values of resulting entry.

The client-side rules for processing referral URLs are given in section 5.4.

Note that the "superior reference referral" defined in RFC 2251 [RFC2251] used as a "default referral" for out-of-scope searches is explicitly unsupported in FIRS. Superior reference referrals

which are encountered as a part of this service are to be treated as errors and silently ignored.

3.5.3. Subordinate reference referrals

Subordinate reference referrals are defined in [RFC3296], and are returned whenever the search base specified in a query exists as a referral to some other entry. This means that the query MUST be restarted with the referral target.

Specifically, subordinate reference referrals are defined in [RFC3296], and use the SearchResultDone response with a Referral result code as defined in [RFC2251]. Subordinate reference referrals use a subset of the labeledURI syntax as defined in [RFC2079], and use the syntax definitions from [RFC2255] when LDAP URLs in particular are provided, although section 3.5.2 of this document also defines additional restrictions on the allowable URL syntax. 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 partition has been redirected to another directory partition.

Almost all of the search functions used with FIRS 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 directory partition.

Servers MUST support the use of subordinate reference referrals, and clients MUST be prepared to accept and process any subordinate reference referrals they receive.

When subordinate reference referrals are used, 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.

3.5.4. Continuation reference referrals

Continuation reference referrals are defined in RFC 2251 [RFC2251], and are returned when a search operation has been successfully processed but the answer data also includes referrals to other entries. These referrals are usually provided as supplemental data to an answer, although it's also possible for a continuation reference referral to be the only data in an answer.

Specifically, continuation reference referrals use the SearchResultReference response, which is defined and described in section 4.5.3 of [RFC2251]. Continuation reference referrals use a subset of the labeledURI syntax as defined in [RFC2079], and use the syntax definitions from [RFC2255] when LDAP URLs in particular are to be provided, although section 3.5.2 of this document also defines additional restrictions on the allowable URL syntax. 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 directory partition, and can occur with any of the search operations described in this document.

Servers MUST support the use of continuation reference referrals, and clients MUST be prepared to accept and process any subordinate reference referrals that they receive.

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.

4. Global FIRS Object Classes and Attributes

Each of the schema definitions provided in this document include attribute definitions, naming rules, and other definitions which are designed to facilitate the consistent storage and retrieval of information within the FIRS service.

4.1. The inetResources Object Class

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

4.1.1. Naming syntax

This document requires the presence of an entry named "cn=inetResources" in the root of every directory partition which provides FIRS services.

4.1.2. Schema definition

Every directory partition which provides public FIRS data MUST have a "cn=inetResources" entry in the root of the directory partition. The inetResources entry MUST exist with the top and inetResources object classes defined. If the entry exists as a referral source, 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 for use with FIRS are subordinate to the inetResources object class and inherit its attributes.

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

```
inetResources
( 1.3.6.1.4.1.7161.1.1.1
  NAME 'inetResources'
  DESC 'The inetResources container for the FIRS service'
  SUP top
  STRUCTURAL
  MUST cn
  MAY ( inetLocalIdentifier $ o $ ou $ description $
  inetResourceComments $ businessCategory $ telephoneNumber $
  facsimileTelephoneNumber $ labeledURI $
  preferredDeliveryMethod $ physicalDeliveryOfficeName $
  postOfficeBox $ postalAddress $ postalCode $ street $ l $
  st $ c $ inetAbuseContacts $ inetGeneralContacts $
  inetSecurityContacts $ inetTechContacts $
  inetGeneralDisclaimer ) )
```

The attributes from the inetResources object class are described below:

businessCategory, see RFC 2256 [RFC2256], section 5.16

c (country), see [RFC2256], section 5.7

cn (commonName), see [RFC2256], section 5.4

description, see [RFC2256], section 5.14

facsimileTelephoneNumber, see [RFC2256], section 5.24

l (locality), see [RFC2256], section 5.8

labeledURI, see RFC 2079 [RFC2079]

o (organization), see [RFC2256], section 5.11

ou (organizational unit), see [RFC2256], section 5.12

physicalDeliveryOfficeName, see [RFC2256], section 5.20

postalAddress, see [RFC2256], section 5.17

postalCode, see [RFC2256], section 5.18

postOfficeBox, see [RFC2256], section 5.19

preferredDeliveryMethod, see [RFC2256], section 5.29

st (stateOrProvinceName), see [RFC2256], section 5.9

street (streetAddress), see [RFC2256], section 5.10

telephoneNumber, see [RFC2256], section 5.21

```
inetLocalIdentifier
( 1.3.6.1.4.1.7161.1.1.2
  NAME 'inetLocalIdentifier'
  DESC 'Provider name for this entry'
  EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{1024} )

inetResourceComments
( 1.3.6.1.4.1.7161.1.1.3
  NAME 'inetResourceComments'
  DESC 'General comments about this entry'
  EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{1024} )

inetGeneralDisclaimer
( 1.3.6.1.4.1.7161.1.1.4
  NAME 'inetGeneralDisclaimer'
  DESC 'General disclaimer text regarding this data'
  EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{1024} )

inetGeneralContacts
( 1.3.6.1.4.1.7161.1.1.5
  NAME 'inetGeneralContacts'
  DESC 'Contacts for general administrative issues.'
  EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.7161.1.7.1 )
```

```
inetAbuseContacts
( 1.3.6.1.4.1.7161.1.1.6
  NAME 'inetAbuseContacts'
  DESC 'Contacts for reporting abusive behavior or
  acceptable-use policy violations.'
  EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.7161.1.7.1 )

inetSecurityContacts
( 1.3.6.1.4.1.7161.1.1.7
  NAME 'inetSecurityContacts'
  DESC 'Contacts for general security issues.'
  EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.7161.1.7.1 )

inetTechContacts
( 1.3.6.1.4.1.7161.1.1.8
  NAME 'inetTechContacts'
  DESC 'Contacts for general technical issues.'
  EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.7161.1.7.1 )
```

4.1.3. Example

An example of the inetResources object class in use is shown in Figure 1 below.

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

Figure 1: The Example Widgets inetResources container entry.

4.2. The inetAssociatedResources Object Class

The inetAssociatedResources object class defines attributes which are useful for cross-referencing entries with other resources. For example, it allows inetOrgPerson entries to be associated with IPv4 networks or DNS domains, providing generic cross-reference pointer attributes (this information may be useful if a single organization has multiple DNS domains registered). In short, any resource can be associated with any other resource through the use of this object class.

4.2.1. 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 defined under the rules associated with the structural object class that defines the Internet resource in question. As such, the naming rules associated with that entry take precedence, and the inetAssociatedResources object class does not define an independent naming syntax.

4.2.2. 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, and only has optional attributes.

The inetAssociatedResources is intended to be used with the resource-specific structural object classes defined for use with FIRS. 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.

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

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 multiple associated DNS domain name resources using a multi-valued array, with each domain name using the inetDnsDomainSyntax naming rules defined in [FIRS-DNS].

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

```
inetAssociatedResources
( 1.3.6.1.4.1.7161.1.2.1
  NAME 'inetAssociatedResources'
  DESC 'Internet resources associated with this resource.'
  SUP top
  AUXILIARY
  MAY ( inetAssociatedContacts $ inetAssociatedDnsDomains $
   inetAssociatedIpv4Networks $ inetAssociatedIpv6Networks $
   inetAssociatedAsNumbers ) )
```

The attributes from the inetAssociatedResources object class are described below:

```
inetAssociatedAsNumbers
( 1.3.6.1.4.1.7161.1.2.2
  NAME 'inetAssociatedAsNumbers'
  DESC 'Autonomous system numbers associated with this
  Internet resource.'
  EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.7161.1.7.0 )

inetAssociatedContacts
( 1.3.6.1.4.1.7161.1.2.3
  NAME 'inetAssociatedContacts'
  DESC 'Other contacts associated with this Internet
  resource.'
  EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.7161.1.4.0 )

inetAssociatedDnsDomains
( 1.3.6.1.4.1.7161.1.2.4
  NAME 'inetAssociatedDnsDomains'
  DESC 'DNS domains associated with this Internet resource.'
  EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.7161.1.3.0 )
```

```
inetAssociatedIpv4Networks
( 1.3.6.1.4.1.7161.1.2.5
  NAME 'inetAssociatedIpv4Networks'
  DESC 'IPv4 networks associated with this Internet
  resource.'
  EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.7161.1.5.0 )

inetAssociatedIpv6Networks
( 1.3.6.1.4.1.7161.1.2.6
  NAME 'inetAssociatedIpv6Networks'
  DESC 'IPv6 networks associated with this entry.'
  EQUALITY caseIgnoreMatch
  SYNTAX 1.3.6.1.4.1.7161.1.6.0 )
```

4.2.3. Example

An example of the inetAssociatedResources object class is shown in Figure 2 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 Widgets network"
|
+-attribute: inetAssociatedAsNumbers
| value: "65535"
|
+-attribute: inetAssociatedDnsDomains
  value: "2.0.192.in-addr.arpa"
```

Figure 2: The inetAssociatedResources attributes associated with the 192.0.2.0/24 IPv4 network entry.

4.3. The referral Object Class

Entries use the referral object class to define subordinate reference referrals and continuation reference referrals, thereby facilitating the programmatic redirection of queries in support of the referral mechanisms defined in section 3.5.

Referral entries MUST conform to the schema specification defined in [RFC3296].

Referral sources MUST NOT contain any user-definable attributes (other than the mandatory naming attribute), and MUST NOT have any subordinate child entries.

Refer to section 3.5 for the rules that govern referral URLs in FIRS. Refer to section 5.4 for information on processing referral URLs in FIRS.

5. Global Query Processing Rules

Another critical aspect to FIRS is the query-processing behavior. These rules govern the ways in which a client parses a query, locates a server which is authoritative for the resource being queried, generates LDAPv3 queries, and processes any resulting referrals. More specifically:

* Query pre-processing. The first step is for the client to prepare the query. Portions of this process require the client to determine the type of resource being queried for, and to determine the initial partition which should be used for the query. Since this process is different for each particular resource-type, the rules which govern this behavior are defined in each of the resource-specific specifications.

* Bootstrap processing. Once a partition has been determined, the client must locate the LDAP servers which are authoritative for the resource in question. Section 5.2 defines three different bootstrap models that clients can use as part of this process, while each of the resource-specific specifications define which of the models are to be used for each particular resource-type.

* Query processing. Once a server has been located, the client must submit the LDAP query which was formed during the pre-preprocessing phase. Section 5.3 defines the global considerations for all FIRS queries, while the resource-specific specifications also define additional parameters.

* Query post-processing. FIRS explicitly supports different types of LDAP referral mechanisms, any of which may result in the client application restarting the query or

initiating a brand-new query. These mechanisms and their behavioral rules are defined in section 5.4.

Each of these phases are discussed in more detail below.

5.1. Query Pre-Processing

Client input is generally limited to a single well-formed unit of data, such as a domain name ("example.com") or an email address ("admins@example.com"), and this single piece of information must be used to subsequently build a fully-formed LDAPv3 query, including the assertion value, the search base, the matching filter, and so forth. All of these steps are part of the pre-processing phase.

Although the exact sequence of steps will vary according to the resource-type being queried, there are some commonalities between each of them. Among these steps:

* Determine the resource type. Different kinds of resources have different processing steps, validation mechanisms, and so forth, each of which require that the resource-type be appropriately identified. Clients MAY use any mechanisms necessary to force this determination.

* Validate and normalize the data. In all cases, the input data MUST be validated and normalized according to the syntax rules defined in the specification which governs the resource-type. As an example of this step, queries for internationalized domain names must be validated and normalized into a canonical UTF-8 form before any other steps can be taken. Similarly, IPv6 addresses are required to conform to specific syntax rules, and input address may need to be expanded or compressed in order to comply with the syntax requirements.

* Determine the authoritative directory partition for the named resource. In most cases, the authoritative partition will be a variation of the input query string, but this is not always the case. For example, the default partition for an email address will be extrapolated from the domain component of the email address itself, while the authoritative partition for an ASN uses a reserved (special-purpose) domain name. In some cases, the authoritative partition may change during the subsequent query-processing steps.

* Determine the search base for the query. Each resource type has resource-specific query-processing rules which will dictate how the authoritative partitions are mapped to the search base. In some cases, the cn=inetResources container entry in the authoritative partition will be used "as-is", while in other cases, the cn=inetResources container entry in a delegation parent of the authoritative partition will be used instead. In some cases, the search base may change during subsequent query-processing steps.

* Determine the assertion value for the query. The assertion value will usually be the normalized form of the input query. In some cases, the assertion value may change during subsequent query-processing steps.

* Determine the matching filter. Each resource-type has its own matching filter rules. For example, contact entries are matched with a simple equalityMatch comparison, while in other cases the matching filter will be an extensibleMatch which is peculiar to the resource-type in use.

Once all of the pre-processing steps have been successfully completed, the client will have to locate an LDAPv3 server which is authoritative for the search base before it can submit the query. This process is described in section 5.2 below.

5.2. Query Bootstrap Models

Once a client has determined which partition should be queried for the specified resource, the client will need to determine which LDAP servers are authoritative for that partition.

The FIRS service supports three different bootstrap models for this process, although these models only differ in relatively minor ways; once a server has been located, the rest of the query process follows the same basic path (issuing the LDAPv3 query, following referrals, and so forth).

The three bootstrapping models defined for use with this service are the "targeted" model which is functionally identical to traditional lookups for LDAP servers, the "top-down" model which causes a client to submit a query to the root of a delegation hierarchy, and a "bottom-up" model which causes a client to work up through a delegation hierarchy until a server has been located.

5.2.1. Targeted query processing

The "targeted" model is similar to the traditional model of LDAP lookups, in that a client queries a specified LDAP server for a particular resource under the assumption that the named resource exists on the named server. If the known resource or the known server do not exist or cannot be located (notwithstanding any referrals which may be returned), then the query process exits.

The targeted model can be used when an application-specific partition or resource has been specified, but can also be used if the client prefers to use a "default" server for all operations. The latter may occur when clients use proxy servers, caching servers, or other fixed servers, in lieu of navigating the global directory database with every query.

The targeted model is primarily suited for locating Internet resources which are managed and delegated by a central body, but which is not necessarily located in a directory partition under a top-level domain. For example, AS numbers, IPv4 address blocks, and IPv6 address blocks are all managed under specific partitions which are not directly linked to a specific top-level domain, so those queries have to be started at specific partitions, and would not be efficiently served by partitions higher or lower in the delegation hierarchy.

The steps for processing targeted queries are described below:

a. Determine the IP address and port number to be used (this information may be determined from user input, a configuration file, a URL, or from any other source).

    1. If a non-ASCII domain name has been specified for this purpose, convert the domain name into its ASCII-compatible form using the "ToASCII" process defined in [RFC3490] before performing any lookups.

    2. Locate the LDAP servers associated with the domain name through the SRV query steps provided in section 5.2.4. If this step fails, use DNS lookups for A resource records instead. If no resource records are available, report the error to the user and exit.

b. Once a server has been determined, submit the search operation. If the search operation fails, report the error

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

c. 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 5.4 and restart the query process at step 5.2.1.a. For each continuation reference referral, display the answer data received so far, process the LDAP URLs according to the rules described in section 5.4 and start new query processes for each referral at step 5.2.1.a, 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.

d. Exit the query operation.

5.2.2. Top-down processing

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, a query is issued to the specified servers. This query may be subsequently redirected to other servers through the use of LDAP referrals.

The top-down model is primarily suited for locating Internet resources which are centrally managed and delegated, and where information about the delegation is available from a delegation body with a top-level domain. The best example of this is resources under the top-level domains themselves, such as queries for domain delegations under the "com" zone.

Note that the top-down model does not use incrementally larger domain names for the bootstrap process. Instead, it is assumed that the root partition in the delegation tree will be able to provide any necessary redirection services. For example, if the domain name of "www.example.co.uk" is used in a query, the query will be sent to the "dc=uk" partition, which should provide a

referral for the "dc=co,dc=uk" partition, which in turn should provide a referral for the "dc=example,dc=co,dc=uk" partition.

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

a. Determine the directory partition 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.

b. Form the LDAP search base for the query.

   1. If the client application allows non-ASCII input, convert the domain name formed in step 5.2.2.a above into its ASCII-compatible form using the "ToASCII" process defined in RFC 3490.

   2. Convert the right-most element from the domain name formed in step 5.2.2.b.1 into a domainComponent DN (such as "dc=com" or "dc=arpa"). This represents the directory partition for the current query.

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

c. Locate the LDAP servers associated with the resource by processing the domain name formed in step 5.2.2.a above through the SRV query steps provided in section 5.2.4.

d. If the SRV lookup succeeds:

1. Choose the best LDAP server, using the weighting formula described in RFC 2782 [RFC2782].

2. Formulate the LDAP search using the search base and search filter constructed earlier. 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".

3. 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.

4. 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 5.4 and restart the query process at step 5.2.2.b. For each continuation reference referral, display the answer data received so far, process the LDAP URLs according to the rules described in section 5.4 and start new query processes for each referral at step 5.2.2.b, 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.

e. 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.

f. Exit the query operation.

5.2.3. Bottom-up processing

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. If no servers are available for that partition, the parent partition in the delegation hierarchy is used instead, with this process repeating until a server has been located.

The bottom-up model is best used when a leaf-node partition needs to be queried directly, either because there is no direct delegation path for the resource in question, or because the user-managed partition is preferable to the centralized delegation information. For example, there is no global delegation body which assigns and manages contact identifiers, so these identifiers need to be directed towards the leaf-node partitions directly. The bottom-up model can also be used for other kinds of resources if desirable, although in most cases the bottom-down model will be more useful for those resources.

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. If the client application allows non-ASCII input, convert the domain name formed in step 5.2.3.b above into its ASCII-compatible form using the "ToASCII" process defined in RFC 3490.

2. Convert the domain name formed in step 5.2.3.c.1 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 directory partition for the current query.

3. 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 5.2.3.b above through the SRV query steps provided in section 5.2.4.

e. If the SRV lookup fails with an NXDOMAIN response code (as described in RFC 2308 [RFC2308]), then the domain name used for the SRV lookup does not exist, and a substitute LDAP server and search base must be used instead. 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 5.2.3.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 5.2.2.b, 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 [RFC2782].

2. 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". If the SRV lookups had failed (resulting in "com" being used as the authoritative directory partition), then the search base for the query would also be trimmed accordingly ("cn=inetResources,dc=com").

3. 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.

4. 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 5.4 and

restart the query process at step 5.2.3.d. For each continuation reference referral, display the answer data received so far, process the LDAP URLs according to the rules described in section 5.4 and start new query processes for each referral at step 5.2.3.d, 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).

h. Exit the query operation.

5.2.4. SRV processing

The bootstrapping models described in this document make use of DNS SRV resource records to locate the LDAP servers associated with the resource provided in the query input.

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. If the client allows non-ASCII characters to be input, then convert the domain name input into its ASCII-compatible form by using the "ToASCII" process described in [RFC3490].

c. Append the SRV label pair to the left of the input domain name from step 5.2.4.b. In the case of a query for the "example.com" domain, this would result in an SRV-specific domain name of "_ldap._tcp.example.com".

d. Issue a DNS query for the SRV resource records associated with the domain name formed in step 5.2.4.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 [RFC2782] for a detailed explanation of SRV resource records and their handling.

If a preferred connection target is listed with multiple IP addresses, clients should cycle through the IP addresses before using the next-preferred connection target.

5.3. Query Processing

Once an authoritative server for the partition in question has been located, the LDAP query can be submitted. In order to ensure interoperability, this specification defines several behavioral rules which clients and servers SHOULD conform with. These guidelines are discussed in the following sections.

5.3.1. The inetResourcesControl server control

The inetResourcesControl server control is the master control object for the FIRS service, and provides the version of each object class that is available for use on the current server, and also lists the matching filters that the server is willing to use for each of the listed object classes.

The OID for inetResourcesControl is 1.3.6.1.4.1.7161.1.0.0. This value MUST be provided in the OID field of the control.

The value section of the inetResourcesControl contains nested sequences of data. The first element in each sequence identifies an object class, while first nested element identifies a matching filter which may be used for that object class, while the next set of nested elements identify the attributes that may be used with each matching filter.

The structure of the value section of inetResourcesControl is illustrated by the following ASN.1 definition:

```
inetResourcesControlValue ::= SEQUENCE {
  objectClass           LDAPOID,
  matchingFilters       SEQUENCE OF matchingFilter,
  attributes            SEQUENCE OF attribute }

matchingFilter ::= LDAPSTRING

attribute ::= LDAPOID
```

Each object class, matching filter, and attribute MUST be presented as a string. Object classes MUST be listed as OIDs so that clients can determine the supported object classes according to their version. Matching filters MUST also be provided as OIDs, except for the "stock" matching filter operators defined in [RFC2251], which MUST be presented with the textual identifiers shown therein. Attributes MUST be listed with their textual identifiers.

At a minimum, servers MUST support the equalityMatch and extensibleMatch filters from [RFC2251] for every object class listed, and SHOULD always declare these filters. Furthermore, servers MUST support the "cn" attribute for every matching filter, and SHOULD declare these attributes. The Asterisk character ("*") MAY be provided as a wildcard to indicate that the server will accept any matching filter for the associated object class, or to indicate that the server will accept any attribute for the associated matching filter. Servers MUST allow any supported matching filter to be used as part of an extensibleMatch operation, and clients MAY assume that any allowed operation will be acceptable as part of an extensibleMatch.

An example of an inetResourcesControl server control is shown below for illustration purposes:

```
{ 1.3.6.1.4.1.7161.1.0.0, FALSE, {
        1.3.6.1.4.1.7161.1.1.1 {
                equalityMatch {*},
                extensibleMatch {*} } }, {
        1.3.6.1.4.1.7161.1.3.1 {
                1.3.6.1.4.1.7161.1.3.0.1 {*},
                equalityMatch {*},
                substringMatch {cn},
                extensibleMatch {*} } } }
```

Figure 3: An example inetResourcesControl server control.

In the example shown in Figure 3, the inetResourcesControl type is identified by the OID of 1.3.6.1.4.1.7161.1.0.0, while the criticality field is set to FALSE, as per the requirements in [RFC2251]. The contents of the control value identify the current OID for the inetResources object class along with the [RFC2251] textual identifiers of the equalityMatch and extensibleMatch operators, each of which will accept any attributes. Figure 3 also identifies the OID for the inetDnsDomain object class along with the OID for the inetDnsDomainMatch and the [RFC2251] textual identifiers for the equalityMatch, substringMatch and extensibleMatch operators, although the substringMatch filter is only advertised for use with the "cn" attribute.

FIRS-compliant servers SHOULD return the inetResourcesControl server control as an unsolicited response to a successful bind request. Clients MUST use the OID of the inetResourcesControl for the purpose of validating the contents of the control, and MUST use the OIDs of the listed object classes to discover schema versioning information.

Servers MAY restrict the contents of the inetResourcesControl value according to the authenticated identity of the client. For example, servers can choose to enable computationally-intense searches for authorized users while refusing to provide the same searches for anonymous users.

If a client does not receive a usable inetResourcesControl control as part of the bind response, the client SHOULD issue a request for the control before proceeding. If a client is still unable to obtain a usable inetResourcesControl server control, the client

MAY choose a different server for the partition, or MAY choose to assume that the equalityMatch matching filter will be supported for any of the known types, or MAY choose to undergo any other recovery efforts. In any event, clients SHOULD NOT use the absence or contents of the control to completely abort query processing unless all of the servers for a partition have refused to provide service to the client.

For example, if the server advertises support for the inetDnsDomain object class but does not advertise support for the inetDnsDomainMatch matching filter, the client MAY issue discrete equality searches for each of the specific domain name resources (note that this kind of query can fail to produce referrals in some cases, but will usually produce at least some answers).

In all cases, if any given server advertises support for a particular object class or matching filter, the client MUST make use of the server-provided service.

5.3.2. Matching 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 ensure interoperability, default values are defined here for many of these elements.

[RFC2251] 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 FIRS clients and servers, this document defines some recommended and mandatory values for searches within the FIRS service.

> NOTE: These rules ONLY apply to the FIRS search operations in particular. Any queries for other resources SHOULD NOT impose these restrictions. Also note that other documents which define additional resource types can also define different restrictions, and those definitions will take precedence over the global defaults.

Servers MUST be prepared to enforce these rules independently of the client settings, and clients MUST be prepared to receive truncated search results accordingly.

The default values of an LDAPv3 search filter in FIRS are:

* Search base. The directory partition 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 5.1, although FIRS searches are normally constrained to the "cn=inetResources" container of a particular directory partition.

* Scope. In order to successfully locate referral stub child entries, clients MUST use a sub-tree scope for FIRS searches. Servers MUST NOT arbitrarily limit the scope of search operations.

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

* Size limit. The size limit field specifies the maximum number of entries that a server should return. For the FIRS 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 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 FIRS 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 the time limit 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 FIRS queries, the filter MUST use the matching rules defined for the relevant resource type.

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

5.3.3. Query-volume restrictions

The restrictions listed in section 5.3.2 represent suggested defaults, although server operators MAY impose any kinds of usage limits they deem necessary or desirable.

Specifically, server operators MAY restrict the amount of information provided to specific clients and/or users over a given amount of time, within reason. For example, servers MAY restrict clients to an arbitrary number of queries per-hour or per-day, or may impose mandatory time intervals between queries, and so forth. Similarly, servers MAY restrict clients to an arbitrary number of answers over a given time period, such as limiting clients to 100 answers regardless of the number of queries which were used to generate those answers.

Servers which refuse to process a query due to volume policy SHOULD use the "unwillingToPerform" response code ("53") to inform the client of these restrictions, and SHOULD provide explanatory text in the error message. These errors SHOULD be generated when the session is first established, if at all possible.

In the worst cases, servers MAY deny all service to abusive clients. This can be implemented by rejecting the TCP connection outright, or by providing an explanatory error early in the session, or at any other point.

Clients MUST be prepared for connection requests and queries to be denied for any reason, and MUST treat these conditions as non-permanent failures. Clients MAY retry the operations if a known error condition is corrected (such as authentication errors), but SHOULD NOT automatically generate retry attempts.

5.3.4. Authentication restrictions

Servers operators SHOULD allow anonymous authentication for read-only access to public delegation data. Clients SHOULD use anonymous authentication by default.

Wherever a server operator requires or desires clients to authenticate for access, servers MUST support the simple authentication mechanism defined in RFC 2222 [RFC2222], although server operators MAY require the use of any authentication mechanisms in addition to or instead of the simple mechanism.

Server operators MAY define any access controls on the data, as necessary for policy considerations. Server operators SHOULD NOT impose unreasonable requirements for proprietary authentication mechanisms for routine purposes.

Server operators MAY withhold privileged information from non-privileged clients or users, as necessary.

Clients MUST NOT equate the absence of any attributes with the absence of data, and SHOULD assume that the authenticated user is not authorized to view any data which has not been provided.

If a client specifically requests an entry or an attribute which the server is unwilling to provide due to ACL settings, the server MUST use the appropriate LDAPv3 error message. For example, if the user is unable to view an entry or a requested attribute because it has not yet provided sufficient authentication credentials, the server MUST return the "invalidCredentials" error. Similarly, if the client has requested an entry or attribute which the server is unwilling to provide due to policy reasons, the server MUST return the unwillingToPerform error to the client.

See section 5.3.5 for mechanisms that can be used to determine and/or describe usage restrictions on specific attribute values.

5.3.5. Extended attribute ACLs

In normal operations, attributes and values that the client is not authorized to view would not be returned in response to queries for that data, with the client equating a lack of data in a particular attribute with "no data that you are authorized to view". However, [CRISP-REQ] defines additional response types for conveying explicit restrictions on data (such as reporting that

the data is restricted due to privacy considerations), and which requires more comprehensive reporting than simply omitting data that the user is not authorized to view.

This extended data is provided through the use of LDAP attribute options, as described in section 4.1.5 of [RFC2251] (this is the same mechanism as used with language tags), using "inetExtAcl-" as the option prefix.

The current-valid list of attribute options are:

* InetExtAcl-S0 -- Security Restricted. Some attribute values have not been returned due to security requirements which have not been met.

* InetExtAcl-P0 -- Privacy Restricted. Some attribute values have not been returned due to privacy requirements which have not been met.

* InetExtAcl-S1 -- Security Cleared. The security requirements on the associated attribute values have been met and the user has been granted access to the data.

* InetExtAcl-P1 -- Privacy Cleared. The privacy requirements on the associated attribute values have been met and the user has been granted access to the data.

* InetExtAcl-R0 -- Do Not Distribute. The associated attribute values are not to be reused outside this session.

Each attribute instance MUST have one of the above attribute options, but MUST NOT have more than one option. Multiple instances of the attribute option MAY be assigned to an attribute, although the instances SHOULD NOT have conflicting meanings.

As a simple example, "mail;inetExtAcl-S1:admins@example.com" would indicate that this instance of the "mail" attribute had an ACL protecting it from normal use, but that the user was authorized to view the attribute data. Meanwhile, the attribute instance of "mail;inetExtAcl-S1;inetExt-R0:admins@example.com" would indicate that the user had been granted access to the data, but that the user must not distribute the data.

Multiple and varying instances of an attribute can co-exist in an entry simultaneously if necessary. For example, the following entries can all co-exist within a single entry.

```
mail:admins@example.com
mail;inetExtAcl-S1:postmaster@example.com
mail;inetExtAcl-P1:jack@example.com
mail;inetExtAcl-P1:inetExtAcl-R0:jill@example.com
```

In that example, searches for the "mail" attribute would produce the public information, while searches for the other attribute instances would produce the alternate data.

Each instance of an attribute can have its own ACLs in the directory, thereby allowing specific instances of an attribute to be restricted to certain users. For example, ACLs on the inetExtAcl-S1 instance of an attribute can be defined so that the entry owner can view the data, while the inetExtAcl-S0 instance can be set such that help-desk operators are able to see that there are hidden attribute values (but without exposing the values to those users), but with both instances being hidden from anonymous users so that the general public does not know that there are any extended attribute options.

Each instance of the attributes can be requested through normal query processes, although the inetExtAcl-S0 and inetExtAcl-P0 attributes will always be empty (the presence of the attribute option implies that the related data could not be returned), and thus those instances will never be returned.

In order to simplify these requests and responses, an inetExtAclControl client control is provided that specifically allows for the request of all extended ACL attribute options. The OID for the inetExtAclControl control is 1.3.6.1.4.1.7161.1.0.1. Clients MUST provide this control as part of the search request, and servers which support this control MUST return all of the regular and extended ACL attributes that are defined for an entry (according to the ACLs appropriate for the current user).

5.3.6. Protocol and schema version controls

The FIRS collection of specifications are explicitly bound to the LDAPv3 protocol, as defined by [RFC3377] and its subordinate specifications. If a new version of the LDAP protocol emerges, it is expected that some type of mechanism will be included for end-

points to use when negotiating over the version in use. Lacking such a mechanism, FIRS is explicitly restricted to LDAPv3.

LDAP attributes, object classes, syntaxes and matching filters have OIDs which uniquely identify the format of the data they provide, and which act as simple schema-version identifiers in the generic case. [RFC2251] defines standardized mechanisms for retrieving and reading the OIDs associated with object classes and attributes (among other resource types). These mechanisms MAY be used whenever a FIRS client reads an entry, and MUST be used whenever a FIRS client modifies or creates an entry (even though FIRS does not define mechanisms for updating entries, it is assumed that some clients will allow this behavior).

The inetResourcesControl server control described in section 5.3.1 provides a mechanism that clients can use to determine the version of an object class or matching filter that the server supports.

Any modifications to any existing schema definitions MUST be accompanied by new OID assignments for the affected elements.

5.4. Referral Processing

As was discussed in section 3.5, FIRS supports two types of LDAP referrals, which are subordinate reference referrals and continuation reference referrals. Both referral types use URLs for the purpose of providing referral targets, using the rules described in section 3.5 of this document.

Non-compliance with the URL formatting requirements provided in section 3.5.2 amounts to an error, and is sufficient cause for a client to stop processing a query.

The procedure for processing referral URLs is as follows:

a. [RFC2251] 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. Locate the LDAP URLs in the referral data, and discard any URLs which use any other service types. FIRS clients MUST

support LDAP URLs. URLs with other service type identifiers SHOULD be ignored.

c. Extract any port number which may have been provided with the URL, and set it aside for possible use with the subsequent connection attempt. Use the port number discovered through any subsequent SRV lookups (as described below), or as a last resort use the default port number associated with the protocol identifier.

d. Determine the authoritative partition and search base specified in the referral URL.

   1. If no distinguished name element was provided, determine the authoritative partition and search base from the provided assertion value, according to the procedures for the bootstrap model that is most relevant to the resource-type.

   2. Otherwise, use the distinguished name element for the search base of the subsequent search operation.

   3. Extract the sequence of domainComponent distinguished names from the search base, and use them as the authoritative partition.

e. Determine the server address and port number specified in the referral URL.

   1. If a host identifier was not provided, map the domainComponent elements determined in step 5.4.d to a DNS domain name and submit a DNS lookup for the SRV resource records associated with that domain name. If this step fails, report the error to the user and exit the query.

   2. If the host identifier is an IP address, extract it and skip to step 5.4.f.

   3. If no port number was provided in the URL, submit a DNS lookup for the SRV resource records associated with the domain name, as described in section 5.2.4. If this lookup succeeds, skip to step 5.4.f.

4. If the SRV lookup from the previous step fails, or if no port number was specified, submit a DNS lookup for the A resource records.

f. Determine the new assertion value and/or matching filter specified in the referral URL.

1. If the URL's path element does not contain a filter element, reuse the current matching filter and assertion value.

2. If the URL's path element contains a filter element, use it to form the new matching filter and/or assertion value.

g. Discard the remainder of the URL.

h. Use the discovered parameter values to start a new query.

Note that step 5.4.g requires the client to discard the remainder of the URL, although this step may be changed in subsequent versions of this specification. In particular, [CRISP-REQ] requires the ability to pass an inter-server "referral bag", and this mechanism may be implemented through the use of extensions in the LDAP URL.

6. Security Considerations

Security considerations are discussed in [FIRS-ARCH].

7. IANA Considerations

IANA considerations are discussed in [FIRS-ARCH].

8. Normative References

[RFC1274] Barker, P., and Kille, S. "The COSINE and Internet X.500 Schema", RFC 1274, November 1991.

[RFC2079] Smith, M. "Definition of an X.500 Attribute Type and an Object Class to Hold Uniform Resource Identifiers (URIs)", RFC 2079, January 1997.

[RFC2222] Myers, J. "Simple Authentication and Security Layer (SASL)", RFC 2222, October 1997.

[RFC2247] Kille, S., Wahl, M., Grimstad, A., Huber, R., and Sataluri, S. "Using Domains in LDAP/X.500 DNs", RFC 2247, January 1998.

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

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

[RFC2253] Wahl, M., Kille, S., and Howes, T. "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of DNs", RFC 2253, December 1997.

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

[RFC2255] Howes, T., and Smith, M. "The LDAP URL Format", RFC 2255, December 1997.

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

[RFC2277] Alvestrand, H. "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

[RFC2308] Andrews, M. "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, March 1998.

[RFC2596] Wahl, M., and Howes, T. "Use of Language Codes in LDAP", RFC 2596, May 1999.

[RFC2782] Gulbrandsen, A., Vixie, P., and Esibov, L. "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC2798] Smith, M. "Definition of the inetOrgPerson LDAP Object Class", RFC 2798, April 2000.

[RFC3296] Zeilenga, K. "Named Subordinate References in Lightweight Directory Access Protocol (LDAP) Directories", RFC 3296, July 2002.

[RFC3377] Hodges, J., and Morgan, R. "Lightweight Directory Access Protocol (v3): Technical Specification", RFC 3377, September 2002.

[RFC3490] Falstrom, P., Hoffman, P., and Costello, A. "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

[FIRS-ARCH] Hall, E. "The Federated Internet Registry Service: Architecture and Implementation Guide", draft-ietf-crisp-firs-arch-03, August 2003.

[FIRS-ASN] Hall, E. "Defining and Locating Autonomous System Numbers in the Federated Internet Registry Service", draft-ietf-crisp-firs-asn-03, August 2003.

[FIRS-CONTCT] Hall, E. "Defining and Locating Contact Persons in the Federated Internet Registry Service", draft-ietf-crisp-firs-contact-03, August 2003.

[FIRS-DNS] Hall, E. "Defining and Locating DNS Domains in the Federated Internet Registry Service", draft-ietf-crisp-firs-dns-03, August 2003.

[FIRS-DNSRR] Hall, E. "Defining and Locating DNS Resource Records in the Federated Internet Registry Service", draft-ietf-crisp-firs-dnsrr-02, July 2003.

[FIRS-IPV4] Hall, E. "Defining and Locating IPv4 Address Blocks in the Federated Internet Registry Service", draft-ietf-crisp-firs-ipv4-03, August 2003.

[FIRS-IPV6] Hall, E. "Defining and Locating IPv6 Address Blocks in the Federated Internet Registry Service", draft-ietf-crisp-firs-ipv6-03, August 2003.

[US-ASCII] Cerf, V. "ASCII format for Network Interchange", RFC 20, October 1969.

9. Changes from Previous Versions

draft-ietf-crisp-firs-core-03:

* Several clarifications and corrections have been made.

* Added a discussion on attribute references.

* Added a discussion on referral source entry names.

* Clarified the rules for LDAP referral URLs.

* Temporarily removed the examples for referral processing, pending additional clarification text.

* Renamed the firsVersion control to inetResourcesControl and redefined its usage slightly.

* Renamed inetPrivateIdentifier to inetLocalIdentifier

* Added the inetExtAcl attribute option family, and defined the inetExtAcl client control.

draft-ietf-crisp-firs-core-02:

* Several clarifications and corrections have been made.

* Changed the referral requirements so that servers are allowed to provide non-LDAP URLs but that FIRS clients are required to ignore non-LDAP URLs. This synchronizes referral mechanisms in the back-end data-stores, and moves the narrower requirement to the client.

* Added an inetPrivateIdentifier attribute for storing operator-specific labels (E.G., legacy NIC handles).

* Added the firsVersion server control, which provides a limited amount of version- and feature-negotiation support to FIRS.

* 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.

draft-ietf-crisp-firs-core-01:

* Several clarifications and corrections have been made.

* Significant portions of text were moved to [FIRS-ARCH].

draft-ietf-crisp-firs-core-00:

* Restructured document set, separating the architectural discussion from the technical descriptions. Several sections were relocated to [FIRS-ARCH] as a result of this change.

* "Attribute references" have been eliminated from the specification. All referential attributes now provide actual data instead of URL pointers to data. Clients that wish to retrieve these values will need to start new queries using the data values instead of URLs.

* The various modified* operational attributes in the core schema have been eliminated as unnecessary.

* 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.

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 "Objectives" 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.

10. Author's Address

Eric A. Hall
ehall@ehsco.com

11. Acknowledgments

Funding for the RFC editor function is currently provided by the Internet Society.

Portions of this document were funded by VeriSign Labs.

The first version of this specification was co-authored by Andrew Newton of VeriSign Labs, and subsequent versions continue to be

developed with his active participation. Edward Lewis and Peter Gietz also contributed significant feedback to this specification in the later stages of its developments.

12. Full Copyright Statement