                  TISDAG - Technical Infrastructure for
                    Swedish Directory Access Gateways

Status of this Memo

Copyright Notice

Abstract

   The strength of the TISDAG (Technical Infrastructure for Swedish
   Directory Access Gateways) project's DAG proposal is that it defines
   the necessary technical infrastructure to provide a single-access-
   point service for information on Swedish Internet users.  The
   resulting service will provide uniform access for all information --
   the same level of access to information (7x24 service), and the same
   information made available, irrespective of the service provider
   responsible for maintaining that information, their directory service
   protocols, or the end-user's client access protocol.

Table of Contents

List of Tables

1.0 Introduction

1.1 Project Goal

   The overarching goal of this project is to develop the necessary
   technical infrastructure to provide a single-access-point service for
   searching for whitepages information on Swedish Internet users.  The
   service must be uniform for all information -- the same level of
   access to information (7x24 service), and the same whitepages
   information made available, irrespective of the service provider
   responsible for maintaining that information.

1.2 Executive Summary of Technical Study Result

   The strength of the TISDAG project's DAG proposal is that it defines
   the necessary technical infrastructure to provide a single-access-
   point service for information on Swedish Internet users.  The
   resulting service will provide uniform access for all information --

the same level of access to information (7x24 service), and the same
information made available, irrespective of the service provider
responsible for maintaining that information, their directory service
protocols, or the end-user's client access protocol.

Instead of requiring centralized mirroring of complete information
records from Swedish directory service providers, the DAG system uses
a well-defined index object summary of that data, updated at the
directory service provider's convenience.  When an end-user queries
the DAG, the referral information is used (by the end-user's
software, or by a module within the DAG, as appropriate) to complete
the final query directly at the directory service provider's system.
This ensures that the end-user gets the most up-to-date complete
information, and promotes the directory service provider's main
interest:  its service.  The architecture of the DAG itself is very
modular; support for future protocols can be added in the operational
system.

1.3 Document Overview

This document is broken into 5 major sections:

Requirements: As a service, the DAG system will have several
different types of users.  In order to be successful, those users'
needs (requirements) must be met.  This in turn defines certain
constraints, or system requirements, that must be met.  This section
aims to capture the baseline requirement assumptions to be addressed
by the system, and thus lays the groundwork on which the rest of the
proposed system is built.

Functional Specification Overview: Working from the users'
requirements, specific technologies and  functionality details are
outlined to architect a system that will meet the stated
requirements.  This includes a conceptual architecture for the
system.  While the Requirements section outlines the needs the
different users have for the eventual DAG system,  implementing and
providing the eventual service will entail constraints or conditions
that need to be met in order to be able to participate in the overall
system.

Architecture: Once the system has been defined conceptually, a
proposed software architecture is specified to produce the desired
functionality and meet the stated requirements.

Software Specifications: This section provides the specifications for
software components to meet the architecture described above.

Service Specifications: Once the software has been designed, the
success of the DAG system will rest on its operational
characteristics.  Details of service requirements are given in this
section.

1.4 Terminology

DAG-CAP: Client Access Point -- point of communication between
client-access software and the DAG system.

DAG-System: The Directory Access Gateway system resulting from the
TISDAG  project.  A collection of infrastructural software and
services for the purpose of providing unified access to Swedish
whitepages information.

DAG/IP: DAG-Internal Protocol -- communication protocol used between
software components of the DAG.

End-User: People performing White Pages searches and look-ups (via
various forms of client software).

DAG-SAP:  Service Access Point -- point of communication between the
DAG and WDSP software.

WDSP: Whitepages Directory Service Provider -- ISPs, companies, or
other interested entities.

Whitepages Information: Collected information coordinates for
individual people.  This typically includes (but is not limited to) a
person's name, and e-mail address.

2.0 Requirements

There are 2 primary classes of users for the proposed Whitepages
directory access gateway:

- End-users
- WDSPs

As outlined below, needs of each of these user classes imposes a set
of constraints on the design of the DAG system itself.  Some of the
requirements shown below are assumed starting criteria for the DAG
service; others have been derived from data collected in the
Technical Survey or other expertise input.

2.1 End-User Requirements

    The End-User is to be provided with a specific set of search types:

    Name
    Name + Organization
    Role + Organization
    Name + Locality
    Name + Organization + Locality
    Role + Organization + Locality

    The search results will, if available, include the following
    information for each "hit":

    - Full name
    - E-mail address
    - Role
    - Organization
    - Locality
    - Full address
    - Telephone numbers

    Access to the service must be available through reasonable and
    current protocols -- such that directory-service-aware software can
    make use of it seamlessly, and there are no reasonable technological
    impediments to making this service useful to all Swedish Internet
    users.

    Following on that, its responses are expected to be timely; a
    standard search should not take more time than the average access to
    a web-server.

2.2 WDSPs Requirements

    Given that the WDSPs that participate in this service are already in
    the business of providing a service of whitepages information, they
    have certain requirements that must be respected in order to make
    this a successful and useful service to all concerned.

    The DAG system must provide reasonable assurances of data integrity
    for WDSPs; the information the End-User sees should correspond
    directly to that provided by the WDSPs.  The DAG system should be
    non-preferential in providing whitepages information -- the service
    is to the End-User, and the source of whitepages information should
    not influence the search and information presentation processes.

The DAG system must be able to reflect information updates within a
reasonable time after receipt from WDSPs; on the flip side, while the
DAG system will function best with regular updates from WDSPs, the
update and participation overhead for WDSPs should be held within
reasonable bounds of what the WDSP should do to support regular
access to its information.

Furthermore, given that WDSPs provide directory service information
with an eye to value-added service, wherever possible End-Users
should be redirected to the WDSP responsible for individual directory
service entries for final and further information.

2.3 DAG-System Requirements

In order to address the requirements of End-Users and WDSPs, the DAG
system itself has certain design constraints that must be taken into
account.

The system must be implementable/operational by Dec 31/98 -- which
implies that it must be designed and constructed with already extant
technologies.

The System will have certain requirements for participation -- e.g.,
7x24 WDSP availability.

In terms of scaling, the system should be able to handle 8M records
at the outset, with a view to handling larger information systems in
the future.

The system must also be capable of extension to other, related
applications (e.g., serving security certificate information).

3.0 Functional Specification

In the TISDAG pilotservice we have decided to apply some limitations
as to what is specified for the DAG/IP.  These limitations are
presented in this text in the following manner:

     TISDAG: This is a TISDAG comment

3.1 Overview

The conceptual environment of the DAG system can be described in
three major components:

- client access software for end-users
- the DAG system core
- WDSP directory service software

   This is illustrated in Figure 3.1

   The DAG (Directory Access Gateway) is the infrastructural core of the
   service; it maintains the necessary data and transformation
   facilities to permit the smooth connection of diverse directory
   service Client Software to the existing WDSPs' directory servers.
   The key challenges in designing this portion of the system are:

   Quantity of data -- the quantity of whitepages information that will
   be made available, and diversity of its sources (different WDSPs)
   introduce challenges in terms of finding a structure that will allow
   efficient searching, and facilitate the timeliness of updating the
   necessary information.

   Multiplicity of access protocols -- in order to support the use of
   existing whitepages-aware software with a minimum of perturbation,
   the DAG system will have to present a uniform face in several
   different access protocols, each with its own information search and
   representation paradigm.

   This specification will outline the following areas:

   - the functioning of the DAG core itself
   - the interface between the DAG core and End-Users' Directory Service
     Access software
   - the interface between the DAG core and Directory Services Servers

3.2 The DAG Core

   In order to reduce the quantity of data the DAG itself must maintain,
   and to keep the maintenance of the whitepages information as close as
   possible to the source of information (the WDSPs themselves), the DAG
   will only maintain index information and will use "query routing" to
   efficiently refer End-User queries to WDSPs for search refinement and
   retrieval of information.  Although originally developed for the
   Whois++ protocol, query routing is being pursued in a protocol-
   independent fashion in the IETF's FIND WG, so the choice of this
   approach does not limit the selection and support of whitepages
   access protocols.

   The DAG will look after pursuing queries for access protocols that do
   not support referral mechanisms.  In order to achieve the support of
   multiple access protocols and differing data paradigms, the DAG will
   be geared to specifically support a limited set of whitepages
   queries.

```
                                 +--------+       @
                           +  ->|        |      -+-
                          /|Protocol|        |       |
                         / |   /  +--------+      / \
                        /  |  "B"
                       +   |  /
                       |   |<-
          +-------+    |   |
      O   |       |    |   |
    -+-   |       |<--------->|   |
     |    |       | Protocol  |   |
    / \   |       |   "A"     |   |<-
          +-------+    |   |Protocol
                       |   |   \
                   +   |  "A"  +--------+       @
                    \  |    \  |        |      -+-
                     \ |     ->|        |       |
                      \|       +--------+      / \
                       +
```

```
                           The
          End      Client      DAG          Directory   Directory
          Users    Software    System       Server      Service
                               Core         Software    Providers
```
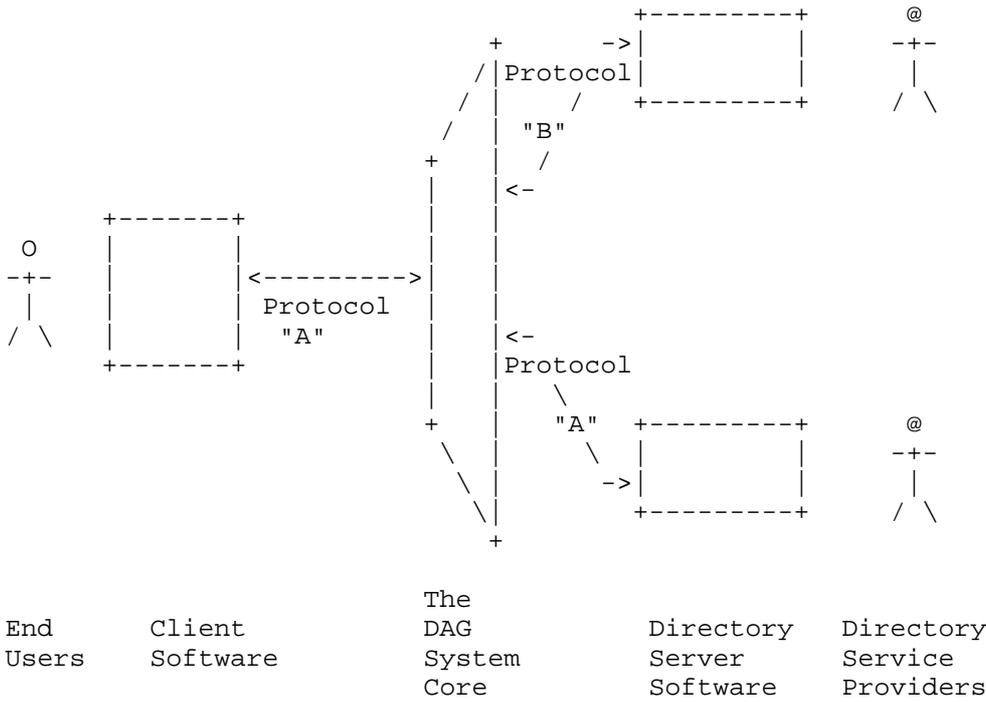
                Figure 3.1 The role of the DAG system

3.3 Client Interface

   The DAG will respond to End-User queries in

   - e-mail (SMTP)
   - WWW (HTTP)
   - LDAPv2
   - Whois++
   - LDAPv3

   The DAG will provide responses including the agreed-upon data.  For
   access protocols that can handle referrals, responses will be data
   and/or referrals in that query protocol.  These are Whois++ and
   LDAPv3.  N.B.: the LDAPv3 proposal defines a referral as a URL; no
   limitation is placed on the access protocol.  However it cannot be
   assumed that all clients will be able to handle all access protocols,
   so only referrals to LDAPv3 servers will be returned.

3.3.1 Acceptable User Input

    User Input is defined in terms of

    - Searchable Attributes
    - Matching semantics
    - Character sets

    These, in conjunction with the DAG schema, defined in Appendix A,
    form the basis of the required query expression.  Individual queries
    are discussed in more detail in the Client Access Point (DAG-CAP)
    component descriptions for supported protocols.

    Supported Query Types

    The DAG system is designed to support fragment-matching queries on a
    limited set of data attributes -- "Name", "Organizational Role",
    "Organization", and "Locality".  The selected permissible query
    combinations of attributes are listed in Table 3.1.  From the table
    it can be seen that not all combinations of the three attributes are
    supported -- only those that are needed for the desired
    functionality.

    Symbol   Description
    -------  -----------
    N        Name
    NL       Name + Locality
    NO       Name + Organization
    NOL      Name + Organization + Locality
    RO       Role + Organization
    ROL      Role + Organization + Locality

    Table 3.1 DAG-supported queries

    The RO and ROL queries are separated from the rest as they are
    searches for "virtual" persons -- roles within an organization (e.g.,
    president, or customer service desk) for which one might want to find
    contact information.

    Matching Semantics

    As befits the individual client query protocols, more string matching
    expressions may be provided.  The basic semantics of the DAG expect
    the following to be available in all client access software (as
    relevant):

- Full word, exact match
- Word substring match (E.g., "cat" would match "scatter")
- Case-sensitive and case-insensitive matching

    TISDAG: LDAP/X.500, supports case-sensitivity as such but some of
    the most used attributes, such as the commonName attribute, are
    defined in the standard to be of the case-insensitive
    attributetypes.  The impact on the DAG system is that even if the
    index collected from a LDAP/X.500 server might have upper and
    lower case letters in the tokens, they can not be handled as such
    since that would be inferring meaning in something which is
    natively regarded as meaningless.  The conclusion of the above is
    that The Referral Index should be case-insensitive and case-
    sensitivity should be supported by the SAPs if the native access
    protocol supports it.

Character Sets

Wherever possible, the DAG System supports and promotes the use of
Unicode Version 2.0 for character sets (see [21]) specifically the
UTF-8 encoding (see Appendix A.2 of [21] or [20]) Accommodation is
made, where necessary, to support the deployed base of existing
software.

Specifically:

DAG/IP: All internal communications using the DAG/IP are carried out
in UTF-8.

    TISDAG: not just UTF-8, but UTF-8 based on composed UNICODE
    version 2 character encodings.

DAG-CAP input: Where specific access protocols permit selection of
character sets, DAG-CAPs must support UTF-8.  They may additionally
support other anticipated character set encodings.

DAG-SAP communications with WDSPs:  Where specific access protocols
permit selection of character sets, DAG-SAPs must support UTF-8 and
use UTF-8 whenever the remote WDSP supports it.  They may
additionally support other character set encodings.

CIP Index Objects: The Index Objects supplied by the WDSPs to the DAG
system shall contain data encoded in UTF-8.

    TISDAG: The same limitation as for DAG/IP, that is the basic data
    should be UTF-8 encoded composed UNICODE version 2 character
    encodings.

3.3.2 Data Output Spec

   Schema Definition

   The schema used for the DAG service  is defined in Appendix A.  This
   is a very basic information schema, intended to carry the necessary
   information  for the DAG service, and not more.  Although generic
   "whitepages" schema definitions do exist the more sophisticated and
   detailed the information presentation, the more difficult it is to
   map the schema seamlessly across protocols of different paradigms.
   Thus, the "KISS" ("Keep it simple, sir") principle seems appropriate
   here.

   Individual DAG-CAPs define how they express this schema.

   Referral Definition

   For client access protocols that make use of the concept of
   referrals, DAG-CAP definitions will define the expression of
   referrals in those protocols.  The DAG/IP defines the expression of
   referrals (see Appendix  C).

   Error conditions

   Each DAG-CAP may provide more detailed error messages, but will
   define minimally the support for the following error conditions:

   - unrecognized query
   - too many hits

   Apart from these errors, the DAG-CAP may choose to refuse a query by
   redirecting the end-user to a different DAG-CAP of the same protocol.

3.4 Directory Server Interface

   The DAG will use the Common Indexing Protocol (CIP) server-server
   protocol to obtain updated index objects from WDSPs.  For query-
   routing purposes, WDSPs are expected to  provide Whois++, LDAPv2 or
   LDAPv3 interface to their data (although their preferred access may
   be something completely different).  N.B.:  In the responses from the
   technical survey, all respondents currently provide access to their
   service in one of these protocols.

   In order to provide a useful and uniform service, WDSPs are expected
   to provide 7x24 access to their whitepages information.  WDSPs are
   also expected to implement operations, administration, maintenance,
   and provisioning processes designed to minimize service down time for
   both planned and unplanned administration and maintenance activities.

4.0 Architecture

4.1 Software Components

   The conceptual architecture of the DAG is represented in Figure 4.1.
   General architectural specifications are described below, followed by
   individual component specifications Sections 5.5 through 5.12.

4.1.1 Internal Communications

   Communications between components of the DAG  will be by TCP/IP
   connections, using the DAG-Internal Protocol (DAG/IP).  DAG/IP is
   used by DAG-CAPs to communicate with the Referral Index and DAG-SAPs.
   Thus, the DAG/IP defines

   - the DAG-CAPs' range of query ability in the Referral Index (to
     gather referrals in response to the end-user's requests)
   - the responses (and their formats) of the Referral Index to the
     DAG-CAP requests
   - the DAG-CAPs' range of query ability to the DAG-SAPs for pursuing
     referrals when the DAG-CAP needs to do chaining for the client
     access software
   - the responses (and their formats) of the DAG-SAPs to the DAG-CAPs.

   The detail of the planned DAG/IP is given in Appendix C.  The detail
   of the DAG-CAP--Referral Index and DAG-CAP--DAG-SAP interactions  is
   given in the definitions of individual DAG-CAPs and DAG-SAPs, below
   (Sections 5.5 through 5.12).

4.1.2 Referral Index

   The Referral Index is responsible for maintaining the index of WDSP
   information, and providing a list of reasonable referrals in response
   to DAG-CAP search requests.  These "referrals" provide pointers to
   identify WDSPs that may have information that matches the end-user's
   query.

4.1.3 DAG-CAPs

   Individual DAG-CAPs are responsible for providing a particular client
   access protocol interface to the DAG service.  DAG-CAPs receive end-
   user queries in a particular query access protocol, convert the
   request into a query for the Referral Index ( i.e., expressed in
   DAG/IP), and then convert the Referral Index's response into a form
   that is appropriate for the client access protocol.  This may mean
   passing back the referrals directly, calling on DAG-SAPs to do the
   work of translating the referral into results ("chaining"), or a
   combination of both.

```
                 +----------------------------------+
                 |+====+                            |
        HTTP   <-->+|    |<------+  (Full chaining)    |
                 |||    |       |                    |
                 |+====+        |                    |
                 |              |          +----+|
                 |              |  Referral-->|    ||
                 |              |  Result  <--|    |+<--> Whois++
                 |              |          +----+|
                 |+====+        |                    |
        SMTP   <-->+|    |<------+  (Full chaining)    |
                 |||    |       |                    |
                 |+====+        |                    |
                 |              |          +----+|
                 |              |  Referral-->|    ||
                 |              |  Result  <--|    |+<--> LDAPv2
                 |              |          +----+|
                 |+====+        |                    |
      Whois++<-->+|    |<------+  (Chain LDAPv2/3)   |
                 |||    |       |                    |
                 |+====+        |                    |
                 |              |          +----+|
                 |              |  Referral-->|    ||
                 |              |  Result  <--|    |+<--> LDAPv3
                 |              |          +----+|
                 |+====+        |                    |
        LDAPv2 <-->+|    |<------+  (Full chaining)    |
                 |||    |       |                    |
                 |+====+        |                    |
                 |              |                    |
                 |+====+        |                    |
        LDAPv3 <-->+|    |<------+  (Chain Whois++)   |
                 |||    |       |                    |
                 |+====+        |                    |
                 |              |                    |
                 |              v                    |
                 |   +----------------------+        |
                 |   |  Referral Index      |<--------------> Common
                 |   |                      |       | Indexing Protocol
                 |   +----------------------+       | (CIP)
                 +----------------------------------+
```
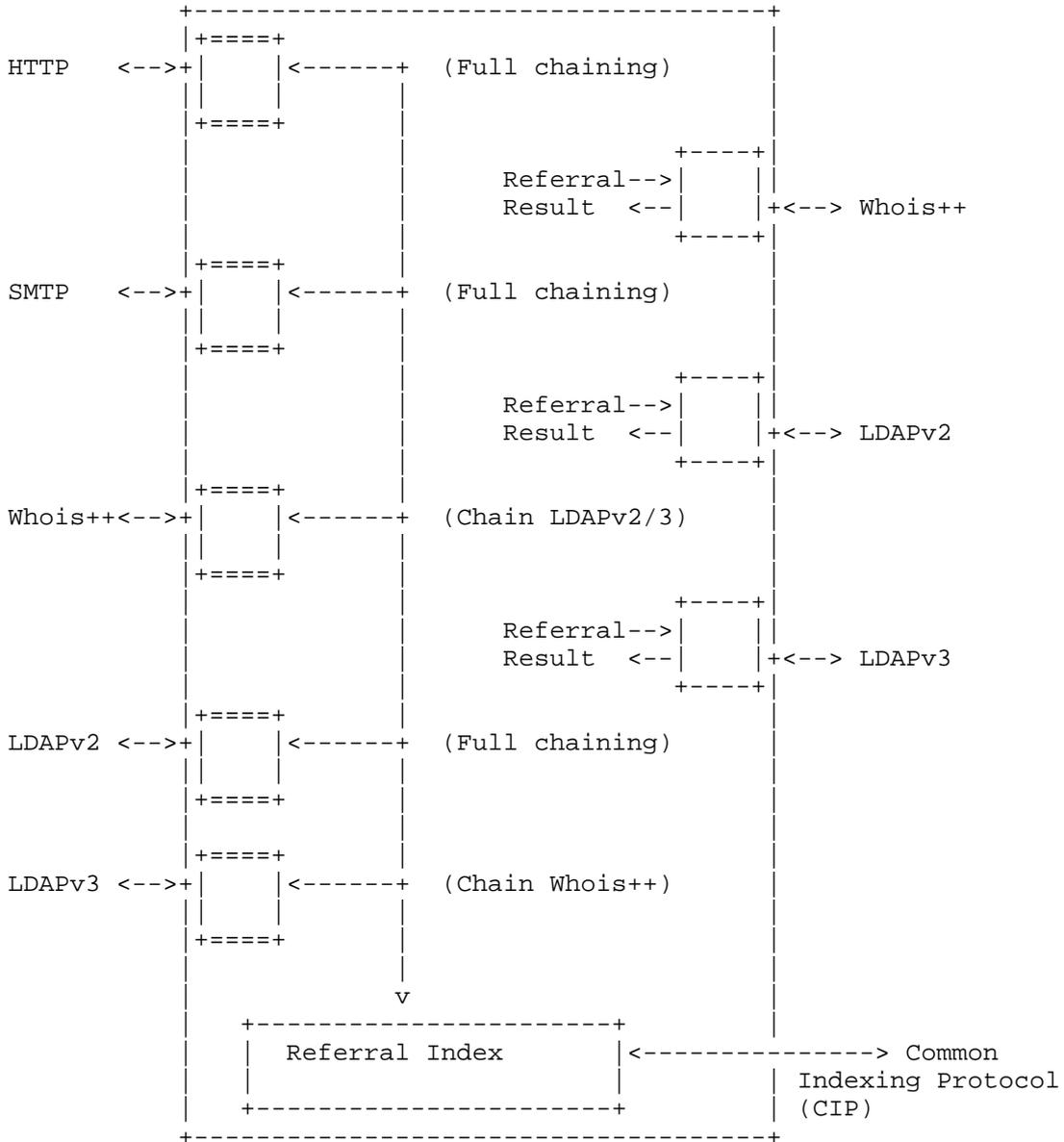
        All internal communications are in DAG/IP.

        Figure 4.1 Conceptual Architecture of the DAG

4.1.4 DAG-SAPs

   Individual DAG-SAPs are called upon (by DAG-CAPs) to take DAG-
   generated referrals and pursue them -- issuing the indicated query at
   the specified WDSP service.  Results from individual WDSPs are
   converted back into DAG/IP-specific format for the DAG-CAP that made
   the request.  Each DAG-SAP is responsible for handling referrals to
   WDSPs of a particular protocol (e.g., LDAPv2, Whois++, etc).

4.2 Important Architectural Notes

   This section notes some of the thinking that has driven the
   architectural and software design specification for the DAG system.
   This helps to provide the context in which to understand the software
   specifications that follow, and should give clues for the eventual
   extension of the DAG system.  This section also acts, in some ways,
   as an FAQ (Frequently Asked Questions) section, as the content is
   shaped by questions received during the tech spec development phase.
   It attempts to illuminate context that may not otherwise be apparent
   on a first reading of the software specifications.

4.2.1 2 Distinct Functions:  Referrals and Chaining

   At all times, it must be kept in mind that the primary function of
   the DAG system is to provide users with referrals to WDSP services
   that may have the information they seek.  Since it is the case that
   not all supported client protocols can handle referrals, the DAG
   system also provides a chaining service to pursue referrals that the
   user's client software cannot handle itself.  This chaining service
   does attempt to match the user's query against data from WDSPs, but
   this is to be seen as a secondary, or support function of the DAG
   system.  In the perfect future, all access protocols will be able to
   handle all referrals!

4.2.2 Limited Query and Response Semantics

   The DAG system does not attempt to be a chameleon, or the ultimate
   whitepages query service.  It focuses on providing referrals for
   information on the limited number of query types outlined in the
   functional specifications of the DAG service.  This makes the DAG
   system a good place to start a search, but refinements and detailed
   inquiries are beyond its scope.

4.2.3 Visibility

   Given the limited query syntax of the DAG system it will not always
   be possible to exactly match a query posed to a CAP into a query
   posed to a SAP.  This will have the effect that for instance a LDAPv2

      client that issues a query to the DAG system which by the DAG system
      is chained to a LDAP server might not get the same results as if the
      client where directly connected to the server in question.

4.2.4 Richness of Query semantics

      Even the limited query syntax of the DAG system is capable of
      expressing queries that might NOT be possible to represent in the
      access protocols to the WDSPs.  In these cases the DAG-SAP either can
      refuse the query or try to emulate it.

4.2.5 N+M Protocol Mappings

      As part of the chaining service offered by the DAG system, a certain
      amount of mapping between protocols is required -- in theoretical
      terms, there  are "N" allowable end-user query access protocols, and
      "M" supported WDSP server protocols.  The architecture of the
      software is constructed to use a single internal protocol (the
      DAG/IP) and data schema, providing a common language between all
      components.  Without this, each input protocol module (DAG-CAP) would
      have to be constructed to be able to handle every WDSP protocol --
      NxM protocol mappings.  This would make the system complex, and
      difficult to expand to include new protocols in future.

4.2.6 DAG-CAPs and DAG-SAPs are completely independent of each other

      For the above reasons, the DAG-CAP and DAG-SAP modules are intended
      to be completely independent of each other.  A DAG-SAP responds to a
      query that is posed to it in the DAG/IP, without regard to the
      protocol of the DAG-CAP that passed the query.

4.2.7 The Role of the DAG-CAP

      Thus, the DAG-CAP is responsible for using the DAG/IP to obtain
      referral information and, where necessary, chained responses.  Where
      necessary, it performs adjustments to accommodate the differences in
      semantics between the DAG/IP and its native protocol.  This might
      involved doing post-filtering of the results returned by the DAG-SAPs
      since the query issued in DAG/IP to the DAG-SAP might be "broader"
      then the original query.

      Thus, the DAG-CAP "knows" only 2 protocols:  its native protocol, and
      the DAG/IP.

4.2.8 The Role of the DAG-SAP

   Similarly, the DAG-SAP is responsible for responding to DAG/IP
   queries by contacting the designated WDSP server.  Where necessary,
   it performs adjustments to accommodate the differences in semantics
   between the DAG/IP and its native protocol.  These adjustments might
   mean that, as a consequence, the DAG-SAP will receive results that do
   not match the original query.  In such cases the DAG-SAP should
   attempt to do post-pruning in order to reduce the mismatch between
   the original query and the results returned.

   Thus, the DAG-SAP "knows" only 2 protocols:  its native protocol, and
   the DAG/IP.

4.2.9 DAG/IP is internal

   No module outside of the DAG system should be aware of the DAG/IP's
   construction.  End-users use the query protocols supported by DAG-
   CAPs; WDSPs are contacted using the query protocols supported in the
   DAG-SAPs.

4.2.10 Expectations

   The expectation is that the DAG system, although defined as a single
   construct, will operate by running modules on several different,
   perhaps widely distributed (in terms of geography and ownership),
   computers.  For this reason, the DAG/IP specified in such a way that
   it will operate on inter-machine communications.

4.2.11 Future Extensions

   The DAG system architecture was constructed with a specific view to
   extensibility.  At any time, an individual component may be improved
   (e.g., the Mail DAG-CAP may be given a different query interface)
   without disrupting the system.

   Additionally, future versions of the DAG system may support other
   access protocols -- for end-users, and for WDSPs.

5.0 Software Specifications

5.1 Notational Convention

   It is always a challenge to accurately represent text protocol in a
   printed document; when is a new line a "newline", and when is it an
   effect of the text formatter?

In order to be adequately illustrated, this document includes many
segments of protocol grammars, sample data, and sample input/output
in a text protocol.  In order to distinguish newlines that are
significant in a protocol, the symbol

   <NL>

is used.  For example,

This is an example of a very long line of input.  There is only one
newline in it (at the end), in spite of the fact that this document
shows it spanning several lines of text.<NL>

5.2 DAG-CAP Basics

5.2.1 Functionality

   Every DAG-CAP must support the full range of DAG queries, as defined
   in 3.3.1.

   Each DAG-CAP accepts queries in its native protocol.  Individual
   DAG-CAP definitions define the expected expression of the DAG queries
   in the native protocol.

   The DAG-CAP is then responsible for:

   - converting that expression into a query in the DAG/IP to obtain
     relevant referrals from the Referral Index.  This might mean that
     parts of the original query are disregarded (e.g., if the query
     included attributes not supported by the DAG application, or if the
     query algebra was not supported by the DAG application);
   - returning referrals in the client's native protocol, where
     possible;
   - expressing the client query to the necessary DAG-SAPs, given the
     limitations mentioned above, to chain those referrals not usefully
     expressible in the client's native protocol;
   - possibly doing post-filtering on the DAG-SAP results; and
   - converting the collected DAG-SAP results for expression in the
     client's native protocol (and schema, where applicable).

   Each DAG-CAP defines the nature of the interaction with the end-user
   (e.g., synchronous or asynchronous, etc).  Additionally, each DAG-CAP
   must be able to carry out the following, in order to permit load-
   limiting and load-balancing in the DAG system:

   - direct the client to a different DAG-CAP of the same type (for
     load-balancing)

   - decline to return results because too many referrals were generated
     (to discourage data-mining).  Ideally, this should include the
     generation of a message to refine the query in order to produce a
     more manageable number of referrals/replies.

   DAG-CAPs must be capable of accepting and respecting DAG-SAP service
   referrals (for DAG-SAP load-sharing).

   In protocols that permit it, the DAG-CAP should indicate to the end-
   user which services were unavailable for chaining referrals (i.e., to
   indicate there were parts of the search that could not be completed,
   and information might be missing).

      TISDAG: Any CAP that receives commands other than queries, like
      help, answers those on its own.  A CAP should not pass any system
      command on to the RI.

5.2.2 Configuration

   It must be possible to change the expected address of the DAG-CAP by
   configuration of the software (i.e., host and port, e-mail address,
   etc).

   For DAG-CAPs that need to access DAG-SAPs for query chaining, for
   each type (protocol) of DAG-SAP that is needed, the DAG-CAP must be
   configurable in terms of:

   - at least one known DAG-SAP of every necessary protocol to contact
   - for each DAG-SAP, the host and port of the DAG-SAP software

   The DAG-CAPs must also be configurable in terms of a maximum number
   of referrals to handle for a user transaction (i.e., to prevent data
   mining, the DAG-CAP will refuse to reply if the query is too general
   and too many hits are generated at the Referral Index).

   The DAG-CAP must be configurable in terms of alternate DAG-CAPs of
   the same type to which the end-user software may be directed if this
   one is too busy.

5.2.3 Error handling

   Apart from error conditions arising from the operation of the DAG-CAP
   itself, DAG-CAPs are responsible for communicating error conditions
   occurring elsewhere in the system that affect the outcome of the
   user's query (e.g., in the DAG-RI, or in one or more DAG-SAPs).

   If the DAG-CAP sends a query to the DAG-RI and receives an error
   message, it should attempt to match the the received DAG errorcode
   into its native access protocol's error codes.  The same action is
   appropriate when the DAG-CAP is "chaining" the query to one DAG-SAP.

   There are also occasions when the DAG-CAP may have to combine
   multiple errorcodes into a single expression to the user.  When the
   DAG-CAP is "chaining" the query through DAG-SAPs to one or more
   WDSPs, situations can arise when there is a mix of responsecodes from
   the DAG-SAPs.  If this happens, the DAG-CAP should try to forward
   information to the end-user software that is as specific as possible,
   for instance which of the WDSPs has not been able to fulfill the
   query and why.

   See Appendix D for more information concerning error condition
   message mappings.

5.2.4 Pruning of results

   Since there is no perfect match between the query syntaxes of the DAG
   system on one hand and the different access protocols that the DAG-
   CAPs and DAG-SAPs supports on the other, there will be situations
   where the results a DAG-CAP has to collect is "broader" then what
   would have been the case if there had been a perfect match.  This
   might have adverse effects on the system to the extent that
   administrative limits will "unnecessary" be exceeded on WDSPs or that
   the collected results exceeds the sizelimit of the DAG-CAP.

   Since the DAG-CAP is the only part of the DAG system that actually
   knows what the original query was, the DAG-CAP can prune the results
   received from the DAG-SAPs in such a way that the results presented
   to the client better matches the original question.

5.3 DAG-SAP Basics

5.3.1 Functionality

   Every DAG-SAP must support the full range of DAG queries, as defined
   in 3.3.1.  Results must be complete DAG schemas expressed in well-
   formed DAG/IP result formats (see Appendix C).  Each DAG-SAP accepts
   queries in DAG/IP and converts them to the native schema and protocol
   for which it is designed to proxy.

   The DAG-SAP is then responsible for

   - converting the query into the native schema and protocol of the
     WDSP to which the referral points.  (If the query is not
     representable in the native protocol, it must return an error

      message.  If it is emulatable, the DAG-SAP can attempt emulate it
      by posing a related query to the WDSP and post-pruning the results
      received);
    - contacting that WDSP, using the host, port, and protocol
      information provided in the referral;
    - negotiating the query with the remote WDSP;
    - accepting results from the WDSP, possibly doing post-filtering on
      the result set; and
    - conveying the results back to the calling DAG-CAP using the DAG/IP
      and its schema.

   Note that this implicitly means that the DAG-SAP is responsible for
   chaining and pursuing any referrals it receives from WDSP services.
   The DAG-SAP returns only search results to the DAG-CAP that called
   it.

5.3.2 Configuration

   DAG-SAPs must be configurable to accept connections only from
   recognized DAG components.

   DAG-SAPs that have service limits must be configurable to redirect
   DAG-CAPs to alternate DAG-SAPs of the same type when necessary.

5.3.3 Error handling

   A DAG-SAP must translate error codes received from a WDSP server to
   DAG error codes according to Appendix D.

5.3.4 Pruning of results

   Since it might not be possible to exactly map a DAG query into a
   query in the access protocol supported by the a DAG-SAP, the DAG-SAP
   should try to translate it into a more general query (or if necessary
   into a set of queries).  If so, the DAG-SAP must then prune the
   result set received before furthering it to the DAG-CAP.

5.3.5 Constraint precedence

   Some constraints, search and case, can appear both as local and
   global constraints.  If this happens in a query then the local
   constraint specification overrides the global.  For a query like the
   following:

   fn=leslie;search=exact and org=think:search=substring

   the resulting search constraint for "fn=leslie" will be "exact" while
   it for "org=think" will be "substring".

5.4 The Referral Index

5.4.1 Architecture

   The Referral Index contains (only) information necessary to deliver
   referrals to DAG-CAPs based on the query types supported by the DAG
   itself.  The Referral Index creates an index over these objects so
   that it can respond to DAG-CAP queries using the DAG/IP.  The
   information is drawn directly from interactions with participating
   WDSPs' software, using the Common Indexing Protocol (CIP).

5.4.2 Interactions with WDSPs (CIP)

   WDSPs that wish to participate in the DAG system must register
   themselves (see Section 5.4.6).  Once registered, the Referral Index
   will interact with the WDSPs using the Common Indexing Protocol as
   defined in [1], using the Index Object defined in Section 5.4.3.

5.4.3 Index Object Format

   The CIP index object type is based on the Tagged Index Object as
   defined in [12].  Appendix E details the expected content of the
   index objects as they are to be provided by the WDSPs.

      TISDAG: The tokens in the Tagged Index Object should be UTF-8
      encoded composed UNICODE version 2 character encoding.

5.4.4 DAG-Internal I/O

   The Referral Index interacts with the rest of the DAG internal
   modules (DAG-CAPs) by listening for queries and responding in the
   DAG/IP (defined in Appendix C).

5.4.5 The Index Server

   The Referral Index must index the necessary attributes of the CIP
   index object in order to respond to queries of the form described in
   Table 3.1.

   The semantics of the chosen CIP object (defined in Appendix E) are
   such that a referral to a WDSP server is sent back if (and only if)

   - the index object of the WDSP contains all the tokens of the query,
     in the attributes specified, according to the logic of the DAG/IP
     query, and
   - all of those tokens are found with a common tag.

This means that a query for the name "Fred Flintstone" (2 tokens)
will yield a referral to a server that has a record for "Fred Amadeus
Flintstone", but not to a WDSP with 2 differently tagged records, for
"Fred Amadeus" and "Julie Flintstone".  Depending on the access
protocol being used and the original end-user query, the referral to
the WDSP with "Fred Amadeus Flintstone" may yield a successful
result, or it may not.  But, it is known that the other WDSP would
not have yielded successful searches.  That is, the referral approach
may yield false-positive results, but will not miss appropriate
WDSPs.

5.4.6 Configuration

The Referral Index must provide the ability to register interested
WDSPs, as outlined in Appendix E.

The Referral Index must be able to configure the port for DAG/IP
communications.  Also, it must be configurable to recognize only
registered DAG-CAPs.

5.4.7 Security

The Referral Index will accept queries only from recognized
(registered) DAG-CAPs.  This will reduce "denial of service" attack
types, but is also a reflection on the fact that the Referral Index
uses the DAG/IP, (i.e., internal) protocol, which should not be
exposed to non-DAG software.

The Referral Index must be able to use authenticated communication to
receive data from WDSPs (see Appendix E).

5.5 Mail (SMTP) DAG-CAP

This is the default Mail DAG-CAP.  More sophisticated ones could
certainly be written -- e.g., for pretty-printed output, or for
handling different philosophies of case-matching.

This DAG-CAP has been designed on the assumption that mail queries
will be human-generated (i.e., using a mail program/text editor), as
opposed to being queries formulated by software agents.  The input
grammar should therefore be simple and liberal in acceptance of
variations of whitespace formatting.

5.5.1 Mail DAG-CAP Input

   Mail DAG-CAP input is expected to be a regular or MIME-encoded (see
   [9] and [10]) SMTP mail message, sent to an advertised mail address.
   The mail DAG-CAP parses the message and replies to it with a MIME-
   encoded message containing the results of the DAG search.

   One query is accepted per e-mail message -- text after a single valid
   query has been read is simply ignored.

   The body of the query message must follow the syntax defined below.
   Note that all input control terms ("type=", "name=" etc) are shown in
   lower case for convenience, but could be upper case or mixed case on
   input.

```
mailquery       = [mnl] [controls] mnl terms mnl
controls        = [msp] "searchtype" [msp] "=" [msp]
                     ( matchtype /
                       casetype /
                       matchtype msp casetype /
                       casetype msp matchtype /
                       <nothing> )
matchtype       = "substring" / "exact"
                ; default:  substring
casetype        = "ignore" / "sensitive"
                ; default:  ignore

terms           = n / n-l / n-o / n-o-l / r-o / r-o-l

n               = n-term
n-l             = ( n-term l-term  / l-term n-term)
n-o             = ( n-term o-term  / o-term n-term )
n-o-l           = ( n-term o-term l-term /
                 n-term l-term o-term /
                 l-term n-term o-term /
                 l-term o-term n-term /
                 o-term l-term n-term /
                 o-term n-term l-term )
r-o             = ( r-term o-term / o-term r-term )
r-o-l           = ( r-term o-term l-term /
                 r-term l-term o-term /

                 l-term o-term r-term /
                 l-term r-term o-term /
                 o-term l-term r-term /
                 o-term r-term l-term )
n-term          = [msp] "name" [msp] "=" [msp] string mnl
o-term          = [msp] "org" [msp] "=" [msp] string mnl
```

```
    l-term          = [msp] "loc" [msp] "=" [msp] string mnl
    r-term          = [msp] "role" [msp] "=" [msp] string mnl

    string          = <US-ASCII or quoted-printable encoded
                      ISO-8859-1 or UTF-8 except nl and sp>
    msp             = 1*(sp)
     sp              = " "
    mnl             = 1*(nl)

    nl              = <linebreak>
```

    The following are valid mail queries:

    Example 1:

```
    searchtype =   <NL>
    name = thinking cat<NL>
```

    Example 2:

```
    searchtype = exact ignore<NL>
    name=thinking cat<NL>
```

    Example 3:

```
    role=thinking cat<NL>
    org =space colonization<NL>
```

    Example 4:

```
    name=thinking cat <NL>
    <NL>
    <NL>
    My signature line follows here in the most annoying
    fashion <NL>
```

    Note that the following are not acceptable queries:

    Example 5:

```
    searchtype= exact substring <NL>
    name = thinking cat <NL>
```

    Example 6:

```
    name=thinking cat org= freedom fighters anonymous<NL>
```

In Example 5, two conflicting searchtypes are given.  In Example 6,
no linebreak follows the n-term.

5.5.2 Translation from Mail query to DAG/IP

Querying the Referral Index

A key element of translating from the Mail DAG-CAP input into the
DAG/IP query format is to "tokenize" the input terms into single
token elements for the DAG/IP query.  For example,  the n-term

name= thinking cat<NL>

is tokenized into 2 n-tokens:

thinking
cat

which are then mapped into the following in the DAG/IP query (dag-n-
terms):

FN=thinking and FN=cat<NL>

The same is true for all r-terms, l-terms and o-terms.  The primary
steps in translating the mail input into a DAG/IP query are:

translate quoted-printable encoding, if necessary
translate base64 encoding, if necessary
tokenize the strings for each term
construct the DAG/IP query from the resulting components, as
described in more detail below

DAG/IP constraints are constructed from the searchtype information in
the query.

```
dag-matchtype = "search=" <matchtype> /
            "search=substring"  ; if matchtype not
                                ; specified

dag-casetype  = "case=ignore"  /    ; if casetype not
                                ; specified or
                                ; casetype=ignore
            "case=consider"     ; if casetype=sensitive

constraints   = ":" dag-matchtype ";" dag-casetype
```

The terms for the DAG/IP query are constructed from the tokenized
strings from the mail input.

```
dag-n-terms   = "FN=" n-token 0*( " and FN=" n-token)
dag-o-terms   = "ORG=" o-token 0*( " and ORG=" o-token)
dag-l-terms   = "LOC=" l-token 0*( " and LOC=" l-token)
dag-r-terms   = "ROLE=" r-token 0*( " and ROLE=" r-token)
```

This means that the relevant DAG/IP queries are formulated as one of
two types:

```
dagip-query   = ( ( ( n-query / nl-query / no-query /
                    nol-query ) [" and template=DAGPERSON"]":"
                 dag-matchtype ";" dag-casetype) /
                ( ( ro-query / rol-query )
                  [" and template=DAGORGROLE"]":"
                  dag-matchtype ";" dag-casetype)  )

n-query       = dag-n-terms
nl-query      = dag-n-terms " and " dag-l-terms
no-query      = dag-n-terms " and " dag-o-terms
nol-query     = dag-n-terms " and " dag-o-terms " and "
              dag-l-terms
ro-query      = dag-r-terms " and " dag-o-terms
rol-query     = dag-r-terms " and " dag-o-terms " and "
              dag-l-terms
```

The examples given earlier are then translated as follows.

Example 1:

FN=thinking and FN=cat:search=substring;case=ignore<NL>

Example 2:

FN=thinking and FN=cat:search=exact;case=ignore<NL>

Example 3:

ROLE=thinking and ROLE=cat and ORG=space and
ORG=colonization:search=substring;case=ignore<NL>

Querying a DAG-SAP

In querying a DAG-SAP (irrespective of the protocol of that DAG-SAP),
the DAG/IP query must include information about the target WDSP
server.  This information is drawn from the Referral Index SERVER-
TO-ASK referral information, and is appended to the query as
specified in Appendix C):

```
    ":host=" quoted-hostname ";port=" number ";server-info="
    quoted-serverinfo ";charset=" charset
```

    where the response from the Referral Index included:

```
    "# SERVER-TO-ASK " serverhandle nl
    " Server-info: " serverinfo nl
    " Host-Name: " hostname nl
    " Host-Port: " number nl

    " Protocol: " prot nl
    " Source-URI: " source nl
    " Charset: " charset nl
    "# END" nl
```

    and the "quoted-hostname" and "quoted-serverinfo" are obtained from
    "hostname" and "serverinfo" respectively, by quoting the DAG/IP
    special characters.

    For example, the referral

```
    # SERVER-TO-ASK dagsystem01<NL>
     Server-info: o=thinkingcat, c=se<NL>
     Host-Name: thinkingcat.com<NL>
     Host-Port: 2839<NL>
     Protocol: ldapv2<NL>
     Source-URI: http://www.thinkcat.com
     Charset: T.61<NL>
     # END<NL>
```

    would yield the addition

```
    :host=thinkingcat\.com;port=2839;server-info=o\=thinkingcat\,\
    c\=se;charset=T\.61
```

    in its query to an LDAPv2 DAG-SAP.

    (N.B.: See Appendix C for further definitions of the terms used in
    the SERVER-TO-ASK response).

    Note that it is the DAG-SAP's responsibility to extract these terms
    from the query and use them to identify the WDSP server to be
    contacted.  See the individual DAG-SAP definitions, below.

5.5.3 Chaining queries in Mail DAG-CAP

   The Mail DAG-CAP has to chain all referrals -- to the Whois++ DAG-
   SAP, LDAPv2  DAG-SAP, or LDAPv3 DAG-SAP as appropriate for the
   referral.

5.5.4 Expression of results in Mail DAG-CAP

   The results message is sent to the "Reply-To:"  address of the
   originating mail, if available (see [4] for appropriate
   interpretation of mail originator headers).  The original query is
   repeated, along with the message-id.  The remainder of the body of
   the mail message is the concatenation of responses from the DAG-SAP
   calls, each result having the WDSP's SOURCE URI (from the referral)
   appended to it, and the system messages also having been removed.

   At the end of the message, the WDSP servers that failed to respond
   (i.e., the DAG-SAP handling the referral returned the "% 403
   Information Unavailable" message) are listed with their server-info.

5.5.5 Expression of Errors in Mail DAG-CAP

   If the mail DAG-CAP receives a message that is not parsable using the
   query grammar described above, it returns an explanatory message to
   the query mail's reply address saying that the query could not be
   interpreted, and giving a description of valid queries.

   If the number of referrals sent by the Referral Index is greater than
   the pre-determined maximum (for detecting data-mining efforts, or
   otherwise refusing over-general queries, such as "FN=svensson"), the
   mail DAG-CAP will send an explanatory message to the query mail's
   reply address describing the "over-generalized query" problem,
   suggesting the user resubmit a more precise query, and describing the
   list of valid query types.

   If the mail DAG-CAP receives several different result codes from the
   DAG-SAPs it should represent those in an appropriate manner in the
   response message.

   A mail DAG-CAP may redirect a connection to another mail DAG-CAP for
   reasons of load-balancing.  This is done simply by forwarding the
   mail query to the address of the alternate mail DAG-CAP.

5.6 Web (HTTP) DAG-CAP

5.6.1 Web DAG-CAP Input

   The web DAG-CAP provides its interface via standard HTTP protocol.
   The general expectation is that the web DAG-CAP will provide a form
   page with radio buttons to select "substring or exact match" and
   "consider case or ignore case".  Other information (about name, role,
   organization, locality) is solicited as free-form text.

   The DAG-CAP receives queries via an HTTP "post" method (the outcome
   of the form action for the page described above, or generated
   elsewhere).  The rest of this section describes the variables that
   are to be expressed in that post.  The actual layout of the page and
   most user interface issues are left to the discretion of the builder.
   Note that the Web DAG-CAP may be called upon to provide responses in
   different content encoding, and must therefore address the "Accept-
   Encoding:" request header in the HTTP connection.

   Although the Web protocol, HTTP, is not itself capable of handling
   referrals, through the use of two extra variables this client is
   given the option of requesting referral information and then pursuing
   individual referrals through the Web DAG-CAP itself, as a proxy for
   those referrals.  This is handled through the extra "control
   variables" to request referrals only, and to indicate when the
   transaction is a continuation of a previous query to pursue a
   referral.

   There has been call to have a "machine-readable" version of the
   search output.  As HTML is geared towards visual layout, user agents
   that intend to do something with the results other than present them
   in an HTML browser have few cues to use to extract the relevant
   information from the HTML page.  Also, "minor" visual changes,
   accomplished with extensive HTML updates, can disrupt user agents
   that were built to blindly parse the original HTML.  Therefore,
   provision has been made to return "raw" format results.  These are
   requested by specifying "Accept-Content: application/whoispp-
   response"  in the request header of the HTTP message to the HTTP
   DAG-CAP.

    The variables that are expected are:

    transaction       = "new" / "chain"  ; default is "new". This
                        ; should not be user-settable.  It is used
                        ; in constructed URLs
    resulttype        = "all" / "referrals" ; default is "all"
    matchtype         = "substring" / "exact"
    casetype          = "case ignore" / "case sensitive"
    n-term            = string
    o-term            = string
    l-term            = string
    r-term            = string
    host-term         = string
    port-term         = string
    servinfo-term     = string
    prot-term         = string ; the protocol of the referral
    string            = <UNICODE-2-0-UTF-8> / <UNICODE-1-1-UTF-8> /
                        <ISO-8859-1>

5.6.2 Translation from Web query to DAG/IP

    Querying a DAG-SAP Directly

    If the transaction variable is "chain", the information in the POST
    is used to pursue a particular referral, not do a search of the
    Referral Index.  The appropriate DAG-SAP (deduced from the prot-term)
    is contacted and issued the query directly.

    Results from this type of query are always full results (i.e., not
    referrals).

    Querying the Referral Index

    A key element of translating from the Web DAG-CAP input into the
    DAG/IP query format is to "tokenize" the input terms into single
    token elements for the DAG/IP query.  For example,  the n-term

    name= thinking cat

    is tokenized into 2 n-tokens:

    thinking
    cat

    which are then mapped into the following in the DAG/IP query (dag-n-
    terms):

    FN=thinking and FN=cat

   The same is true for the r-term, l-term and o-term.

   The primary steps in translating the HTTP input into a DAG/IP query
   are:

   translate encodings, if necessary
   tokenize the strings for each term
   construct the DAG/IP query from the resulting components, as
   described in more detail below

   DAG/IP constraints are constructed from the searchtype information in
   the query.

```
   dag-matchtype = "search=" <matchtype> /
               "search=substring"     ; if matchtype not
                                       ; specified

   dag-casetype  = "case=ignore"  /        ; if casetype not
                                       ; specified or
                                       ; casetype="case ignore"
               "case=consider"       ; if casetype=
                                       ; "case sensitive"

   constraints   = ":" dag-matchtype ";" dag-casetype
```

   The terms for the DAG/IP query are constructed from the tokenized
   strings from the HTTP post input.

```
   dag-n-terms   = "FN=" n-token 0*( " and FN=" n-token)
   dag-o-terms   = "ORG=" o-token 0*( " and ORG=" o-token)
   dag-l-terms   = "LOC=" l-token 0*( " and LOC=" l-token)
   dag-r-terms   = "ROLE=" r-token 0*( " and ROLE=" r-token)
```

   This means that the relevant DAG/IP queries are formulated as one of
   two types:

```
   dagip-query   = ( ( ( n-query / nl-query / no-query / nol-query )
                   [" and template=DAGPERSON"]":" dag-matchtype
                   ";" dag-casetype) /
                ( ( ro-query / rol-query )
                   [" and template=DAGORGROLE"]":" dag-matchtype
                   ";" dag-casetype)  )

   n-query       = dag-n-terms
```

```
   nl-query       = dag-n-terms " and " dag-l-terms
   no-query       = dag-n-terms " and " dag-o-terms
   nol-query      = dag-n-terms " and " dag-o-terms " and "
               dag-l-terms
   ro-query       = dag-r-terms " and " dag-o-terms
   rol-query      = dag-r-terms " and " dag-o-terms " and "
               dag-l-terms
```

   Querying a DAG-SAP

   In querying a DAG-SAP (irrespective of the protocol of that DAG-SAP),
   the DAG/IP query must include information about the target WDSP
   server.  This information is drawn from the Referral Index SERVER-
   TO-ASK referral information, and is appended to the query as
   specified in Appendix C:

   ":host=" quoted-hostname ";port=" number ";server-info="
   quoted-serverinfo ";charset=" charset

   where the response from the Referral Index included:

   "# SERVER-TO-ASK " serverhandle <NL>
   " Server-info: " serverinfo <NL>
   " Host-Name: " hostname <NL>
   " Host-Port: " number <NL>
   " Protocol: " prot <NL>
   " Source-URI: " source <NL>
   " Charset: " charset <NL>
   "# END" <NL>

   and the "quoted-hostname" and "quoted-serverinfo" are obtained from
   "hostname" and "serverinfo" respectively, by quoting the DAG/IP
   special characters.

   For example, the referral

   # SERVER-TO-ASK dagsystem01<NL>
    Server-info: o=thinkingcat, c=se<NL>
    Host-Name: thinkingcat.com<NL>
    Host-Port: 2839<NL>
    Protocol: ldapv2<NL>
    Source-URI: http://www.thinkingcat.com
    Charset: T.61<NL>
   # END<NL>

   would yield the addition

       :host=thinkingcat\.com;port=2839;server-info=o\=thinkingcat\,\
       c\=se;charset=T\.61

       in its query to an LDAPv2 DAG-SAP

       (N.B.: See Appendix C for further definitions of the terms used in
       the SERVER-TO-ASK response).

       Note that it is the DAG-SAP's responsibility to extract these terms
       from the query and use them to identify the WDSP server to be
       contacted.  See the individual DAG-SAP definitions, below.

5.6.3 Chaining queries in Web DAG-CAP

       If the resulttype was "all", all of the referrals received from the
       Referral Index are chained using the appropriate DAG-SAPs.  If only
       referrals were requested, the Referral Index results are returned.

5.6.4 Expression of results in Web DAG-CAP

       text/html results

       The default response encoding is text/html.  If the resulttype was
       "all",  the content of the chaining responses from the DAG-SAPs,
       without the system messages, is collated into a single page response,
       one result entry per demarcated line ( e.g., bullet item).  The FN or
       ROLE value should be presented first and clearly.  The SOURCE URI for
       each WDSP referral should be presented as an HREF for each of the
       WDSPs results.

       At the end of the message, the WDSP servers that failed to respond
       (i.e., the DAG-SAP handling the referral returned the "% 403
       Information Unavailable" message) are listed with their server-info.

       If, however, the resulttype was "referrals", the results from the
       Referral Index are returned as HREF URLs to the Web DAG-CAP itself,
       with the necessary information to carry out the query (including the
       "HOST=", etc, for the referral).

       For example, if the original query:

       n-term="thinking cat"
       resulttype="referrals"

       drew the following referral from the Referral Index:

       # SERVER-TO-ASK DAG-Serverhandle<NL>
        Server-Info: c=se, o=tce<NL>

```
   Host-Name: answers.tce.com<NL>
   Host-Port: 1111<NL>

   Protocol: ldapv3<NL>
   Source-URI: http://some.service.se/
   Charset: UTF-8<NL>
  # END<NL>
```

   the response would be an HTML page with an HREF HTTP "POST" URL to
   the Web DAG-CAP with the following variables set:

```
  n-term="thinking cat"
  transaction="chain"
  servinfo-term="c=se, o=tce"
  host-term="answers.tce.com"
  port-term="1111"
  prot-term="ldapv3"
```

   The Source-URI should be established in the response as its own HREF
   URI.

   application/whoispp-response Results

   If Accept-Encoding: " HTTP request header had the value
   "application/whoispp-response", the content of the HTTP response will
   be constructed in the same syntax and attribute mapping as for the
   Whois++ DAG-CAP.

   If the resulttype was "all", all the referrals will have been chained
   by the Web DAG-CAP, and the response will include only full data
   records.

   If the resulttype was "referrals", then all referrals are passed
   directly back in a single response, in correct Whois++ referral
   format (conveniently, this is how they are formulated in the DAG/IP).
   Note that this will include referrals to LDAP-based services as well
   as Whois++ servers.

5.6.5 Expression of Errors in Web DAG-CAP

   A Web DAG-CAP may redirect a connection to another web DAG-CAP for
   reasons of load-balancing.  This is done simply by using an HTTP
   redirect.

Standard Errors

If the web DAG-CAP receives a message that is not parsable using the
query grammar described above, it sends an explanatory HTML page
saying that the query could not be interpreted, and giving a
description of valid queries.

If the number of referrals sent by the Referral Index is greater than
the pre-determined maximum (for detecting data-mining efforts, or
otherwise refusing over-general queries, such as "FN=svensson"), the
web DAG-CAP will send a page with an explanatory message describing
the "over-generalized query" problem, suggesting the user resubmit a
more precise query, and describing the list of valid query types.

If the web DAG-CAP receives more than one result code from the DAG-
SAPs, it must represent them all in a appropriate manner in the
response.

application/whoispp-response Errors

An invalid query is responded to with a simple text response with the
error: "% 500 Syntax Error".

If too many referrals are generated from the Referral Index, the
simple text response will have the message "% 503 Query too general".

5.7 Whois++ DAG-CAP

TISDAG: The system commands polled-for/-by should elicit the empty
set as a return value until we better understand the implications
of doing otherwise.

5.7.1 Whois++ DAG-CAP Input

Input to the Whois++ DAG-CAP follows the Whois++ standard ([6]).
Minimally, the Whois++ DAG-CAP must support the following queries:

```
Query Type      Expression in Whois++
-----------     ----------------------------------
N               One or more "name=" and
                template=USER

NL              One or more "name=" and
                One or more "address-locality=" and template=USER

NO              One or more "name=" and
                one or more "organization-name=" and template=USER
```

```
    NOL            One or more "name=" and
                   one or more  "organization-name=" and
                   one or more  "address-locality=" and template=USER

    RO             One or more "org-role=" and
                   one or more "organization-name=" and template=ORGROLE

    ROL            One or more "org-role=" and
                   one or more "organization-name=" and
                   one or more "address-locality=" and template=ORGROLE
```

      Table 5.1 Allowable Whois++ Queries

   The following constraints must be supported for queries:

   "search=" (substring / exact)
   "case=" (ignore / consider)

   If no constraints are defined in a query the default is exact and
   ignore.  For example,

   FN=foo and loc=kista and fn=bar<NL>

   is a perfectly valid Whois++ NL query for "Foo Bar" in "Kista".

5.7.2 Translation from Whois++ query to DAG/IP

   Querying the Referral Index

   The Whois++ DAG-CAP formulates a DAG/IP query by forwarding the
   search terms received (as defined in Table 5.1).

   For example, the above query would be expressed as:

   FN=foo and LOC=kista and FN=bar and template=DAGPERSON<NL>

   Querying a DAG-SAP

   In querying a DAG-SAP (irrespective of the protocol of that DAG-SAP),
   the DAG/IP query must include information about the target WDSP
   server.  This information is drawn from the Referral Index SERVER-
   TO-ASK referral information, and is appended to the query as
   specified in appendix C:

   ":host=" quoted-hostname ";port=" number ";server-info="
   quoted-serverinfo ";charset=" charset

   where the response from the Referral Index included:

   "# SERVER-TO-ASK " serverhandle<NL>
   " Server-info: " serverinfo<NL>
   " Host-Name: " hostname<NL>
   " Host-Port: " number<NL>
   " Protocol: " prot<NL>
   " Source-URI: " source<NL>
   " Charset: " charset<NL>
   "# END"<NL>

   and the "quoted-hostname" and "quoted-serverinfo" are obtained from
   "hostname" and "serverinfo" respectively, by quoting the DAG/IP
   special characters.

   For example, the referral

   # SERVER-TO-ASK dagsystem01<NL>
    Server-info: o=thinkingcat, c=se<NL>
    Host-Name: thinkingcat.com<NL>
    Host-Port: 2839<NL>
    Protocol: ldapv2<NL>
    Source-URI: http://www.thinkingcat.com/
    Charset: T.61<NL>
   # END<NL>

   would yield the addition

   :host=thinkingcat\.com;port=2839;server-info=o\=thinkingcat\,\
   c\=se;charset=T\.61

   in its query to an LDAPv2 DAG-SAP.

   (N.B.: See Appendix C for further definitions of the terms used in
   the SERVER-TO-ASK response).

   Note that it is the DAG-SAP's responsibility to extract these terms
   from the query and use them to identify the WDSP server to be
   contacted.  See the individual DAG-SAP definitions, below.

5.7.3 Chaining in Whois++ DAG-CAP

   The Whois++ DAG-CAP relies on DAG-SAPs to chain any non-Whois++
   referrals (currently, the LDAPv2 and LDAPv3 DAG-SAPs).

5.7.4 Expression of results in Whois++

   Results are expressed in Whois++ by collating the DAG/IP results
   received from DAG-SAPs (using the FULL response), and using the
   template and attribute mappings defined in Appendix B. For each
   result from a given referral, the SOURCE attribute is added, with the
   value of the SOURCE-URI from the referral.

   Any referrals to other Whois++ servers provided by the Referral Index
   are sent directly to the Whois++ client as follows:

   server-to-ask  =   "# SERVER-TO-ASK " DAG-Serverhandle<NL>
                    " Server-Handle: " SERVER-INFO<NL>
                    " Host-Name: " HOST<NL>
                    " Host-Port: " PORT<NL>
                    " Protocol: " PROTOCOL<NL>
                    "# END"<NL>

   where SERVER-INFO, HOST, PORT, PROTOCOL are drawn from the referral
   provided in the DAG/IP, and the SOURCE-URI information is lost.

5.7.5 Expression of Errors in Whois++ DAG-CAP

   As appropriate, the Whois++ DAG-CAP will express operational errors
   following the Whois++ standard.  There are 4 particular error
   conditions of the DAG system that the DAG-CAP will handle as
   described below.

   When the Whois++ DAG-CAP receives a query that it cannot reply to
   within the (data) constraints of the DAG, it sends an error message
   and closes the connection.  The error message includes

   % 502 Search expression too complicated<NL>

   If the number of referrals sent by the Referral Index is greater than
   the pre-determined maximum (for detecting data-mining efforts, or
   otherwise refusing over-general queries, such as "FN=svensson"), the
   Whois++ DAG-CAP will send an error message and close the connection.
   The error message includes

   % 503 Query too general<NL>

   (N.B.: this is different from the "Too many hits" reply, which does
   send partial results.)

   A Whois++ DAG-CAP may redirect a connection to another Whois++ DAG-
   CAP for reasons of load-balancing.  This is expressed to the end-user
   client software using the SERVER-TO-ASK response with appropriate
   information to reach the designated alternate DAG-CAP.

   If a Whois++ DAG-CAP receives several different response codes from
   DAG-SAPs it should try to represent them all in the response to the
   end-user client.

   The proposed mapping between DAG/IP response codes and Whois++
   response codes are given in Appendix D.

5.8 LDAPv2 DAG-CAP

5.8.1 LDAPv2 DAG-CAP Input

   Input to the LDAPv2 DAG-CAP follows the LDAPv2 standard ([19]).
   Minimally, the LDAPv2 DAG-CAP must support the following queries
   (adapted from the ASN.1 grammar of the standard):

   BindRequest ::=
        [APPLICATION 0] SEQUENCE {
                   version   INTEGER (1 .. 127),
                   name      LDAPDN,
                   authentication CHOICE {
                        simple       [0] OCTET STRING,
                        krbv42LDAP   [1] OCTET STRING,
                        krbv42DSA    [2] OCTET STRING
                    }

        }

   BindResponse ::= [APPLICATION 1] LDAPResult

   SearchRequest ::=
    [APPLICATION 3] SEQUENCE {
       baseObject    "dc=se",
       scope         wholeSubtree          (2),
       derefAliases  ENUMERATED {
                   neverDerefAliases    (0),
                   derefInSearching     (1),
                   derefFindingBaseObj  (2),
                   derefAlways          (3)
       },
       sizeLimit     INTEGER (0 .. maxInt),
       timeLimit     INTEGER (0 .. maxInt),
       attrsOnly     BOOLEAN,
       filter        Filter,

```
        attributes      SEQUENCE OF AttributeType
    }

    Filter ::=
     CHOICE {
        and                 [0] SET OF Filter,
        or                  [1] SET OF Filter,
        not                 [2] Filter,
        equalityMatch       [3] AttributeValueAssertion,
        substrings          [4] SubstringFilter
     }

    SubstringFilter ::=
     SEQUENCE {
        type                AttributeType,
        SEQUENCE OF CHOICE {
            initial         [0] LDAPString,
            any             [1] LDAPString,
            final           [2] LDAPString
        }
     }
```

Queries against attributes in the prescribed LDAP standard schema
(see Appendix B) are accepted.

N.B., this is a minimal set of supported queries, to achieve the
basic DAG-defined queries.  An LDAP DAG-CAP may choose to support
more complex queries than this, if it undertakes to do the
translation from the DAG/IP to the LDAPv2 client in a way that
responds to the semantics of those queries.

   TISDAG:  Since LDAPv2 didn't specify any characterset but relied
   on X.500 to do so, in practice several different charactersets are
   in use in Sweden today.  That the LDAPv2 CAP has no way of knowing
   which characterset that are in use by a connecting client is a
   problem that the TISDAG project can not solve.

   Users of the DAG system will have to configure their specific
   client according to information on the TISDAG web page.  That page
   provides very specific information (including port number) that
   can be given to LDAPv2 users.  The LDAP DAG-CAP listening on the
   default port (389) will be the LDAPv3 one.

5.8.2 Translation from LDAPv2 query to DAG/IP

    Querying the Referral Index

    The essential stratagem for mapping LDAP queries into DAG/IP Referral
    Index queries is to tokenize the string-oriented LDAP
    AttributeValueAssertions or SubstringFilters and construct an
    appropriate DAG/IP token-oriented query in the DAG/IP.  This will
    generalize the LDAP query and yield false-positive referrals, but
    should not miss any appropriate referrals.

    There are 3 particular cases to be considered:

    equalityMatch queries
    substring queries
    combination equalityMatch and substring queries

        TISDAG: If the LDAP filter contains a cn-term and no objectclass
        specification it is unclear if the search is for a person or a
        role.  When this happens the DAG query should cover all bases and
        map the query into a query for both people and roles.

    EqualityMatch queries can be handled by simply tokenizing the
    AttributeValueAssertions, making one DAG/IP query term per token
    (using the appropriate DAGSchema attribute) and carrying out an
    exact match in the DAG/IP.

    Consider the following example, represented in the ASCII
    expression of LDAP Filters as described in [13]):

    (& (cn=Foo Bar)(objectclass=inetOrgPerson))

    This query can be represented in the DAG/IP as

    FN="Foo" and FN="Bar":search=exact<NL>

    N.B.
    The search is set up to be "case=ignore" (the DAG/IP's default)
    because the relevant LDAP schema attributes are all derivatives
    of the "name" attribute element, which is defined to have a case
    insensitive match.

    If no objectclass were defined the query in DAG/IP would have
    been

    (FN="Foo" and FN="bar") or (ROLE="Foo" and ROLE="bar"):search=exact

inetOrgPerson is used as the objectclass in this and the following
examples, although person or organizationalPerson could also have
been used.

This query will yield false-positive referrals; the original
LDAP query should only match against records for which the "cn"
attribute is exactly the phrase "Foo Bar", whereas the DAG/IP
query will yield referrals any WDSP containing records that
include the two tokens "foo" and "bar" in any order.

For example, this DAG/IP query will yield referrals to WDSPs
with records including:

cn: Bar Foo
cn: Le Bar Foo
cn: Foo Bar AB

LDAP substring queries must also be tokenized in order to construct a
DAG/IP query.  The additional point to bear in mind is that LDAP
substring expressions are directed at phrases, which obscure
potential token boundaries.  Consequently, all points between
substring components must be considered as potential token
boundaries.

Thus, the LDAP query

(& (cn=black) (o=c*t) (objectclass=inetOrgPerson))

could be expressed as a  DAG/IP query with 3 tokens, in a substring
search:

FN=black and ORG=c and ORG=t:search=substring<NL>

This query will yield false-positive results as the tokenized query
does not preserve the order of appearance in the LDAP substring, and
it doesn't preserve phrase-boundaries.  That is,

ORG=c and ORG=t:search=substring

will match

tabacco

which is not a match by the LDAP query semantics.

Combined EqualityMatch and Substring queries need special attention.
When an LDAP query includes both EqualityMatch components and
substring filter components, the DAG/IP query to the Referral Index

can be constructed by following the same mechanisms of tokenization,
but the whole search will become a substring search, as the DAG/IP
defines only search types across the entire query for Referral Index
queries.

Thus,

(& (cn=Foo Bar) (o=c*t) (objectclass=inetOrgPerson))

can be expressed as

FN=Foo and FN=Bar and ORG=c and ORG=t:search=substring<NL>

Alternatively, the LDAP DAG-CAP could conduct two separate queries
and take the intersection (the logical "AND") of the two sets of
referrals returned by the Referral Index.

Note that DAG/IP can accept phrases for searches -- the query

FN=Foo\ bar<NL>   (note the escaped space)

is perfectly valid.  However, it would match only those things which
have been tokenized in a way that preserves the space, which is the
empty set in the case of the data stored here.

Querying a DAG-SAP

It is never invalid to use the same substantive query to a DAG-SAP as
was used to obtain referral information from the Referral Index.
However, the over-generalization of these queries may yield excessive
numbers of results, and will necessitate some pruning of results in
order to match the returned results against the semantics of the
original LDAP query.  It is the LDAP DAG-CAP that is responsible for
this pruning, as it is the recipient of the original query, and
responsible for responding to its semantics.

In concrete terms, when making the DAG/IP query which is to be sent
to a  DAG-SAP the above mentioned queries are still valid queries,
but an alternative finer-grained query is also possible, namely:

FN=foo and FN=bar and ORG=c;search=lstring and ORG=t;search=tstring

Particularly in the case of the LDAPv2 DAG-CAP, however, there will
be cause to use LDAP(v2/v3) DAG-SAPs.  Since these DAG-SAPs also deal
in phrase-oriented data, a less-over-generalized query can be passed
to them:

FN=Foo\ Bar:search=exact<NL>

In querying a DAG-SAP (irrespective of the protocol of that DAG-SAP),
the DAG/IP query must include information about the target WDSP
server.  This information is drawn from the Referral Index SERVER-
TO-ASK referral information, and is appended to the query as
specified in Appendix C:

":host=" quoted-hostname ";port=" number ";server-info="
quoted-serverinfo ";charset=" charset

where the response from the Referral Index included:

```
"# SERVER-TO-ASK " serverhandle<NL>
" Server-info: " serverinfo<NL>
" Host-Name: " hostname<NL>
" Host-Port: " number<NL>
" Protocol: " prot<NL>
" Source-URI: " source<NL>
" Charset: " charset<NL>
"# END<NL>
```

and the "quoted-hostname" and "quoted-serverinfo" are obtained from
"hostname" and "serverinfo" respectively, by quoting the DAG/IP
special characters.

For example, the referral

```
# SERVER-TO-ASK dagsystem01<NL>
 Server-info: o=thinkingcat, c=se<NL>
 Host-Name: thinkingcat.com<NL>
 Host-Port: 2839<NL>
 Protocol: ldapv2<NL>
 Source-URI: http://www.thinkingcat.com <NL>
 Charset: T.61<NL>
# END<NL>
```

would yield the addition

```
:host=thinkingcat\.com;port=2839;server-info=o\=thinkingcat\,\
c\=se;charset=T\.61
```

in its query to an LDAPv2 DAG-SAP.

(N.B.: See Appendix C for further definitions of the terms used in
the SERVER-TO-ASK response).

Note that it is the DAG-SAP's responsibility to extract these terms
from the query and use them to identify the WDSP server to be
contacted.  See the individual DAG-SAP definitions, below.

5.8.3 Chaining queries in LDAPv2 DAG-CAP

   The LDAPv2 DAG-CAP relies on DAG-SAPs to resolve every referral.

5.8.4 Expression of results in LDAPv2

   As described above, results from DAG-SAPs will have to be post-
   processed in cases where the original query was generalized for
   expression in DAG/IP.

   Acceptable results are expressed in the LDAP search response:

   SearchResponse ::=
    CHOICE {
         entry        [APPLICATION 4] SEQUENCE {
                  objectName    LDAPDN,
                  attributes    SEQUENCE OF SEQUENCE
                            {
                                     AttributeType,
                                     SET OF AttributeValue
                            }
                  },
         resultCode  [APPLICATION 5] LDAPResult
    }

   where

   LDAPDN = DN / "cn=" (FN/ROLE) [",o="ORG] ",dc=se"
   attributes = <all attributes mapped from DAG schema, and
                  "objectClass = inetOrgPerson",
                  "objectClass = top",
                  "objectClass = person" or
                  "objectClass = organizationalRole", as
                  appropriate, and "labeledURI = <SOURCE-URI>"
                  for each result from a given referral>

   (Where DN,FN,ORG and ROLE are the values from the DAG schema).

   I.e., where available, the entry's true DN is used; otherwise (e.g.,
   for data coming from Whois++ servers), a reasonable facsimile is
   constructed.

5.8.5 Expression of Errors in LDAPv2 DAG-CAP

   As appropriate, the LDAPv2 DAG-CAP will express system responses
   following the LDAPv2 standard.

Appendix D gives the proposed mapping between DAG/IP response codes
and LDAPv2 resultcodes.

There are 4 particular error conditions of the DAG system that the
DAG-CAP will handle as described below.

When the LDAPv2 DAG-CAP receives a query that it cannot reply to
within the (data) constraints of the DAG queries, it sends an error
message and closes the connection.  The error message includes the
LDAPv2 resultCode:

```
noSuchAttribute        (for incorrect schema attributes)
inappropriateMatching  (when a match type other than those
                        supported is used, e.g. approxMatch)
unwillingToPerform     (when the query is not one of the
                        defined types)
```

If the number of referrals sent by the Referral Index is greater than
the pre-determined maximum (for detecting data-mining efforts, or
otherwise refusing over-general queries, such as "FN=svensson"), the
LDAPv2 DAG-CAP will send an error message.  The error message
includes one of the following resultCodes:

```
sizeLimitExceeded
timeLimitExceeded
```

An LDAPv2 DAG-CAP may redirect a connection to another LDAPv2 DAG-CAP
for reasons of load-balancing.  This is expressed to the end-user
client software using the "umich referral" convention to direct the
client software to an alternate DAG-CAP by passing the URL in an
error message.

Since a LDAPv2 DAG-CAP only can send one resultcode back to a client;
If a LDAPv2 DAG-CAP receives several different result codes from the
DAG-SAPs it will have to construct a resultmessage that to some
extent represents the combination of those.  It is proposed that in
these cases the following actions are taken:

- All the response codes are collected
- Each response code are translated into the corresponding LDAPv2
  resultcode.
- A resultcode is chosen to represent the collected response on the
  following grounds:
    If  "success" is the only resultcode represented after these
    steps the return that result code.
    If apart from "success" there is one other resultcode represented
    return that other resultcode.

        If apart from "success" there are two or more resultcodes
        represented return the resultcode "other".

5.9 LDAPv3 DAG-CAP

5.9.1 LDAPv3 DAG-CAP Input

   Input to the LDAPv3 DAG-CAP follows the LDAPv3 definition (currently
   defined in [17]).  Minimally, the LDAPv3 DAG-CAP must support the
   following queries (adapted from the ASN.1 grammar of the standard):

   BindRequest ::= [APPLICATION 0] SEQUENCE {

                version                 INTEGER (1 .. 127),
                name                    LDAPDN,
                authentication          AuthenticationChoice }

        AuthenticationChoice ::= CHOICE {
                simple                  [0] OCTET STRING,
                                         -- 1 and 2 reserved
                sasl                    [3] SaslCredentials }

        SaslCredentials ::= SEQUENCE {
                mechanism               LDAPString,
                credentials             OCTET STRING OPTIONAL }

   BindResponse ::= [APPLICATION 1] SEQUENCE {
           COMPONENTS OF LDAPResult,
           serverSaslCreds    [7] OCTET STRING OPTIONAL }

   SearchRequest ::= [APPLICATION 3] SEQUENCE {
        baseObject      c=se,
        scope           wholeSubtree            (2) },
        derefAliases    ENUMERATED {
                neverDerefAliases       (0),
                derefInSearching        (1),
                derefFindingBaseObj     (2),
                derefAlways             (3) },
         sizeLimit       INTEGER (0 .. maxInt),
        timeLimit       INTEGER (0 .. maxInt),
        typesOnly       BOOLEAN,
        filter          Filter,
        attributes      AttributeDescriptionList }

```
Filter ::= CHOICE {
     and               [0] SET OF Filter,
     or                [1] SET OF Filter,
     not               [2] Filter,
     equalityMatch     [3] AttributeValueAssertion,
     substrings        [4] SubstringFilter }

SubstringFilter ::= SEQUENCE {
     type              AttributeDescription,
     -- at least one must be present
     substrings   initial [0] LDAPString,
     substrings   any     [1] LDAPString,
     substrings   final   [2] LDAPString}
```

Queries against attributes in the proscribed LDAP standard schema
(see Appendix B) are accepted.

N.B., this is a minimal set of supported queries, to achieve the
basic DAG-defined queries.  An LDAP DAG-CAP may choose to support
more complex queries than this, if it undertakes to do the
translation from the DAG/IP to the LDAPv3 client in a way that
responds to the semantics of those queries.

5.9.2 Translation from LDAPv3 query to DAG/IP

   Querying the Referral Index

   The essential stratagem for mapping LDAP queries into DAG/IP Referral
   Index queries is to tokenize the string-oriented LDAP
   AttributeValueAssertions or SubstringFilters and construct an
   appropriate DAG/IP token-oriented query in the DAGschema.  This will
   generalize the LDAP query and yield false-positive referrals, but
   should not miss any appropriate referrals.

   There are 3 particular cases to be considered:

   equalityMatch queries
   substring queries
   combination equalityMatch and substring queries

      TISDAG: If the LDAP filter contains a cn-term and no objectclass
      specification it is unclear if the search is for a person or a
      role.  When this happens the DAG query should cover all bases and
      map the query into a query for both people and roles.

EqualityMatch queries can be handled by simply tokenizing the
AttributeValueAssertions, making one DAG/IP query term per token
(using the appropriate DAGSchema attribute) and carrying out an exact
match in the DAG/IP.

Consider the following example, represented in the ASCII expression
of LDAP Filters as described in [13]):

(& (cn=Foo Bar)(objectclass=person))

This query can be represented in the DAG/IP as

FN="Foo" and FN="Bar":search=exact<NL>

N.B.
The search is set up to be "case=ignore" (the DAG/IP's default)
because the relevant LDAP schema attributes are all derivatives of
the "name" attribute element, which is defined to have a case
insensitive match.

If no objectclass where defined the query in DAG/IP would have been

(FN="Foo" and FN="bar") or ( ROLE="Foo" and ROLE="bar"):search=exact

Although person is used as objectclass in this and the following
examples, inetOrgPerson or organizationalPerson could also have been
used.

This query will yield false-positive referrals; the original LDAP
query should only match against records for which the "cn" attribute
is exactly the phrase "Foo Bar", whereas the DAG/IP query will yield
referrals any WDSP containing records that include the two tokens
"foo" and "bar" in any order.

For example, this DAG/IP query will yield referrals to WDSPs with
records including:

cn: Bar Foo
cn: Le Bar Foo
cn: Foo Bar AB

LDAP substring queries must also be tokenized in order to construct a
DAG/IP query.  The additional point to bear in mind is that LDAP
substring expressions are directed at phrases, which obscure
potential token boundaries.  Consequently, all points between
substring components must be considered as potential token
boundaries.

   Thus, the LDAP query

   (& (cn=black) o=c*t) (objectclass=person))

   should be expressed as a DAG/IP query with 3 tokens, in a substring
   search:

   FN=black and ORG=c and ORG=t:search=substring<NL>

   This query will yield false-positive results as the tokenized query
   does not preserve the order of appearance in the LDAP substring, and
   it doesn't preserve phrase-boundaries.  That is,

   ORG=c and ORG=t:search=substring

   will match

   tabacco

   which is not a match by the LDAP query semantics.

   Combined EqualityMatch and Substring queries need special attention.
   When an LDAP query includes both EqualityMatch components and
   substring filter components, the DAG/IP query to the Referral Index
   can be constructed by following the same mechanisms of tokenization,
   but the whole search will become a substring search, as the DAG/IP
   defines search types across the entire query.

   Thus,

   (& (cn=Foo Bar) (o=c*t) (objectclass=person))

   can be expressed as

   FN=Foo and FN=Bar and ORG=c and ORG=t:search=substring<NL>

   Alternatively, the LDAP DAG-CAP could conduct two separate queries
   and take the intersection (the logical "AND") of the two sets of
   referrals returned by the Referral Index.

   Note that DAG/IP can accept phrases for searches -- the query

   FN=Foo\ bar<NL>    (note the escaped space)

   is perfectly valid.  However, it would match only those things which
   have been tokenized in a way that preserves the space, which is the
   empty set in the case of the data stored here.

Querying a DAG-SAP

It is never invalid to use the same substantive query to a DAG-SAP as
was used to obtain referral information from the Referral Index.
However, the over-generalization of these queries may yield excessive
numbers of results, and will necessitate some pruning of results in
order to match the returned results against the semantics of the
original LDAP query.  It is the LDAP DAG-CAP that is responsible for
this pruning, as it is the recipient of the original query, and
responsible for responding to its semantics.

In concrete terms, when making the DAG/IP query which is to be sent
to a  DAG-SAP the above mentioned queries are still valid queries,
but an alternative finer-grained query is also possible, namely:

FN=foo and FN=bar and ORG=c;search=lstring and ORG=t;search=tstring

In querying a DAG-SAP (irrespective of the protocol of that DAG-SAP),
the DAG/IP query must include information about the target WDSP
server.  This information is drawn from the Referral Index SERVER-
TO-ASK referral information, and is appended to the query as
specified in Appendix C):

"host=" quoted-hostname ";port=" number ";server-info="
quoted-serverinfo ";charset=" charset

where the response from the Referral Index included:
"# SERVER-TO-ASK " serverhandle <NL>
" Server-info: " serverinfo<NL>
" Host-Name: " hostname<NL>
" Host-Port: " number<NL>
" Protocol: " prot<NL>
" Source-URI: " source<NL>
" Charset: " charset<NL>
"# END"<NL>

and the "quoted-hostname" and "quoted-serverinfo" are obtained from
"hostname" and "serverinfo" respectively, by quoting the DAG/IP
special characters.

For example, the referral

# SERVER-TO-ASK dagsystem01<NL>
 Server-info: o=thinkingcat, c=se<NL>
 Host-Name: thinkingcat.com<NL>
 Host-Port: 2839<NL>
 Protocol: ldapv2<NL>
 Source-URI:http://www-thinkingcat.se/

```
 Charset: T.61<NL>
# END<NL>
```

would yield the addition

```
:host=thinkingcat\.com;port=2839;server-info=o\=thinkingcat\,\
c\=se;charset=T\.61
```

in its query to an LDAPv2 DAG-SAP.

(N.B.: See Appendix C for further definitions of the terms used in the SERVER-TO-ASK response).

Note that it is the DAG-SAP's responsibility to extract these terms from the query and use them to identify the WDSP server to be contacted.  See the individual DAG-SAP definitions, below.

5.9.3 Chaining queries in LDAPv3 DAG-CAP

The LDAPv3 DAG-CAP relies on DAG-SAPs to resolve all referrals except those to LDAPv3 servers (i.e., Whois++ referrals, currently).

5.9.4 Expression of results in LDAPv3

As described above, results from DAG-SAPs will have to be post-processed in cases where the original query was generalized for expression in DAG/IP.  Acceptable results are expressed in LDAPv3 messages containing search result entries (see the standard for more detail):

```
SearchResultEntry ::= [APPLICATION 4] SEQUENCE {
     objectName      LDAPDN,
     attributes      PartialAttributeList }

PartialAttributeList ::= SEQUENCE OF SEQUENCE {
     type    AttributeDescription,
     vals    SET OF AttributeValue }

SearchResultReference ::= [APPLICATION 19] SEQUENCE OF LDAPURL
-- at least one LDAPURL element must be present

SearchResultDone ::= [APPLICATION 5] LDAPResult
```

where

```
LDAPDN = DN / "cn=" (FN/ROLE) [",o=" ORG] ",dc=se"
```

```
attributes = <all attributes mapped from the DAG schema, and
               "objectClass = inetOrgPerson",
               "objectClass = person",
               "objectClass = top" or
               "objectClass = organizationalRole", as
               appropriate, and "labeledURI = <SOURCE-URI>"
               for each result from a given referral>
LDAPResult = success
```

(Where DN, FN, ROLE, and ORG are the values from the DAG schema).

I.e., where available, the entry's true DN is used; otherwise (e.g., for data coming from Whois++ servers), a reasonable facsimile is constructed.

Referral URLs are constructed from the DAG/IP's SERVER-TO-ASK information as follows:

```
refurl = "ldap://" HOST [":" PORT] "/" (SERVER-INFO / "dc=se")
```

The intention is that WDSPs using LDAPv3 servers will provide an appropriate LDAPDN for their server in the SERVER-INFO.  Clients are then expected to repeat their query at the server designated by this URL (i.e., the refURL does not include the query).

5.9.5 Expression of Errors in LDAPv3 DAG-CAP

As appropriate, the LDAPv3 DAG-CAP will express operational errors following the LDAPv3 standard.  There are 4 particular error conditions of the DAG system that the DAG-CAP will handle as described below.

When the LDAPv3 DAG-CAP receives a query that it cannot reply to within the (data) constraints of the DAG queries, it sends an error message and closes the connection.  The error message includes the LDAPv3 resultCode

```
noSuchAttribute        (for incorrect schema attributes chosen)
inappropriateMatching  (when a match type other than those
supported is used e.g., approxMatch)
unwillingToPerform     (when the query is not one of the defined
types)
```

If the number of referrals sent by the Referral Index is greater than the pre-determined maximum (for detecting data-mining efforts, or otherwise refusing over-general queries, such as "FN=svensson"), the LDAPv3 DAG-CAP will send an error message.  The error message includes the following resultCode:

   adminLimitExceeded

   An LDAPv3 DAG-CAP may redirect a connection to another LDAPv3 DAG-CAP
   for reasons of load-balancing.  In this case, the LDAPv3 DAG-CAP
   sends a result message including only

   SearchResultReference ::= [APPLICATION 19]  AltURL

   SearchResultDone ::= referral

   where

   AltURL = "ldap://" <althostport> ":" <altbase>

   Since a LDAPv3 DAG-CAP only can send one resultcode back to a client;
   If a LDAPv3 DAG-CAP receives several different result codes from the
   DAG-SAPs it will have to construct a resultmessage that to some
   extent represents the combination of those.  It is proposed that in
   these cases the following actions are taken:

   - All the response codes are collected
   - Each response code are translated into the corresponding LDAPv3
     resultcode.
   - A resultcode is chosen to represent the collected response on the
     following grounds:
        If "success" is the only resultcode represented after these steps
        the return that result code.
        If apart from "success" there is one other resultcode represented
        return that other resultcode.
        If apart from "success" there are two or more resultcodes
        represented return the resultcode "other".

5.10 Whois++ DAG-SAP

5.10.1 Input

   The Whois++ DAG-SAP expects valid DAG/IP communications.  Queries
   must include referral information (see below) and search terms that
   conform to the DAG-allowed query types (e.g., not searches for
   organization alone, etc).

   The referral information is added to the end of the DAG-SAP query, as
   defined in the DAG-CAP definition sections:

   ":host=" quoted-hostname ";port=" number ";server-info="
   quoted-serverinfo ";charset=" charset

5.10.2 Translation from DAG/IP to Whois++ query

    The HOST and PORT information are used to make a TCP/IP-based
    connection to the remote (presumed) Whois++ server.  The query
    expressed to the remote Whois++ server is the remainder of the DAG/IP
    query the Whois++ DAG-SAP received, with the following template ID
    translations:

    template=DAGPERSON becomes template=USER

    and

    template=DAGROLE becomes template=ORGROLE

    Additional mappings for attributes are defined in Appendix B.

    Note that the search types used in the DAG/IP are not all required by
    the Whois++ syntax.  Therefore, some Whois++ WDSPs may be using
    servers that do not support searches other than "exact" and "lstring"
    (the search types required by the Whois++ protocol standard).  The
    Whois++ DAG-CAP may

    - send the DAG/IP query as constructed (e.g., with
      "search=substring"), and pass back the "% 502 Search expression too
      complicated" from the WDSP's server,
      - translate the DAG/IP query into a construct using only these
      search types (which will yield incomplete results, as not all
      queries are expressible with those search types),
      - attempt to ascertain what search types are  supported by the
      remote server and reformulate using them (e.g., regular
      expressions).  This would work, but would entail an excessively
      complicated Whois++ DAG-SAP, and might not yield any better results
      if the remote server doesn't support any optional search types.

5.10.3 Translation of Whois++ results to DAG/IP

    Any referrals that the remote WDSP server returns are pursued,
    following the usual Whois++ (client) fashion, by the Whois++ DAG-SAP.

    If it is not possible to establish a Whois++ session with the remote
    server, or if the session is interrupted, before results are
    received, the DAG-SAP will itself return no results and an error
    message, including

    % 403 Information Unavailable<NL>

If the remote server issues any other Whois++ error message and does
not yield any results, the remote server's error message will be
included in the DAG-SAP's own error message; no results will be
returned.

If results are successfully received from the remote server, they
will be expressed using the  DAG/IP -- essentially passing through
all FULL response information received from the remote server, mapped
into the DAGSchema using the mappings defined in Appendix A.

5.11 LDAPv2 DAG-SAP

5.11.1 Input

The LDAPv2 DAG-SAP expects valid DAG/IP communications.  Queries must
include referral information (see below) and search terms that
conform to the DAG-allowed query types (e.g., not searches for
organization alone, etc).

The referral information is added to the end of the DAG-SAP query, as
defined in the DAG-CAP definition sections (as additional terms in
the DAG/IP query):

":host=" quoted-hostname ";port=" number ";server-info="
quoted-serverinfo ";charset=" charset

5.11.2 Translation from DAG/IP to LDAPv2 query

The HOST and PORT information are used to make a TCP/IP-based
connection to the remote (presumed) LDAPv2 server.  The DAG-SAP will
establish a connection with the remote server, following standard
LDAPv2 message exchanges.

The search request itself will be constructed from the DAG/IP query
(without the HOST, SERVER-INFO and PORT terms) as follows:

```
SearchRequest ::=
 [APPLICATION 3] SEQUENCE {
     baseObject    LDAPDN,  -- from the DAG/IP query
     scope         baseObject            (0) },
     derefAliases  ENUMERATED {
                         neverDerefAliases   (0),
                         derefInSearching    (1),
                         derefFindingBaseObj (2),
                         derefAlways         (3)

                   },
     sizeLimit     INTEGER (0 .. maxInt),
```

```
      timeLimit       INTEGER (0 .. maxInt),
      attrsOnly       FALSE
      filter          Filter,
      attributes      SEQUENCE OF AttributeType
                            -- all DAGschema attributes
                               equivalents in  the defined
                               standard LDAP schema
   }

   Filter ::=
    CHOICE {
      and                 [0] SET OF Filter,
      or                  [1] SET OF Filter,
      not                 [2] Filter,
      substrings          [4] SubstringFilter,
    }

   SubstringFilter
    SEQUENCE {
      type                AttributeType,

      SEQUENCE OF CHOICE {
      substrings   initial [0] LDAPString,
      substrings   any     [1] LDAPString,
      substrings   final   [2] LDAPString}
    }
```

   where and, or and not filters are constructed to preserve the logic
   of the DAG/IP query.

   For the purposes of matching token-based DAG/IP queries to reasonable
   LDAP queries, all searches should be passed to the LDAP WDSP as
   substring searches.  The WDSP results must then be pruned to respect
   token boundaries, where necessary.

   So, for example,  the DAG/IP query

   FN=Foo\ Bar and ORG=Thinking\ Cat:search=substring<NL>

   would be sent to the designated LDAP WDSP as

   (& (fn=*Foo Bar*) (o=*Thinking Cat*) (objectclass=person))

   Interestingly, the query

   FN=Foo\ Bar and ORG=Thinking\ Cat:search=exact<NL>

   would also be sent to the designated LDAP WDSP as

```
(& (fn=*Foo Bar*) (o=*Thinking Cat*) (objectclass=person))
```

but the WDSPs returned results would have to be pruned to remove any
results that had non-tokenizing characters on either side of "Foo
Bar" and "Thinking Cat".

The final consideration for mapping DAG/IP queries into LDAP queries
is the issue of character case.  In LDAP, individual attribute
syntaxes define the consideration of case.  All of the attributes
used here are case-insensitive in their definitions.  Therefore, all
LDAP WDSP queries are inherently case-insensitive; if the DAG/IP
query calls for a case-sensitive match, the LDAP DAG-SAP will have to
do pruning of the results from the DAG-SAP.

5.11.3 Translation of LDAPv2 results to DAG/IP

If it is not possible to establish an LDAPv2 session with the remote
server, or if the session is interrupted before results are received,
or if the remote server issues any kind of error message and produces
no result, the DAG-SAP will itself return no results and an error
message, including

% 403 Information Unavailable<NL>

If results are successfully received from the remote server, the
attributes and values that are provided for each result message will
be incorporated into the DAG/IP result, according to the schema
mappings laid out in Appendix B.

One particular adjustment must be done to accommodate differences
between LDAP and the DAG/IP.  The attributes on which searches are
keyed ("cn", "l", and "o" in the LDAP schemas) are all defined as
being case-insensitive for equality matching.  Thus, if the DAG/IP
query includes the constraint "case=consider", the results from the
remote server must be post-processed to remove any wrong-cased ones.

   TISDAG: The serverhandle and localhandle in the DAG/IP response
   should be constructed as follows:

serverhandle is: <hostname-without-periods><port> (because
   server DN's are not enforceably unique).  E.g., a
   services.bunyip.com server on 7778 would
   become servicesbunyipcom7778.
  localhandle is:  the RDN (relative distinguished name), with
   spaces replaced by "_".  E.g., cn=leslie_daigle

5.12 LDAPv3 DAG-SAP

5.12.1 Input

   The LDAPv3 DAG-SAP expects valid DAG/IP communications.  Queries must
   include referral information (see below) and search terms that
   conform to the DAG-allowed query types (e.g., not searches for
   organization alone, etc).

   The referral information is added to the end of the DAG-SAP query, as
   defined in the DAG-CAP definition sections:

   ":host=" quoted-hostname ";port=" number ";server-info="
   quoted-serverinfo ";charset=" charset

5.12.2 Translation from DAG/IP to LDAPv3 query

   The HOST and PORT information are used to make a TCP/IP-based
   connection to the remote (presumed) LDAPv3 server.  The DAG-SAP will
   establish a connection with the remote server, following standard
   LDAPv3 message exchanges.

   The search request itself will be constructed from the DAG/IP query
   (without the HOST, SERVER-INFO and PORT terms) as follows:

```
   SearchRequest ::=
    [APPLICATION 3] SEQUENCE {
        baseObject    LDAPDN,  -- from the DAG/IP query
        scope         baseObject            (0) },
        derefAliases  ENUMERATED {
                            neverDerefAliases    (0),
                            derefInSearching     (1),
                            derefFindingBaseObj  (2),
                            derefAlways          (3)
                        },
        sizeLimit     INTEGER (0 .. maxInt),
        timeLimit     INTEGER (0 .. maxInt),
        attrsOnly     FALSE
        filter        Filter,
        attributes    SEQUENCE OF AttributeType
                      -- all DAGschema attributes equivalents in
                         the defined standard LDAP schema
    }

    Filter ::=
     CHOICE {
        and               [0] SET OF Filter,
        or                [1] SET OF Filter,
```

```
    not                 [2] Filter,
    substrings          [4] SubstringFilter,
 }

SubstringFilter
 SEQUENCE {
    type                AttributeType,
    SEQUENCE OF CHOICE {
    substrings   initial [0] LDAPString,
    substrings   any     [1] LDAPString,
    substrings   final   [2] LDAPString}
 }
```

where and, or and not filters are constructed to preserve the logic
of the DAG/IP query.

For the purposes of matching token-based DAG/IP queries to reasonable
LDAP queries, all searches should be passed to the LDAP WDSP as
substring searches.  The WDSP results must then be pruned to respect
token boundaries, where necessary.

So, for example,  the DAG/IP query

FN=Foo\ Bar and ORG=Thinking\ Cat:search=substring<NL>

would be sent to the designated LDAP WDSP as

(&(fn=*Foo Bar*)(o=*Thinking Cat*)(objectClass=person))

Interestingly, the query

FN=Foo\ Bar and ORG=Thinking\ Cat:search=exact<NL>

would also be sent to the designated LDAP WDSP as

(&(fn=*Foo Bar*)(o=*Thinking Cat*)(objectClass=person))

but the WDSP's returned results would have to be pruned to remove any
results that had non-tokenizing characters on either side of "Foo
Bar" and "Thinking Cat".

The final consideration for mapping DAG/IP queries into LDAP queries
is the issue of character case.  In LDAP, individual attribute
syntaxes define the consideration of case.  All of the attributes
used here are case-insensitive in their definitions.  Therefore, all
LDAP WDSP queries are inherently case-insensitive; if the DAG/IP
query calls for a case-sensitive match, the LDAP DAG-SAP will have to
do pruning of the results from the DAG-SAP.

5.12.3 Translation of LDAPv3 results to DAG/IP

   Any referrals that the remote WDSP server returns are pursued,
   following the usual LDAPv3 (client) fashion, by the LDAPv3 DAG-SAP.

   If it is not possible to establish an LDAPv3 session with the remote
   server, or if the session is interrupted before results are received,
   or if the remote server issues any kind of error message and produces
   no result, the DAG-SAP will itself return no results and an error
   message, including

   % 403 Information Unavailable<NL>

   If results are successfully received from the remote server, the
   attributes and values that are provided for each result message will
   be incorporated into the DAG/IP result, which will be expressed using
   the DAG/IP and schema mappings as outlined in Appendix A.

   One particular adjustment must be done to accommodate differences
   between LDAP and the DAG/IP.  The attributes on which searches are
   keyed ("cn", "l", and "o" in the LDAP schemas) are all defined as
   being case-insensitive for equality matching.  Thus, if the DAG/IP
   query includes the constraint "case=consider", the results from the
   remote server must be post-processed to remove any wrong-cased ones.

      TISDAG: The serverhandle and localhandle in the DAG/IP response
      should be constructed as follows:

      - serverhandle is: <hostname-without-periods><port> (because
        server DN's are not enforceably unique).  E.g., a
        services.bunyip.com server on 7778 would become
        servicesbunyipcom7778.
      - localhandle is:  the RDN (relative distinguished name), with
        spaces replaced by "_".  E.g., cn=leslie_daigle

5.13 Example Queries

   The following sample end-user queries illustrate some of the more
   delicate steps of query/schema semantics translations in the DAG
   system.

   N.B.:  the data presented in these examples is often senseless,
   provided only to serve as illustrations of matching on word-ordering,
   case sensitivity, etc.

5.13.1 A Whois++ Query

   What the Whois++ DAG-CAP Receives

   In this example, the Whois++ DAG-CAP receives the following query:

   name=thinking and name=cat:search=exact;case=consider<NL>

   The expected answer can be described as:

   Any USER templates that contain the tokens "thinking" and "cat" in a
   name attribute.

   For example:

   Different records:

   name: the thinking cat
   name: sublime cat thinking

   or a single record with 2 or more name attributes

   name: thinking felines
   name: erudite cat

   but not

   name: Thinking Cat Enterprises

   This last record would not match because the query called for case
   sensitivity, and the case of the name attribute's value does not
   match the query.

   What the Whois++ DAG-CAP sends to the Referral Index

   After schema translation, this is sent to the Referral Index as:

   fn=thinking and fn=cat:search=exact<NL>

   What the Whois++ DAG-CAP Sends to an LDAP DAG-SAP

   Note that the Whois++ DAG-CAP will never interact with a Whois++
   DAG-SAP as the Whois++ referrals returned by the Referral Index are
   passed directly back to the Whois++ client.

   The Whois++ DAG-CAP should send the same substantive query to the
   DAG-SAP as it sent to the Referral Index, except that it can include
   the case sensitivity constraint:

       fn=thinking and fn=cat:search=exact;case=consider<NL>

       which will be translated by the DAG-SAP into an LDAP query of the
       form:

       (&(cn=*thinking*)(cn=*cat*)(objectclass=inetOrgPerson))

       which will match a record with:

       cn: Thinking
       cn: Cat

       (i.e., 2 different cn attributes, with the 2 values; LDAP defines
       case sensitivity matching by the schema attribute definition).

       or a record with:

       cn: I wish I had a thinking dog and a singing cat

       The first record should be pruned by the LDAP DAG-SAP, in order to
       respect the semantics of the DAG/IP query.

5.13.2 An LDAP Query

       What the LDAP DAG-CAP Receives

       In this example, the LDAP  DAG-CAP receives the following query
       (using RFC1960 notation):

       (& (cn=th*c*t) (o=green groceries) (objectClass=person))

       What the LDAP user is looking for, with this query, is all records
       within the "green groceries" organization that have a cn attribute
       starting with "th", ending with "t", and having a "c" somewhere in
       the middle.

       cn values that would match this include:

       cn: thinkingcat
       cn: Thinking Cat
       cn: The Black Cat
       cn: Thick Mat

5.13.3 What the LDAP DAG-CAP sends to the Referral Index

   The LDAP DAG-CAP must formulate a token-based query to the Referral
   Index that will not inadvertently exclude records that would match.
   The first challenge lies in the fact that the "*" characters in the
   LDAP string-based query can cover token-boundaries.

   A suitable query to the Referral Index would be:

   FN=th AND FN=C AND FN=T AND ORG=green AND
   ORG=groceries:search=substring<NL>

   This will generate some false positive referrals, directing the query
   to WDSPs containing records with the following attribute values (the
   match letters are in capitals for ease of identification):

   cn: wiTH three blaCk poTs

   o: peaGREEN and cyan GROCERIES
   o: GROCERIES are GREENer than electronics

   Alternative approaches include breaking the original query into
   several queries to the referral index in such a way that the DAG-CAP
   can use only those referrals that appear in all the Referral Index
   responses.  However, this is

   overkill -- the purpose of the Referral Index is to give direction on
     where there may be more information

   difficult to code into the DAG-CAP in a general way -- it has to
     identify, by LDAP query type, when and how to do so

   likely to generate Referral Index queries that are complex and time-
   consuming to process.

   What the LDAP DAG-CAP Sends to a Whois++ DAG-SAP

   The LDAP DAG-CAP may send the same query to a Whois++ DAG-SAP as it
   sent to the Referral Index.  False positives here mean results that
   are not expected as a match by the LDAP client.  The LDAP DAG-CAP
   should prune these results from the information returned by the
   Whois++ DAG-SAP.

   Or it might rewrite the query into:

   FN=th;search=lstring AND FN=C;search=substring AND
   FN=T;search=tstring AND ORG=green AND ORG=groceries:case=ignore<NL>

   What the LDAP DAG-CAP Sends to an LDAP DAG-SAP

   As an architectural principle, it is never wrong to send the same
   query to a DAG-SAP as was formulated for the Referral Index.  It is
   also noteworthy to keep in memory that all DAG-SAPs are handled equal
   by all DAG-CAPs therefore a LDAP DAG-CAP will not need to send a
   different query to a LDAP DAG-SAP then it would to any other DAG-SAP.

   So in this case the LDAP DAP-CAP could either send the same query to
   the LDAP DAG-SAP as it sent to the Referral Index or it could send
   the augmented version that is allowed to be use with the DAG-SAPs,
   namely:

   FN=th;search=lstring AND FN=C;search=substring AND
   FN=T;search=tstring AND ORG=green\ groceries:case=ignore<NL>

   Note that this will be translated, by the LDAP DAG-SAP, into a query
   of the form

   (&(cn=*th*)(cn=*c*)(cn=*t*)(o=*green groceries*)
   (objectClass=person))

   which is still more general than the original query.

   Note the translation from "FN=th;search=lstring" into "cn=*th*".
   This is necessary, as the DAG/IP lstring constraint is based on
   tokens, whereas "cn=th*" refers to the beginning of the attribute's
   value (phrase, not token).  The DAG-SAP should therefore prune out
   any results that include things like "oTHer plaCes for visiTors" in
   order to match the semantics of the DAG/IP query it received.

   The DAG-CAP should then prune those results to match the semantics of
   the original LDAP query.

6.0 Service Specifications

6.1 Overview

   To satisfy the requirements laid out for the TISDAG project, the
   software built for the DAG system must be able to meet the following
   service specifications:

   - primary designated DAG-CAPs of all types (but not necessarily
     secondary ones set up for load-balancing) must be available to
     provide service or redirect queries on a 7x24 basis.

    - in general, responses to queries should be available in under 10
      seconds; very generalized queries (i.e., when the user truly cannot
      specify enough information to focus the search) can be deferred to
      take much longer (having results is more important than having a
      quick answer)
    - the data provided from each WDSP should be updated in the DAG at
      least once every 7 days

6.2 WDSP Participation

   WDSPs who wish to participate in the DAG system do so by providing
   DAG-compatible access to their service, where DAG-compatible means:

   - access in (exactly) one of LDAPv2, LDAPv3, or Whois++
     - 7x24 service for responding to referrals generated in the DAG
     core (minimally) weekly updates of the index object describing the
     information their service indexes
     - use of USER and ROLE templates for Whois++ servers
     - use of inetorgperson and organizationalrole objectclasses for
     LDAP servers

   To participate, WDSPs must register each DAG-compliant server with
   the DAG system, providing details for each data set that it covers:

   - the host, port and protocol of the server
     - an identifier for the dataset
     - a URL for the service of preference for accessing the data
     (preferred source)
     - protocol-specific information
     - administrative contact information
     - CIP object exchange information

   Note that any WDSP wishing to make data available through the DAG
   system but unable to support these requirements may provide
   information through an agreement with a third-party which does meet
   these requirements.  Thus, data can be replicated between cooperating
   WDSPs.  The DAG referral index does not claim ownership of personal
   information; it directs queries to services that do, by whatever
   agreements with whichever relevant parties.  Note that, in this case,
   the SOURCE-URI may direct end-users to the WDSP's existing services,
   not the service of the third party.

6.3 Load Distribution

   It is anticipated that the DAG system will be quite popular, and
   measures must be available to distribute the load of answering
   queries.

The DAG system is presented as a conceptual whole, made up of several
component parts -- DAG-CAPs, DAG-SAPs and the Referral Index.  Each
of these component parts must be replicable, and service must be
shared between replicas.

It may be interesting to consider allowing large-scale service
providers (large companies, ISPs)  the ability to mirror the Referral
Index or provide alternate DAG-CAPs/DAG-SAPs for their
personnel/customers.  Policies and possibilities for doing that are
beyond the scope of this report; however, the software architecture
has been designed to support such activity.

Figure 6.1 shows that individual components of the DAG system may
each run on non-co-located server hardware, connected by TCP/IP
networks.  These components can be replicated as needed.

```
+====+
|    |   DAG-CAP (Client Access Point)
|    |
+====+
+----+
|    |   DAG-SAP (Service Access Point)
|    |
+----+
            +====+
HTTP   <-->|    |
           |    |                      +----+
            +====+                     |    |<--> Whois++
                                       |    |
            +====+                     +----+
  SMTP   <-->|    |
            |    |             +----+
            +====+             |    |<--> LDAPv2
                               |    |
            +====+             +----+
  Whois++<-->|    |
            |    |
            +====+                +----+
                                  |    |<--> LDAPv3
                                  |    |
                                  +----+
                                  |    |<--> LDAPv3
                                  |    |
                                  +----+
                                  |    |<--> LDAPv3
            +====+                |    |
  LDAPv2 <-->|    |               +----+
            |    |
            +====+
            +====+
  LDAPv3 <-->|    |
            |    |
            +====+
            +----------------------+
            | Referral Index       |<--> Common Indexing Protocol
            |                      |      (CIP)
            +----------------------+
      +-----------------------+
      | Referral Index        |
      |                       |
      +-----------------------+
```
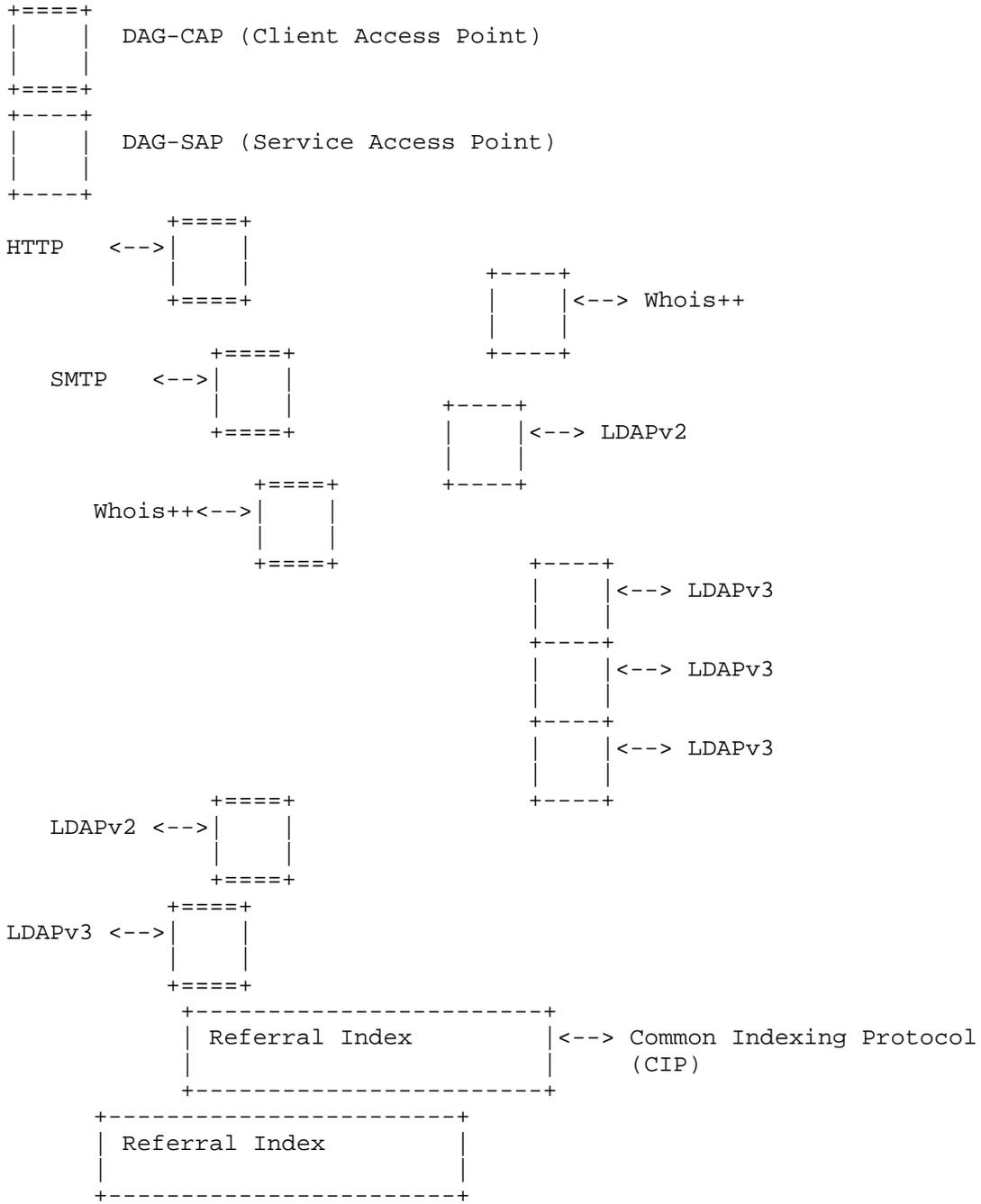
Figure 6.1 Distributable nature of DAG components

   Thus, the software built to this specification must be configurable
   to permit the following actions:

   - DAG-CAP software must be able to handle or redistribute the primary
     load.  Depending on the DAG-CAP software, this may be handled by
     having multiple processes attending to incoming queries, or the
     DAG-CAP at the primary address for the protocol may be nothing more
     than a reflector that redirects incoming queries to the address of
     the least-loaded server at the moment.
   - This is particularly necessary in synchronous connection protocols,
     such as Whois++ and LDAP, where the goal is to minimize the amount
     of time a requesting client is connected to the well-advertised
     address port.
   - DAG-CAP software must be able to direct referrals to different
     DAG-SAPs of the same protocol type.
   - DAG-CAP software must be able to detect overly general queries
     (i.e., have some metric to decide that the number of referrals
     generated by the Referral Index is too great).
   - DAG-SAPs must be able to redirect DAG-CAP queries at their
     discretion, or just refuse service because of loading (therefore
     DAG-CAPs must also be able to find other DAG-SAPs)

6.4 Extensibility

   The DAG system has been designed to allow for extensibility in
   certain key areas:

   It is possible to add new DAG-CAPs and DAG-SAPs transparently.
   Beyond replicating the software of existing DAG-CAPs, new
   implementations for particular protocols (e.g., building a more
   elaborate mail-based query system), or implementations for altogether
   different protocols (e.g., PH) can be added by adhering to the basic
   principles of DAG-CAPs and DAG-SAPs defined in the software
   specification.  The new DAG-CAP is responsible for the translation of
   queries into DAG/IP (post-processing results, if necessary) and
   results in the new protocol.  No other part of the DAG system is
   affected.

   More functionality may be added to the DAG system service (e.g.,
   adding security certificate references to the schema of returned
   information) by updating the DAG schema.

   Depending on how the load on the service goes, it may be interesting
   to consider reducing the number of queries that are chained for
   protocols that inherently can handle the concept of pursuing
   referrals.  Specifically, LDAPv3 and Whois++ both handle referrals,
   but the current system calls for chaining LDAPv3 (and LDAPv2)
   referrals for the Whois++ DAG-CAP, and vice versa.  Alternatively,

   "virtual" DAG-CAPs could be established for each participating WDSP
   for each protocol the WDSP doesn't support, and referrals to those
   DAG-CAPs could be given to the calling client.  For example, a
   Whois++ client would be given a Whois++ referral to the virtual
   Whois++ DAG-CAP for a WDSP that supports only LDAP.  The importance
   of having one virtual DAG-CAP per WDSP is that the point of
   connection is the only way to distinguish which WDSP the Whois++
   client thought it was connecting to.

7.0 Security

7.1 Information credibility

   Security, in the context of "read-only" directory services, is
   primarily concerned with maintaining data integrity as it passes from
   an originating server to the end-user making an inquiry.  That is,
   some server(s) hold correct user information, and a client accessing
   a directory service should be certain that whichever servers that the
   information has to pass through before reaching the client, it
   receives a true representation of the original information.

   The DAG system as such MUST be completely invisible as the mediator
   of the information from the WDSPs to the querying directory access
   client.  The only possible modifications that can appear is
   translations from one characterset into another.  Hopefully, this
   does not alter the meaning of the information.

7.2 Unauthorized access

   In keeping with the public nature of the proposed TISDAG service, the
   DAG system does not provide any access control system beyond
   components' configuration to accept connections from recognized other
   components.  For more detailed access control, it is up to the
   connected WDSPs to apply the access control.

   Since the DAG system only supports searching and retrieving
   information, no updates can occur through the DAG client access
   points.

   Security in updates (CIP index objects) is provided by encryption and
   signature of objects from registered WDSPs.

8.0 Acknowledgments

Appendix A - DAG Schema Definitions

   The DAG makes use of 2 information schemas -- the DAGPERSON schema
   for information about specific people, and the DAGORGROLE schema for
   organizational roles that may or may not be job positions occupied by
   people at any given time (e.g., an organization's president, customer
   service desk, etc).

   This appendix defines the schemas in terms of the attributes used
   within the DAG/IP.  Mappings to the standard LDAP and Whois++ object
   classes and templates (respectively) are described in Appendix B.

   Because the role of the DAG schemas is to act as an intermediary
   between information provided in different access protocols, with
   different underlying schema paradigms, the attributes in the schema
   are identified as being required or optional.  The required
   attributes are so designated because they are involved in the DAG
   search types and/or the minimal returned response.  They have defined
   mappings in the selected access protocols.  The optional attributes
   have proposed mappings in those protocols.

   It is important to note that the DAG/IP is constructed to carry any
   alternative attribute information that may be provided by a given
   WDSP; individual DAG-SAPs and DAG-CAPs may choose to pass along,
   interpret, or ignore any attributes not defined in this appendix.

   Additionally, note that the order of attributes in the DAG/IP is
   significant, which means that it is possible to use one attribute to
   carry the information describing the type of subsequent ones (e.g.,
   see the "ADR-TYPE" attribute below).

   Finally, attributes may be repeated.  For example, this schema
   structure can carry  multiple phone numbers of different types for
   one person.

A.1 DAG Personal Information Schema (DAGPERSON Schema)

```
   Attribute     Designation   Specific Description
   ---------     -----------   -----------------------------------
   FN            Required      Free-text representation of full name
   EMAIL         Required      Internet e-mail address
   LOC           Required      Locality -- geographic region
   ORG           Required      Person's organization
   ADR-TYPE      Optional      Type of address that follows
                               ("org", "home", "org-postal",
                               "home-postal", "unqualified")
   ADR           Optional      Full address
   ADR-STREET    Optional      Street address component
   ADR-ROOM      Optional      Suite or room number component
   ADR-CITY      Optional      City name
   ADR-STATE     Optional      Region of address
   ADR-COUNTRY   Optional      Country
   ADR-CODE      Optional      Postal code component
   TEL-TYPE      Optional      Type of telephone number (
                               "work",  "home", "mobile",
                               "fax" ,"pager", "unqualified")
                               in the following attribute
   TEL           Optional      A phone number for the person
   SOURCE        Optional      The WDSP's preferred  access to
                               their service -- a URL
   DN            Optional      Entry's "distinguished name"
                               (for LDAP)
```

      Table A.1 DAGPERSON schema attributes

A.2 DAG Organizational Role Information Schema (DAGORGROLE Schema)

```
     Attribute    Designation      Specific Description
     ---------    -----------      --------------------
     ROLE         Required         Name of organizational role
     EMAIL        Required         E-mail address associated with role
     ORG          Required         Name of organization
     LOC          Required         Locality -- geographic region
     TEL-TYPE     Optional         Type of telephone number
                                   in the TEL attribute immediately
                                   following("org" or "fax")
     TEL          Optional         Phone number

     FN           Optional         Full name of current role occupant
     SOURCE       Optional         The WDSP's preferred  access to their
                                    service -- a URL
     DN           Optional         Entry's "distinguished name" (for LDAP)
```

     Table A.2 DAGORGROLE schema attributes

Appendix B - Schema Mappings for Whois++ and LDAP

   The DAG/IP makes use of two specific schemas, as defined above.
   However, schemas particular to access protocols need to be handled in
   order to appropriately address incoming user queries, and chaining
   queries to WDSPs.  The recognized standard schemas are:

   - the USER template for Whois++ ([8])
   - the ORGROLE template for Whois++ ([8])
   - the inetOrgperson objectclass for LDAP ([16])
   - the organizationalrole objectclass for LDAP ([18])

   The DAG/IP schemas were developed based on the information that the
   TISDAG project requirements wish to return in results, in conjunction
   with information about standard schemas used in the basic WDSP access
   protocols (LDAPv2/v3 and Whois++).  However, particularly in the case
   of address information, the schemas used for those protocols allow
   for considerable scope of information representation.  In practice,
   this means that different WDSPs may choose to use different sub-parts
   of the schema, or even implement local customizations.

   Therefore, Appendix A outlines a very basic schema that can carry all
   the necessary information.  The basic DAG-CAPs and DAG-SAPs are
   designed to work to that information structure.  This appendix
   outlines the expected behaviour for DAG-SAPs mapping into the DAG/IP
   schema, and DAG-CAPs extracting information to pass along to client
   software after a chaining operation has returned results.

B.1 LDAP and the DAG Schemas

   The only time information is carried in the DAG schemas is when a
   DAG-SAP is returning information (obtained from  WDSPs' servers) to a
   DAG-CAP using the DAG/IP.  The "canonical" mappings between standard
   LDAP object classes (inetorgPerson, defined in [16] and
   organizationalRole, defined in [18] and the DAGPERSON schema and
   DAGORGROLE schema are defined such that information passed from an
   LDAP DAG-SAP to an LDAP DAG-CAP (e.g., in the case of an LDAPv3 DAG-
   SAP returning information chained for an LDAPv2 DAG-CAP) will be
   mapped into the same attributes as it was extracted.

   However, the representation of some attributes (such as address) is
   truly widely varied between protocol paradigms.  The goal with the
   "reasonable approximation" mappings that are provided is to give
   DAG-CAPs a basic mechanism for communicating information drawn from
   non-LDAP DAG-SAP sources.  The mappings may not be perfect, but they
   will convey the information to the end-user in some LDAP-
   understandable fashion, which is the goal of this project's effort.

   The canonical mappings for the LDAP inetorgPerson object class and
   the DAGPERSON schema are given in Table B.1.  A few reasonable
   approximation mappings follow in Table B.2.  Beyond that, DAG-SAPs
   may pass along any additional attributes in the DAG/IP, and DAG-CAPs
   may elect to forward or interpret any that are recognizable (e.g.,
   the sn ("surname") attribute is not listed here, but a DAG-SAP might
   return that in the DAG/IP, and a DAG-CAP, recognizing the string
   representation, could elect to include it in its LDAP response to the
   client).

```
   DAGPERSON Attribute      LDAP inetorgPerson attribute
   -------------------      ----------------------------
   FN                       cn
   EMAIL                    mail
   LOC                      l
   ORG                      o

   ADR-TYPE=org
   ADR-STREET               street
   ADR-ROOM                 roomNumber
   ADR-STATE                st
   ADR-COUNTRY              c

   ADR-TYPE=org-postal
   ADR                      postalAddress
   ADR-ROOM                 postOfficeBox
   ADR-CODE                 postalCode
```

```
ADR-TYPE=home-postal
ADR                        homePostalAddress


TEL-TYPE=work
TEL                        telephoneNumber


TEL-TYPE=home
TEL                        homePhone


TEL-TYPE=fax
TEL                        facsimileTelephoneNumber


TEL-TYPE=mobile
TEL                        mobile


TEL-TYPE=pager
TEL                        pager


DN                         dn
SOURCE                     labeledURI
```

Table B.1 Canonical DAGPERSON schema & LDAP inetorgPerson attributes

```
DAGROLE Attribute          LDAP organizationalRole attribute
----------------------     ---------------------------------
ADR-TYPE=unqualified
ADR                        street
ADR-STREET                 street
ADR-ROOM                   room
ADR-STATE                  st
ADR-COUNTRY                c


TEL-TYPE=unqualified
TEL                        telephoneNumber
```

Table B.2 Reasonable Approximations for LDAP organizationalRole
attributes

For example, consider the following LDAP record information, in LDIF
[11] format:

```
dn: cn=Barbara Jensen, ou=Product Development, o=Ace Industry,
c=US
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetorgperson
cn: Barbara Jensen
cn: Barbara J Jensen
cn: Babs Jensen
sn: Jensen
uid: bjensen
telephonenumber: +1 408 5551212
description:  A big sailing fan
```

This would validly be carried in the DAGPERSON schema as follows:

```
DN: cn=Barbara Jensen, ou=Product Development, o=Ace Industry,
c=US
FN: Barbara Jensen
FN: Barbara J Jensen
FN: Babs Jensen
SN: Jensen
TEL-TYPE: work
TEL:  +1 408 5551212
```

The canonical mappings for the LDAP organizationalRole object class
and the DAGORGROLE schema are given in Table B.3 .Beyond that, DAG-
SAPs may elect to send along any attributes, and DAG-CAPs may
interpret any that are recognizable.  N.B., the organizationalRole
class does not include provision for inclusion of an e-mail address.
This mapping rather blithely assumes the availability of the mail
attribute as defined for inetorgPerson.

```
   DAGORGROLE Attribute    LDAP organizationalRole attribute
   -------------------     -------------------------------
   ROLE                    cn
   EMAIL                   mail
   ORG                     o
   LOC                     l

   TEL-TYPE=org
   TEL                     telephoneNumber

   TEL-TYPE=fax
   TEL                     facsimileNumber

   FN                      roleOccupant
   DN                      dn
   SOURCE                  labeledURI
```

   Table B.3 Canonical mappings for LDAP organizationalRole attributes

B.2 Whois++ and the DAG Schemas

   The "canonical" mappings between standard Whois++ templates as
   defined in [8] and the DAGPERSON schema and DAGORGROLE schema are
   defined in Tables B.4 and B.5.  Beyond that, DAG-SAPs may pass along
   any additional attributes in the DAG/IP, and DAG-CAPs may elect to
   forward or interpret any that are recognizable.

```
   DAGPERSON Attribute     Whois++ USER template attribute
   -------------------     -------------------------------
   FN                      name
   EMAIL                   email
   LOC                     address-locality
   ORG                     organization-name

   ADR-TYPE=unqualified
   ADR                     address

   ADR-TYPE=org
   ADR                     organization-address
   ADR-STREET              organization-address-street
   ADR-ROOM                organization-address-room
   ADR-CITY                organization-address-city
   ADR-STATE               organization-address-state
   ADR-COUNTRY             organization-address-country
   ADR-CODE                organization-address-zip-code
```

```
   ADR-TYPE=home           address-type=home
   ADR                     address
   ADR-STREET              address-street
   ADR-ROOM                address-room
   ADR-CITY                address-city
   ADR-STATE               address-state
   ADR-COUNTRY             address-country
   ADR-CODE                address-zip-code

   TEL-TYPE=work           phone-type=work
   TEL                     phone

   TEL-TYPE=home           phone-type=home
   TEL                     phone

   TEL-TYPE=fax
   TEL                     fax

   TEL-TYPE=mobile
   TEL                     cellular

   TEL-TYPE=pager
   TEL                     pager
```

   Table B.4 Canonical DAGPERSON schema & Whois++ USER attributes

```
   DAGORGROLE Attribute      Whois++ ORGROLE attribute
   -------------------       -------------------------
   ROLE                      org-role
   EMAIL                     email
   ORG                       organization-name
   LOC                       organization-address-locality
   FN                        name

   TEL-TYPE=org
   TEL                       phone

   TEL-TYPE=fax
   TEL                       fax
```

   Table B.5 Canonical mappings for Whois++ ORGROLE attributes

Appendix C - DAG-Internal Protocol (DAG/IP)

   The DAG-Internal Protocol (DAG/IP) is currently defined as a
   derivative of the query-interaction protocol of Whois++ as laid out
   in RFC1835 ([6]).

C.1 A word on the choice of DAG/IP

   The use of the DAG/IP is strictly internal to the DAG system.  In
   that regard, it is possible make use of any query language, or define
   a new one.

   The Whois++ protocol was selected as the basis of the DAG/IP for
   several reasons:

   - it has the power and flexibility to convey all necessary queries
   - it is a simple, text-based protocol; clients need not implement the
     full functionality of the protocol in order to carry out minimal
     queries
   - the power of the full-fledge directory service query protocol will
     give DAG-CAP writers the ability to express more sophisticated
     queries if desired (e.g., to produce more intricate "intelligent"
     matching of spellings, common character substitutions, etc).
   - the text-based, delimited attribute results expression facilitates
     optional inclusion of  extra data supplied by WDSPs -- DAG-CAPs can
     easily ignore any unknown information and continue to interpret the
     rest of the result information.

   Also, the use of an existing protocol leverages the experience and
   time of the creators of the protocol -- hammering out such elusive
   and yet necessary details as handling line-endings, quoting special
   characters, etc.

   There is a freely-available test suite of tools for testing servers'
   Whois++ protocol conformance (for the Referral Index, and for DAG-
   SAPs).  Send mail to digger-info@bunyip.com for further information.

C.2 DAG/IP Input and Output -- Overview

   Input interactions in DAG/IP are as defined in RFC1835, "Architecture
   of the WHOIS++ service" ([6]), sections 2.2 and 2.3.  Section C.3 of
   this document adapts the grammar used in more recent descriptions of
   the Whois++ protocol to illustrate the syntax of the DAG/IP.

   DAG/IP output will be a subset of what is defined in RFC1835, section
   2.4, except that referral responses ("SERVER-TO-ASK") contain more
   information.

C.3 BNF for DAG/IP input and output

   The following sections are adapted from the Whois++ grammar.  For
   discussion of the semantic intent of the query protocol, and other
   matters, see Whois++ RFC 1835 [6].

C.3.1 The DAG/IP Input Grammar

   The following grammar, which uses the Augmented BNF (ABNF) notation
   as defined in [5], defines the set of acceptable DAG/IP input.

   N.B.:  As outlined in the ABNF definition, rule names and string
   literals are in the US-ASCII character set, and are case-insensitive.
   Also,  when a character is written explicitly in the grammar, as for
   example ";", it represents the byte value of that character in all of
   the allowed character sets in their encodings used in this protocol.
   Specifically in UNICODE, ";" means the character U+003B, which when
   encoding the character in UTF-8 will generate the byte value 0x3B
   which is then used in the DAG/IP protocol.

   dagip-command   = ( system-command [":" "hold"]
                   / ri-query
                   / sap-query ) nl

   ri-query        =   ri-terms [":" globalcnstrnts]

   sap-query       =   sap-terms [":" [sapcnstrnts][ ":" wdspinfo]]

   system-command =    "constraints"
                   / "describe"
                   / "commands"
                   / "polled-by"
                   / "polled-for"
                   / "version"
                   / "list"
                   / "show" [1*sp datastring]
                   / "help" [1*sp datastring]
                   / "<NL>" [string]

   ri-terms        =   ri-and-expr *(1*sp "or" 1*sp ri-and-expr)

   ri-and-expr     =   ri-basic-expr *(1*sp "and" 1*sp ri-basic-
   expr)

   ri-basic-expr   =   ["not" 1*sp] ri-term / ( "(" ri-terms ")" )

   ri-term         =   generalterm / specificterm / combinedterm

   sap-terms       =   sap-and-expr *(1*sp "or" 1*sp sap-and-expr)

   sap-and-expr    =   sap-basic-expr *(1*sp "and" 1*sp
                       sap-basic-expr)

   sap-basic-expr  =   ["not" 1*sp] sap-term / ( "(" sap-terms ")" )

```
sap-term         =     ( generalterm / specificterm / combinedterm)
                       localcnstrnts

generalterm      =     datastring
```

   TISDAG: Since the DAG system only supports certain attribute
   combinations in its queries, (Table 3.1).  The use of generalterm
   may lead to unexpected behaviour and is therefore deprecated.
   CAPs should therefore not use it even if it is in the protocol.

```
specificterm     =     specificname "=" datastring

specificname     =     "handle" / "value"

combinedterm     =     attributename "=" datastring

sapcnstrnts      =     sapcnstrnt *(";" sapcnstrnt)

sapcnstrnt       =     localcnstrnt / globalcnstrnt

localcnstrnts    =     [";search=" sap-searchvalue] [";case="
                       sap-casevalue]

localcnstrnt     =     "search=" sap-searchvalue / "case="
                       sap-casevalue
```

   ;N.B.:  in the case where local and global constraints
   ;       conflict, local constraints take precedence
   ;       and overrides the global constraint

```
sap-searchvalue  =     "tstring" / searchvalue

sap-casevalue    =     "consider" / "ignore"

globalcnstrnts   =     globalcnstrnt *(";" globalcnstrnt)

globalcnstrnt    =     "search" "=" searchvalue
                  / opt-globalcnst

opt-globalcnst   =     "hold"
                  / "case" "=" casevalue
                  / "maxfull" "=" 1*digit
                  / "maxhits" "=" 1*digit
                  / "language" "=" language
                  / "incharset" "=" characterset
                  / "ignore" "=" attributename
                  / "include" "=" attributename
```

```
; N.B.: If an attribute is named both with the "include" and "ignore"
; constraints, the attribute is to be included in the result, but the
; system message must be "% 112 Requested constraint not fulfilled".

language         = <The language code defined in RFC1766>

characterset     = "UNICODE-2-0-UTF-8"

searchvalue      = "exact" / "substring" / "lstring"

casevalue        = "ignore" / "consider"

wdspinfo         = attrValAss *( ";" attrValAss )

attrValAss       = attributename "=" datastring
```

TISDAG: Within the boundaries of the TISDAG project it has been
decided that the only permitted attributes for wdspinfo are
"host","port","server-info" and "charset".  Regarding "charset"
the values for this attribute are defined to be one of "UTF-8",
"ISO8859-1","T\.61" or "US-ASCII".

```
datastring       = 1*data-elt

attributename    = 1*(<%d32-126 except specialbyte>)
                      ; omit 127, which is DEL

data-elt         = "\" specialbyte / normalbyte

normalbyte       = <%d32-255, except specialbyte>

specialbyte      =   " " / tab / "=" / "," / ":" / ";" / "\" /
                    "*" / "." / "(" / ")" / "[" / "]" / "^" /
                    "$" / "!" / "<NL>"

number           = 1*digit

digit            =   "0" / "1" / "2" / "3" / "4" /
                    "5" / "6" / "7" / "8" / "9"

tab              = %d09
sp               = %d32                 ; space
nl               = %d13 %d10            ; CR LF
```

   NOTE: Spaces (sp) that are significant to a query must be escaped.
   The following characters, when significant to the query, may  be
   preceded and/or followed by a single space:
     : ; , ( ) = !

C.3.2 The DAG/IP Response Grammar

   The following grammar, which uses the Augmented BNF (ABNF) notation
   as defined in RFC2234 (see [5]),

   N.B.:  As outlined in the ABNF definition, rule names and string
   literals are in the US-ASCII character set, and are case-insensitive.
   Also,  when a character is written explicitly in the grammar, as for
   example ";", it represents the byte value of that character in all of
   the allowed character sets in their encodings used in this protocol.
   Specifically in UNICODE, ";" means the character U+003B which when
   encoding the character in UTF-8 will generate the byte value 0x3B
   which is then used in the DAG/IP protocol.

   server-resp     =    goodmessage mnl output mnl endmessage
                    / badmessage nl endmessageclose

   output          =    0*(full-record / server-to-ask)

   full-record     =    "# FULL " template " " serverhandle " "
                           localhandle system-nl
                   1*fulldata
                    "# END" system-nl

      TISDAG: serverhandle is:

     - Whois++, whatever the server-handle on the record returned by
       the WDSP.
     - LDAP, <hostname-without-periods><port> (because server DN's are
       not enforceably unique).  E.g., a services.bunyip.com server on
       7778 would become servicesbunyipcom7778.

      localhandle is:
     - Whois++:  the localhandle on the record returned by the WDSP
     - LDAP, it is the RDN (relative  distinguished name), with spaces
       replaced by "_".  E.g., cn=leslie_daigle

   server-to-ask   =    "# SERVER-TO-ASK " serverhandle system-nl
                    server-to-askdata
                    "# END" system-nl

   fulldata        =    " " attributename ": " attributevalue
   system-nl

```
   server-to-ask-data = " Server-Info: " serverinfo system-nl
                    " Host-Name: " hostname system-nl
                    " Host-Port: " number system-nl
                    " Protocol: " prot system-nl
                    " Source-URI: " source system-nl
                    " Charset: " characterset system-nl

   attributename   =   r-string

   attributevalue  =   longstring

   template        =   <%d32-%d255 except specialbyte>

   serverhandle    =   <%d32-%d255 except specialbyte>

   localhandle     =   <%d32-%d255 except specialbyte>

   serverinfo      =   string

   hostname        =   string

   prot            =   string ; currently one of "ldapv2"
                           ; "ldapv3" "whois++"

   characterset    =   "UTF-8" / "T.61" / "ISO8859-1" / "US-ASCII"

   source          =   string

   longstring      =   string 0*( nl ( "+" / "-" ) string )

   string          =   0*(%d32-255)

   r-string        =   0*(<%d32-126 except specialbyte>)
                        ; omit 127 which is DEL

   specialbyte     =   ":" / " "

   mnl             =   1*system-nl

   system-nl       =   nl [ 1*(message nl) ]

   nl              =   %d13 %d10    ; CR and LF

   message         =   [1*( messagestart "-" string nl)]
                    messagestart " " string nl

   messagestart    =   "% " digit digit digit
```

```
   goodmessage       =   [1*( goodmessagestart "-" string nl)]
                      goodmessagestart " " string nl

   goodmessagestart=    "% 200"

   badmessage        =   [1*( badmessagestart "-" string nl)]
                      badmessagestart " " string nl

   badmessagestart   =   "% 5" digit digit

   endmessage        =   endmessageclose / endmessagecont

   endmessageclose   =   [endmessagestart " " string nl]
                      byemessage

   endmessagecont    =   endmessagestart " " string nl

   endmessagestart   =   "% 226"

   byemessage        =   byemessagestart " " string nl

   byemessagestart   =   "% 203"

   number            =   1*( digit )

   digit             =   "0" / "1" / "2" / "3" / "4" / "5" / "6" /
                      "7" / "8" / "9"
```

C.4 DAG/IP Response Messages

   The following list and discussion of response codes is derived from
   the Whois++ protocol definition, RFC1835 ([6]).

   A system message begins with a '%', followed by a space and a three
   digit number, a space, and an optional text message.  The line
   message must be no more than 81 bytes long, including the terminating
   CR LF pair.  There is no limit to the number of system messages that
   may be generated.

   A multiline system message have a hyphen instead of a space in column
   6, immediately after the numeric response code in all lines, except
   the last one, where the space is used.

   Example 1

   % 200 Command okay

   Example 2

```
% 220-Welcome to
% 220-the Whois++ server
% 220 at ACME inc.
```

The client is not expected to parse the text part of the response
message except when receiving reply 600 or 601, in which case the
text part is in the former case the name of a character set that will
be used by the server in the rest of the response, and in the latter
case when it specifies what language the attribute value is in.  The
valid values for characters sets is specified in the "characterset"
list in the BNF listing in Appendix C.

The theory of reply codes is described in Appendix E in STD 10,
RFC821 ([15]).

| System response code | Description |
| --- | --- |
| 110 Too many hits | The number of matches exceeded the value specified by the maxhits constraint.  Server will still reply with as many records as "maxhits" allows. |
| 111 Requested constraint not supported | One or more constraints in query is not implemented, but the search is still done. |
| 112 Requested constraint not fulfilled | One or more constraints in query has unacceptable value and was therefore not used, but the search is still done. |
| 200 Command Ok | Command accepted and executed. The client must wait for a transaction end system message. |
| 201 Command Completed successfully | Command accepted and executed. |
| 203 Bye | Server is closing connection |

204 Overgeneralized          The server could not exactly
                             match the DAG query into its
                             native access protocol.  The
                             resulting native query was
                             "looser".

220 Service Ready            Greeting message.  Server is
                             accepting commands.

226 Transaction complete     End of data.  All responses to
                             query are sent.

401 Service not available

402 Search expression
    too complicated

403 Information Unavailable   When a remote service is not
                             (currently) available.

404 Time out

500 Syntax error

502 Search expression too     This message is sent when the
    complicated              server is not able to resolve a
                             query (i.e. when a client sent a
                             regular expression that is too
                             deeply nested).

503 Query to general         This is like the "too many hits"
                             situation, but the server does
                             not send along any results.  This
                             message is used to deflect data
                             mining.

505 Operations error         Permanent operations error

600 <token>                  Subsequent attribute values are
                             encoded in the character set
                             specified by <token>.

601 <token>                  Subsequent attribute values are
                             in the language specified by
                             <token>.

601 DEF                        Subsequent attribute values are
                               default values, i.e. they should
                               be used for all languages not
                               specified by "601 <token>" since
                               last "601 ANY" message.

601 ANY                        Subsequent attribute values are
                               for all languages.

Table C.1 List of system response codes

Appendix D - DAG/IP Response Messages Mapping

```
  LDAPv2/v3                                  DAG/IP
  ----------------------------------------   ---------------------
  success                       (0) v2&v3    200 Command Ok
  operationsError               (1) v2&v3    505 Operations error
  protocolError                 (2) v2&v3    505 Operations error
  timeLimitExceeded             (3) v2&v3    404 Timeout
  sizeLimitExceeded             (4) v2&v3    110 To many hits
  compareFalse                  (5) v2&v3    200 OK
  compareTrue                   (6) v2&v3    200 OK
  authMethodNotSupported        (7) v2&v3    505 Operations error
  strongAuthRequired            (8) v2&v3    505 Operations error
  referral                     (10) v3       200 OK
  adminLimitExceeded           (11) v3       110 Too many hits
  unavailableCriticalExtension (12) v3       505 Operations error
  confidentialityRequired      (13) v3       505 Operations error
  saslBindInProgress           (14) v3       N.A.
  noSuchAttribute              (16) v2&v3    200 OK
  undefinedAttributeType       (17) v2&v3    500 Syntax error
  inappropriateMatching        (18) v2&v3    500 Syntax error
  constraintViolation          (19) v2&v3    111 Requested constraint
                                                 not supported
  attributeOrValueExists       (20) v2&v3    200 OK
  invalidAttributeSyntax       (21) v2&v3    500 Syntax error
  noSuchObject                 (32) v2&v3    200 OK
  aliasProblem                 (33) v2&v3    505 Operations error
  invalidDNSyntax              (34) v2&v3    500 Syntax error
  isLeaf                       (35) v2       N.A.
  aliasDereferencingProblem    (36) v2&v3    505 Operations error
  inappropriateAuthentication  (48) v2&v3    500 Syntax error
  invalidCredentials           (49) v2&v3    403 Information Unavailable
  insufficientAccessRights     (50) v2&v3    403 Information Unavailable
   busy                        (51) v2&v3     403 Information Unavailable
  unavailable                  (52) v2&v3    401 Service not available
  unwillingToPerform           (53) v2&v3    505 Operations error
  loopDetect                   (54) v2&v3    505 Operations error
  namingViolation              (64) v2&v3    N.A.
  objectClassViolation         (65) v2&v3    N.A.
  notAllowedOnNonLeaf          (66) v2&v3    N.A.
  notAllowedOnRDN              (67) v2&v3    N.A.
  entryAlreadyExists           (68) v2&v3    N.A.
  objectClassModsProhibited    (69) v2&v3    N.A.
  affectsMultipleDSAs          (71) v3       N.A.
  other                        (80) v2&v3    403 Information Unavailable
```

  Table D.1 LDAPv2/v3 resultcodes to DAG/IP response codes
  mapping

```
DAG/IP                                 LDAP v2/v3
-------------------------------------  -------------------------
110 Too many hits                      sizeLimitExceeded (4)
111 Requested constraint not supported constraintViolation (19)
112 Requested constraint not fullfilled constraintViolation (19)
200 Command Ok                         Success (0)
201 Command Completed successfully     N.A.
203 Bye                                N.A.
204 Overgeneralized                    N.A.
220 Service Ready                      N.A.
226 Transaction complete               N.A.
401 Service not available              unavailable (52)
402 Search expression too complicated  unwillingToPerform (53)
403 Information Unavailable             busy (51)
404 Time out                           timeLimitExceeded (3)
405 Operations error                   operationsError (1)
500 Syntax error                       protocolError (2)
502 Search expression too complicated  unwillingToPerform (53)
503 Query to general                   unwillingToPerform (53)
505 Operations error                   operationsError (1)
600 <token>                            N.A.
601 <token>                            N.A.
601 DEF                                N.A.
601 ANY                                N.A.
```

Table D.2 Mapping from DAG/IP response codes to LDAPv2/v3 resultcodes

```
DAG/IP                                 Whois++
-------------------------------------  ------------------------------
110 Too Many hits                      110 Too Many hits
111 Requested constraint not supported 111 Requested constraint not
                                           supported
112 Requested constraint not fullfilled 112 Requested constraint not
                                           fullfilled
200 Command Ok                         200 Command Ok
201 Command Completed successfully     201 Command Completed
                                           successfully
401 Service not available              401 Service not available
403 Information Unavailable            403 Information not available
404 Timeout                            404 Timeout
405 Operations error                   405 Operations error
500 Syntax error                       500 Syntax error
502 Search expression too complicated  502 Search expression too
                                           complicated
503 Query to general                   506 Query to general
505 Operations error                   505 Operations error
```

Table D.3 Mapping between DAG/IP and Whois++ response codes

Appendix E - DAG CIP Usage

E.1 CIP Index Object

   The CIP object used by the DAG system is based on the Tagged Index
   Object as defined in [12].   The grammar, adapted from that Work in
   Progress, for the specific object used by the DAG is as follows:


```
   index-object = 0*(io-part SEP) io-part
   io-part      = header SEP schema-spec SEP index-info
   header       = version-spec SEP update-type SEP this-update SEP
                  last-update context-size
   version-spec = "version:" *SPACE "x-tagged-index-1"
   update-type  = "updatetype:" *SPACE ( "total" |
                  ( "incremental" [*SPACE "tagbased"|"uniqueIDbased" ])
   this-update  = "thisupdate:" *SPACE TIMESTAMP
   last-update  = [ "lastupdate:" *SPACE TIMESTAMP SEP]
   context-size = [ "contextsize:" *SPACE 1*DIGIT SEP]
   schema-spec  = "BEGIN IO-Schema" SEP 1*(schema-line SEP)
                  "END IO-Schema"
   schema-line  = attribute-name ":" token-type
   token-type   = "TOKEN"
   index-info   = full-index | incremental-index
   full-index   = "BEGIN Index-Info" SEP 1*(index-block SEP)
                  "END Index-Info"
   incremental-index = 1*(add-block | delete-block | update-block)
   add-block    = "BEGIN Add Block" SEP 1*(index-block SEP)
                  "END Add Block"
   delete-block = "BEGIN Delete Block" SEP 1*(index-block SEP)
                  "END Delete Block"
   update-block = "BEGIN Update Block" SEP
                  0*(old-index-block SEP)
                  1*(new-index-block SEP)
                   "END Update Block"
   old-index-block = "BEGIN Old" SEP 1*(index-block SEP)
                  "END Old"
   new-index-block = "BEGIN New" SEP 1*(index-block SEP)
                  "END New"
   index-block  = first-line 0*(SEP cont-line)
   first-line   = attr-name ":" *SPACE taglist "/" attr-value
   cont-line    = "-" taglist "/" attr-value
   taglist      = tag 0*("," tag) | "*"
   tag          = 1*DIGIT ["-" 1*DIGIT]
   attr-value   = 1*(UTF8)
   attr-name    = dag-searchattr / "objectclass"
   dag-searchattr = "FN" / "LOC" / "ROLE" / "ORG"
   TIMESTAMP    = 1*DIGIT
   NAMECHAR     = DIGIT | UPPER | LOWER | "-" | ";" | "."
```

```
SPACE           = <ASCII space, %x20>;
SEP             = (CR LF) | LF
CR              = <ASCII CR, carriage return, %x0D>;

LF              = <ASCII LF, line feed, %x0A>;

DIGIT           = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
                  "8" | "9"

UPPER           = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" |
                  "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" |
                  "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" |
                  "Y" | "Z"
LOWER           = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
                  "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" |
                  "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" |
                  "y" | "z"

US-ASCII-SAFE  = %x01-09 / %x0B-0C / %x0E-7F
                 ;; US-ASCII except CR, LF, NUL
UTF8           = US-ASCII-SAFE / UTF8-1 / UTF8-2 / UTF8-3
                     / UTF8-4 / UTF8-5
UTF8-CONT       = %x80-BF
UTF8-1          = %xC0-DF UTF8-CONT
UTF8-2          = %xE0-EF 2UTF8-CONT
UTF8-3          = %xF0-F7 3UTF8-CONT
UTF8-4          = %xF8-FB 4UTF8-CONT
UTF8-5          = %xFC-FD 5UTF8-CONT
```

N.B.:  The only tokenization type permitted is "TOKEN".  While the
Tagged Index Object memo permits the use of "FULL" (i.e., the entire
value of the attribute is preserved as a single token), that has the
danger of yielding a unique token for every record.  Studies in the
growth of centroid sizes as a function of number of records (see
[14]) demonstrate that such unique tokens (e.g., phone numbers)  are
to be avoided.  While storing tag information requires some number of
extra bytes of storage per token index entry, using unique tokens
causes the number of token entries in the index to continue to grow
linearly with the number of records, thereby affecting search
efficiency.

Note also that tags are to be applied to the data on a per entry
level.  Thus, if two index lines in the same index object contain the
same tag, then it is always the case that those two lines refer back
to the same "record" in the directory.  In LDAP terminology, the two
lines would refer back to the same directory object.

   Additionally if two index lines in the same index object contain
   different tags, then it is always the case that those two lines refer
   back to different records in the directory.

   The attribute "objectclass" is used to denote the record/object types
   in the data summarized in this index object.

   Values for the objectclass attribute should be restricted to:
   dagperson or dagrole, the two DAG schema object types.

E.2 CIP Index Object Creation

   WDSPs are expected to create index objects following the general
   principles outlined in the Whois++ protocol documentation (creation
   of centroids) and the Tagged Index Object documentation ([12]).
   Following the syntax described above, the index object contains token
   information for each attribute in the DAGSchema:

   - a list of all the unique tokens (strings delimited by the specified
     characters) that appear in the WDSP database for the attribute
   - for each token in that list, which records the token appears in

   So, for example,

   Record #1:
      FN: Foo Bar
      ORG: The Snack Bar

   Record #2:
       FN: Bar Smith
      ORG: Snack Shack

   yields (conceptually) the following information for the attribute FN:

   Foo (1), Bar (1,2), Smith (2)

   and the following information for the attribute ORG:

   The (1), Snack (1, 2), Bar (1), Shack (2)

   Note that the record numbers here are used simply as tags or virtual
   record  identifiers to indicate when 2 tokens appear in the same
   record.  The record identifiers are not used for any part of any
   query to the WDSP.

There is some discussion as to whether the use of the same record tag
for all attributes makes it too easy to "decompile" the index object;
i.e., reconstruct a WDSPs data based on re-ordering the tokens
associated with each attribute and tag number.  However, we are
dealing only with the search attributes here, which is a minimal
subset of the quantity of data held by the WDSP.  The conclusion is
then that the improved efficiency given by using the same tag numbers
across attributes outweighs the (remote) possibility of information
reconstruction.

This would yield the index object:

```
version: x-tagged-index-1
update-type: total
this-update: 855938804

last-update:
context-size:
BEGIN IO-Schema
objectclass: TOKEN

FN: TOKEN
ORG: TOKEN
END IO-Schema
BEGIN Index-Info
objectclass: */dagperson
FN: 1/Foo
-1,2/Bar
 -2/Smith
ORG: 1/The
-1,2/Snack
-1/Bar
-2/Shack
End Index-Info
```

   TISDAG: Within the project it has been decided to base consistency
   between updates on consistent tags.  This means that if the
   update-type is "incremental" the specifier must be "tagbased".

E.3 CIP Index Object Sharing

E.3.1 Registration of Servers

It is beyond the scope of this document to define how WDSP servers
shall be registered with the DAG Referral Index.  Such a procedure
must be defined, and the following information established for each
WDSP dataset (adapted from the Tagged Index Object specification,
[12]):

dsi: An OID which uniquely identifies the subtree and scope of the
  dataset for which the index object is created.

base-uri: One or more URI's which will form the base of any referrals
  created based upon the index object that is governed by this
  agreement.  For example, for LDAP the base-uri would specify (among
  other items): the LDAP host,  the base object to which this index
  object refers (e.g., c=SE), and the scope of the index object
  (e.g., single container).

supplier: The hostname and listening port number of the supplier
  server, as well as any alternative servers holding that same naming
  contexts, in case the supplier is unavailable.

source-uri: The URI of the WDSP's preferred source of directory
  service information.  This might be, for instance, an HTTP-based
  service.

consumeraddr: This is a URI of the "mailto:" form, with the RFC 822
  email address of the consumer server.

updateinterval: The maximum duration in seconds between occurrences
  of the supplier server generating an update.  If the consumer
  server has not received an update from the supplier server after
  waiting this long since the previous update, it is likely that the
  index information is now out of date.  A typical value for a server
  with frequent updates would be 604800 seconds, or every week.

attributeNamespace: Every set of index servers that together wants to
  support a specific usage of indices, has to agree on which
  attributenames to use in the index objects.  The participating
  directory servers also has to agree on the mapping from local
  attributenames to the attributenames used in the index.  Since one
  specific index server might be involved in several such sets, it
  has to have some way to connect a update to the proper set of
  indexes.  One possible solution to this would be to use different
  DSIs.

consistencybase: How consistency of the index is maintained over
  incremental updates:
  complete - every change or delete concerning one object has to
    contain all tokens connected to that object.  This method must be
    supported by any server who wants to comply with this standard
   tagbased - starting at a full update every incremental update
    referring back to this full updated has to maintain state-
    information regarding tags, such that a object within the
    original database is assigned the same tagnumber every time.
    This method is optional.

   uniqueID - every object in the Dataset has to have a unique value
      for a specific attribute in the index.  A example of such a
      attribute could be the distinguishedName attribute.  This method
      is also optional.

   securityoption: Whether and how the supplier server should sign and
      encrypt the update before sending it to the consumer server.
      Options for this version of the DAG service are "none": the update
      is sent in plaintext "PGP/MIME": the update is digitally signed and
      encrypted using PGP  (see [7]).  PGP/MIME is recommended.

   security credentials: The long-term cryptographic credentials used
      for key exchange and authentication of the consumer and supplier
      servers, if a security option was selected.  For "PGP/MIME", this
      will be the trusted public keys of both servers.

E.3.2 Transmission of Objects

   CIP Index Objects are sent to the DAG Referral Index by MIME-encoded
   SMTP, following the Common Indexing Protocol specification (see [2]
   and [3]).

Appendix F - Summary of Technical Survey Results

   As part of the TISDAG project, a technical survey was carried out --
   announced on the tisdag@swip.net mailing list, all Swedish WDSPs (and
   potential WDSPs) were encouraged to fill out and submit the WWW-based
   survey form (see http://tisdag.sunet.se/tisdag-survey.html).

   The survey was carried out in May, 1997.  Response was not as good as
   had been hoped -- in the end, 5 WDSPs participated.  We had hoped for
   more responses than this, in order to have a concrete sense of
   directory service providers' current and planned status.  However,
   informal "hallway" conversations with a few people at
   Interoperabilitet'97 in Sollentuna suggest that, while people see the
   TISDAG project as an important and timely step, they don't
   necessarily have an immediate understanding of how it will impact
   them, and what they can/should contribute.  So, the results can be
   seen as informational, though not a definitive statement of the whole
   directory service picture in Sweden.

   Interesting things to note from these results include the fact that,
   although there were only 5 respondents, these are clearly significant
   players -- 4 expect to have more than 100 000 records to contribute
   by 12 months from now.  There were no real surprises in terms of the
   supported protocols or search types.

Table E.1 summarizes information from the survey concerning types of
queries currently supported by WDSPs, and planned for the next 12
months.  Note that, at the time of the survey, the requirement of
searching by ROLE had not been proposed, so the survey did not
specifically ask if WDSPs supported both the DAGPERSON schema
protocol-equivalents (i.e., USER template in Whois++ and
inetorgperson objectclass in LDAP).  In the table, the column
"Complete info?" describes whether or not the WDSP currently returns
at least as much information as is required for a DAG reply.

| Resp | Search Types | Complete info? | Access Protocols (now) | Access Protocols (12 months) |
|------|--------------|----------------|-------------------------|-------------------------------|
| 1 | NOL | Except ROLE | Whois++ | Whois++ |
| 2 | N,NO,NL,NOL | Except ROLE | LDAPv2,DAP,PH, HTTP,Gopher | LDAPv2,LDAPv3,DAP, PH,HTTP,Gopher |
| 3 | N,NL,NOL | Except ROLE | LDAPv2,DAP,HTTP | LDAPv2,LDAPv3,DAP, HTTP |
| 4 | N,NO,NL,NOL | Except ROLE | Whois++,HTTP | LDAPv3,Whois++, HTTP,E-mail |
| 5 | N,NO,NL,NOL | Except ROLE | LDAPv2,Whois Whois++,HTTP | LDAPv2,LDAPv3, Whois,Whois++,PH, Finger,HTTP |

Table F.1 Summary of TISDAG Survey Results: Queries

| Resp | # of Records (now) | # of Records (12 months) | Character Sets |
|------|--------------------|--------------------------|----------------|
| 1 | 94 280 | 120 000 – 130 000 | ISO-8859-1 |
| 2 | 88 000 | 100 000 | ISO-8859-1 |
| 3 | N/A | 100 000 | T.61 (Telex) |
| 4 | 150 000 | 250 000 | ISO-8859-1 UTF-8 UNICODE |
| 5 | 4 300 | 10 000 | ISO-8859-1 |

Table F.2 Summary of TISDAG Survey Results: Operational Information

Appendix G - Useful References

   N.B.:  The following is a collection of Internet standards documents
   (RFCs) and Internet-Drafts from which the material in this report was
   drawn.  Internet-Drafts are works-in-progress, and are not meant to
   be cited.  Where they are used in this document, references are to
   the text contained in the Internet-Draft; i.e., they are not meant to
   imply standards, so much as useful starting points for the work of
   this project.

   Electronic copies of the version of the Internet-Drafts documents
   that were used in preparing this report are available from the
   project web page, http://tisdag.sunet.se.

Bibliography

   [1]  Allen, J. and M. Mealling, "The Architecture of the Common
        Indexing Protocol", RFC 2651, August 1999.

   [2]  Allen, J. and M. Mealing, "MIME Object Definitions for the
        Common Indexing Protocol (CIP)", RFC 2652, August 1999.

   [3]  Allen, J. and P. Leach, "CIP Transport Protocols", RFC 2653,
        August 1999.

   [4]  Crocker, D., "Standard for the Format of ARPA Internet Text
        Messages", STD 11, RFC 822, August 1982.

   [5]  Crocker, D., "Augmented BNF for Syntax Specifications: ABNF",
        RFC 2234, November 1997.

   [6]  Deutsch, P., Schoultz, R., Falstrom, P. and C. Weider,
        "Architecture of the WHOIS++ Service", RFC 1835, July 1995.

   [7]  Elkins, M., "MIME Security with Pretty Good Privacy (PGP)", RFC
        2015, October 1996.

   [8]  Patrik Faltstrom, Martin Hamilton, Leslie L. Daigle, "WHOIS++
        templates", Work in Progress.

   [9]  Freed, N. and N. Borenstein, "Multipurpose Internet Mail
        Extensions (MIME) Part One: Format of Interent Message Bodies",
        RFC 2045, November 1996.

   [10] Freed, N. and N. Borenstein, "Multipurpose Internet Mail
        Extensions (MIME) Part Two: Media Types", RFC 2046, November
        1996.

   [11] Good, G., "The LDAP Data Interchange Format (LDIF) - Technical
        Specification", RFC 2849, June 2000.

   [12] Hedberg, R., Greenblatt, B., Moats, R. and M. Wahl, "A Tagged
        Index Object for use in the Common Indexing Protocol", RFC 2654,
        August 1999.

   [13] Howes, R., "A String Representation of LDAP Search Filters", RFC
        1960, June 1996.

   [14] Paul Panotzki, "Complexity of the Common Indexing Protocol:
        Predicting Search Times in Index Server Meshes",  Master's
        Thesis, KTH, September 1996.

   [15] Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821,
        August 1982.

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

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

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

   [19] Yeong, W., Howes, T. and S. Kille, "Lightweight Directory Access
        Protocol", RFC 1777, March 1995.

   [20] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC
        2279, January 1998.

   [21] The Unicode Consortium, "The Unicode Standard -- Version 2.0",
        Addison-Wesley, 1996.

Authors' Addresses

    Leslie L. Daigle
    Thinking Cat Enterprises

    EMail: leslie@thinkingcat.com


    Roland Hedberg
    Catalogix
    Jegerveien 25
    0777 Oslo
    Norway

    Phone: +47 23 08 29 96
    EMail: Roland@catalogix.se