Technical Overview of Directory Services
Using the X.500 Protocol

Status of this Memo

   This memo provides information for the Internet community. It does
   not specify an Internet standard. Distribution of this memo is
   unlimited.

Abstract

   This document is an overview of the X.500 standard for people not
   familiar with the technology. It compares and contrasts Directory
   Services based on X.500 with several of the other Directory services
   currently in use in the Internet. This paper also describes the
   status of the standard and provides references for further
   information on X.500 implementations and technical information.

   A primary purpose of this paper is to illustrate the vast
   functionality of the X.500 protocol and to show how it can be used to
   provide a global directory for human use, and can support other
   applications which would benefit from directory services, such as
   main programs.

   This FYI RFC is a product of the Directory Information Services
   (pilot) Infrastructure Working Group (DISI). A combined effort of
   the User Services and the OSI Integration Areas of the Internet
   Engineering Task Force (IETF).

1. INTRODUCTION

   As the pace of industry, science, and technological development
   quickened over the past century, it became increasingly probable that
   someone in a geographically distant location would be trying to solve
   the same problems you were trying to solve, or that someone in a
   geographically distant location would have some vital information
   which impinged on your research or business. The stupendous growth
   in the telecommunications industry, from telegraphs to telephones to
   computer networks, has alleviated the problem of being able to

communicate with another person, PROVIDED THAT YOU KNOW HOW TO REACH
THEM.

Thus, along with the expansion of the telecommunications
infrastructure came the development of Directory Services. In this
paper, we will discuss various models of directory services, the
limitations of current models, and some solutions provided by the
X.500 standard to these limitations.

2   MODELS OF DIRECTORY SERVICES

2.1 The telephone company's directory services.

A model many people think of when they hear the words "Directory
Services" is the directory service provided by the local telephone
company. A local telephone company keeps an on-line list of the names
of people with phone service, along with their phone numbers and
their address. This information is available by calling up Directory
Assistance, giving the name and address of the party whose number you
are seeking, and waiting for the operator to search his database. It
is additionally available by looking in a phone book published yearly
on paper.

The phone companies are able to offer this invaluable service because
they administer the pool of phone numbers. However, this service has
some limitations. For instance, you can find someone's number only if
you know their name and the city or location in which they live. If
two or more people have listings for the same name in the same
locality, there is no additional information which with to select the
correct number. In addition, the printed phone book can have
information which is as much as a year out of date, and the phone
company's internal directory can be as much as two weeks out of date.
A third problem is that one actually has to call Directory assistance
in a given area code to get information for that area; one cannot
call a single number consistently.

For businesses which advertise in the Yellow Pages, there is some
additional information stored for each business; unfortunately, that
information is unavailable through Directory Assistance and must be
gleaned from the phone book.

2.2 Some currently available directory services on the Internet.

As the Internet is comprised of a vast conglomeration of different
people, computers, and computer networks, with none of the hierarchy
imposed by the phone system on the area codes and exchange prefixes,
any directory service must be able to deal with the fact that the
Internet is not structured; for example, the hosts foo.com and

v2.foo.com may be on opposite sides of the world, the .edu domain
maps onto an enormous number of organizations, etc. Let's look at a
few of the services currently available on the Internet for directory
type services.

2.2.1 The finger protocol.

The finger protocol, which has been implemented for UNIX systems and
a small number of other machines, allows one to "finger" a specific
person or username to a host running the protocol. This is invoked by
typing, for example, "finger clw@mazatzal.merit.edu". A certain set
of information is returned, as this example from a UNIX system finger
operation shows, although the output format is not specified by the
protocol:

    Login name: clw   In real life: Chris Weider
    Directory: /usr/clw   Shell: /bin/csh
    On since Jul 25 09:43:42   4 hours 52 minutes Idle Time
    Plan:
    Home: 971-5581

where the first three lines of information are taken from the UNIX
operating systems information and the line(s) of information
following the "Plan:" line are taken from a file named .plan which
each user modifies. Limitations of the fingerd program include: a)
One must already know which host to finger to find a specific person,
b) since primarily UNIX machines run fingerd, people who reside on
other types of operating systems are not locateable by this method,
c) fingerd is often disabled on UNIX systems for security purposes,
d) if one wishes to be found on more than one system, one must make
sure that all the .plan files are consistent, and e) there is no way
to search the .plan files on a given host to (for example) find
everyone on mazatzal.merit.edu who works on X.500. Thus, fingerd has
a limited usefulness as a piece of the Internet Directory.

2.2.2 whois

The whois utility, which is available on a wide of variety of
systems, works by querying a centralized database maintained at the
DDN NIC, which was for many years located at SRI International in
Menlo Park, California, and is now located at GSI. This database
contains a large amount of information which primarily deals with
people and equipment which is used to build the Internet. SRI (and
now GSI) has been able to collect the information in the WHOIS
database as part of its role as the Network Information Center for
the TCP/IP portion of the Internet.

The whois utility is ubiquitous, and has a very simple interface. A

typical whois query look like:

    whois Reynolds

and returns information like:

    Reynolds, John F. (JFR22) 532JFR@DOM1.NWAC.SEA06.NAVY.MIL
                                    (702) 426-2604 (DSN) 830-2604
    Reynolds, John J. (JJR40) amsel-lg-pl-a@MONMOUTH-EMH3.ARMY.MIL
                                    (908) 532-3817 (DSN) 992-3817
    Reynolds, John W. (JWR46) EAAV-AP@SEOUL-EMH1.ARMY.MIL
                                    (DSN) 723-3358
    Reynolds, Joseph T. (JTR10) JREYNOLDS@PAXRV-NES.NAVY.MIL
                                  011-63-47-885-3194 (DSN) 885-3194
    Reynolds, Joyce K. (JKR1) JKREY@ISI.EDU   (213) 822-1511
    Reynolds, Keith (KR35)   keithr@SCO.CO   (408) 425-7222
    Reynolds, Kenneth (KR94)   (502) 454-2950
    Reynolds, Kevin A. (KR39)   REYNOLDS@DUGWAY-EMH1.ARMY.MIL
                                    (801) 831-5441 (DSN) 789-5441
    Reynolds, Lee B. (LBR9)   reynolds@TECHNET.NM.ORG   (505) 345-6555

    a further lookup on Joyce Reynolds with this command line:

    whois JKR1

returns:

    Reynolds, Joyce K. (JKR1)   JKREY@ISI.EDU
        University of Southern California
        Information Sciences Institute
        4676 Admiralty Way
        Marina del Rey, CA 90292
        (310) 822-1511

        Record last updated on 07-Jan-91.

The whois database also contains information about Domain Name System
(DNS) and has some information about hosts, major regional networks,
and large parts of the MILNET system.

The WHOIS database is large enough and comprehensive enough to
exhibit many of the flaws of a large centralized database: a) As the
database is maintained on one machine, a processor bottleneck forces
slow response during times of peak querying activity, even if many of
these queries are unrelated, b) as the database is maintained on one
machine, a storage bottleneck forces the database administrators to
severely limit the amount of information which can be kept on each
entry in the database, c) all changes to the database have to be

mailed to a "hostmaster" and then physically reentered into the
database, increasing both the turnaround time and the likelihood for
a mistake in transcription.

2.2.3 The Domain Name System

The Domain Name System is used in the Internet to keep track of host
to IP address mapping. The basic mechanism is that each domain, such
as merit.edu or k-12.edu, is registered with the NIC, and at time of
registration, a primary and (perhaps) some secondary nameservers are
identified for that domain. Each of these nameservers must provide
host name to IP address mapping for each host in the domain. Thus,
the nameservice is supplied in a distributed fashion. It is also
possible to split a domain into subdomains, with a different
nameserver for each subdomain.

Although in many cases one uses the DNS without being aware of it,
because humans prefer to remember names and not IP addresses, it is
possible to interactively query the DNS with the nslookup utility. A
sample session using the nslookup utility:

```
home.merit.edu(1): nslookup
Default Server:  merit.edu
Address:  35.42.1.42

> scanf.merit.edu
Server:  merit.edu
Address:  35.42.1.42

Name:   scanf.merit.edu
Address: 35.42.1.92

> 35.42.1.92
Server:  merit.edu
Address: 35.42.1.42

Name: [35.42.1.92]
Address: 35.42.1.92
```

Thus, we can explicitly determine the address associated with a given
host. Reverse name mapping is also possible with the DNS, as in this
example:

```
     home.merit.edu(2): traceroute ans.net
     traceroute to ans.net (147.225.1.2), 30 hops max, 40 byte packets
       1 t3peer (35.1.1.33) 11 ms 5 ms 5 ms
       2 enss (35.1.1.1) 6 ms 6 ms 6 ms
            ................
       9 192.77.154.1 (192.77.154.1) 51 ms 43 ms 49 ms
      10 nis.ans.net (147.225.1.2) 53 ms 53 ms 46 ms
```

   At each hop of the traceroute, the program attempts to do a reverse
   lookup through the DNS and displays the results when successful.

   Although the DNS has served superlatively for the purpose it was
   developed, i.e. to allow maintenance of the namespace in a
   distributed fashion, and to provide very rapid lookups in the
   namespace, there are, of course, some limitations. Although there has
   been some discussion of including other types of information in the
   DNS, to find a given person at this time, assuming you know where she
   works, you have to use a combination of the DNS and finger to even
   make a stab at finding her. Also, the DNS has very limited search
   capabilities right now. The lack of search capabilities alone shows
   that we cannot provide a rich Directory Service through the DNS.

3: THE X.500 MODEL OF DIRECTORY SERVICE

   X.500 is a CCITT protocol which is designed to build a distributed,
   global directory. It offers the following features:

   * Decentralized Maintenance:
     Each site running X.500 is responsible ONLY for its local part
     of the Directory, so updates and maintenance can be done instantly.

   * Powerful Searching Capabilities:
     X.500 provides powerful searching facilities that allow users to
     construct arbitrarily complex queries.

   * Single Global Namespace:
     Much like the DNS, X.500 provides a single homogeneous namespace
     to users. The X.500 namespace is more flexible and expandable
     than the DNS.

   * Structured Information Framework:
     X.500 defines the information framework used in the directory,
     allowing local extensions.

* Standards-Based Directory:
  As X.500 can be used to build a standards-based directory,
  applications which require directory information (e-mail,
  automated resource locators, special-purpose directory tools)
  can access a planet's worth of information in a uniform manner,
  no matter where they are based or currently running.

3.1 Acronym City, or How X.500 Works

The '88 version of the X.500 standard talks about 3 models required
to build the X.500 Directory Service: the Directory Model, the
Information Model, and the Security Model. In this section, we will
provide a brief overview of the Directory and Information Models
sufficient to explain the vast functionality of X.500.

3.1.1 The Information Model

To illustrate the Information Model, we will first show how
information is held in the Directory, then we will show what types of
information can be held in the Directory, and then we will see how
the information is arranged so that we can retrieve the desired
pieces from the Directory.

3.1.1.1 Entries

The primary construct holding information in the Directory is the
"entry". Each Directory entry contains information about one object;
for example, a person, a computer network, or an organization. Each
entry is built from a collection of "attributes", each of which holds
a single piece of information about the object. Some attributes which
might be used to build an entry for a person would be "surname",
"telephonenumber", "postaladdress", etc. Each attribute has an
associated "attribute syntax", which describes the type of data that
attribute contains, for example, photo data, a time code, or a string
of letters and numbers. As an example, let's look at part of an entry
for a person.

 Entry for John Smith contains:

```
  attribute ---> surName=            Smith  <--- attribute value
          |---> telephoneNumber=   999-9999 <--- attribute value
          |---> title=              Janitor <--- attribute value
                          ...
```

The attribute syntax for the surName attribute would be
CaseIgnoreString, which would tell X.500 that surName could contain
any string, and case would not matter; the attribute syntax for the
telephoneNumber attribute would be TelephoneNumber, which would

specify that telephoneNumber could contain a string composed of
digits, dashes, parenthesis, and a plus sign. The attribute syntax
for the title attribute would also be CaseIgnoreString. A good
analogy in database terms for what we've seen so far might be to
think of a Directory entry as a database record, an attribute as a
field in that record, and an attribute syntax as a field type
(decimal number, string) for a field in a record.

3.1.1.2 Object Classes

At this point in our description of the information model, we have no
way of knowing what type of object a given entry represents. X.500
uses the concept of an "object class" to specify that information,
and an attribute named "objectClass" which each entry contains to
specify to which object class(es) the entry belongs.

Each object class in X.500 has a definition which lists the set of
mandatory attributes, which must be present, and a set of optional
attributes, which may be present, in an entry of that class. An given
object class A may be a subclass of another class B, in which case
object class A inherits all the mandatory and optional attributes of
B in addition to its own.

The object classes in X.500 are arranged in a hierarchical manner
according to class inheritance; the following diagram shows a part of
the object class hierarchy.

```
                          _____
                         |                |    "top" has one mandatory
                         |    top         |    attribute "objectClass",
                         |_____|    and nooptional attributes.
                          |     |    |
                          |     |    |    every other object class is a
     _____|     |    |    subclass of "top"...
    |                           |    |
  __|_____                  __|_____  ...
 |          |                |          |  "organization" inherits one
 | country  |                |organization|  mandatory attribute from
 |_____|                |_____|  "top", "objectClass"; adds one
                                           more mandatory attribute "O"
  "country" inherits one                   (for organization), and has
  mandatory attribute from "top",          many optional attributes. Any
  "objectClass", adds one more             subclass of "organization"
  mandatory attribute "c" (for             would inherit all of the
  country), and has two optional           mandatory and optional
  attributes, "description" and            attributes from "organization"
  "searchGuide". Any subclass of           including the attribute which
  "country" would inherit all of the       "organization" inherited
  mandatory and optional attributes        from "top".
  of the "country" class, including
  the attribute which "country"
  inherited from "top".
```
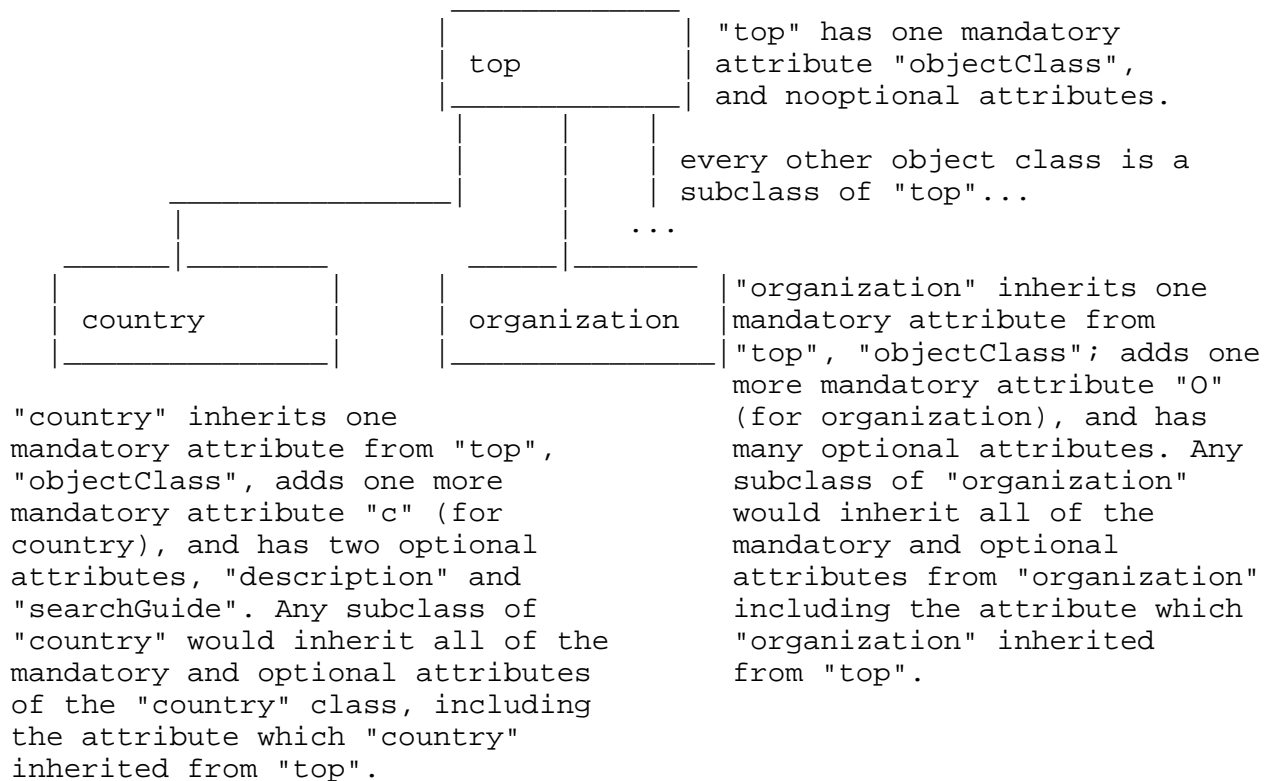
Figure 1.

   One major benefit of the object class concept is that it is in many
   cases very easy to create a new object class which is only a slight
   modification or extension of a previous class. For example, if I have
   already defined an object class for "person" which contains a
   person's name, phone number, address, and fax number, I can easily
   define an "Internet person" object class by defining "Internet
   person" as a subclass of "person", with the additional optional
   attribute of "e-mail address". Thus in my definition of the "Internet
   Person" object class, all my "person" type attributes are inherited
   from "person". There are other benefits which are beyond the scope of
   this paper.

3.1.1.3 X.500's namespace.

   X.500 hierarchically organizes the namespace in the Directory
   Information Base (DIB); recall that this hierarchical organization is
   called the Directory Information Tree (DIT). Each entry in the DIB
   occupies a certain location in the DIT. An entry which has no
   children is called a leaf entry, an entry which has children is
   called a non-leaf node. Each entry in the DIT contains one or more

attributes which together comprise the Relative Distinguished Name
(RDN) of that entry, there is a "root" entry (which has no
attributes, a special case) which forms the base node of the DIT. The
Distinguished Name of a specific entry is the sequence of RDNs of the
entries on the path from the root entry to the entry in question. A
diagram here will help to clarify this:

```
Level of DIT                Root          RDN       Distinguished Name

root                         *           nothing         { }
                           / | \
country (other            /  |  \
things at this           /   |   \              c=us           {c=us}
level)            c=gb   c=us    c=ca
                      /  |  \
                     /   |   \
                    /    |    \
organization     o=SRI  o=Merit  o=DEC  o=Merit      {c=us, o=Merit}
(other things       /   |   \
at this level)     /    |    \
                  /     |     \
Third level       cn=Chris Weider   cn=Chris Weider {c=us, o=Merit,
                                                      cn=Chris Weider}
```
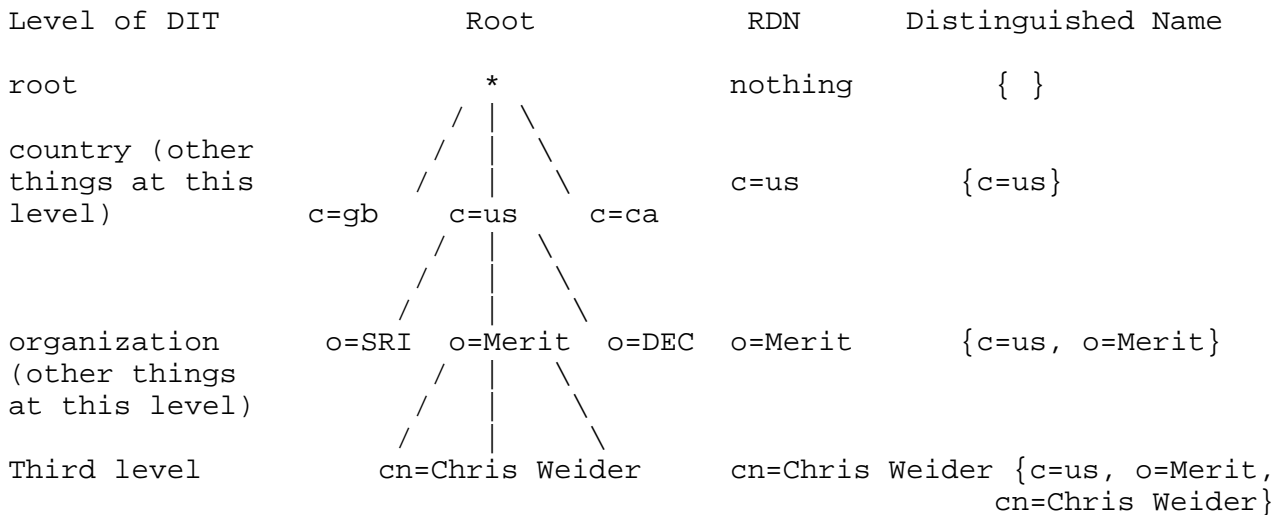
         Figure 2: Building a DN from RDNs (adapted from a
            diagram in the X.500 (88) Blue Book)

   Each entry in this tree contains more attributes than have been shown
   here, but in each case only one attribute for each entry has been
   used for that entry's RDN. As noted above, any entry in the tree
   could use more than one attribute to build its RDN. X.500 also allows
   the use of alias names, so that the entry {c=us, o=Merit, cn=Chris
   Weider} could be also found through an alias entry such as {c=us,
   o=SRI, ou=FOX Project, cn=Drone 1} which would point to the first
   entry.

3.1.2 The Directory Model

   Now that we've seen what kinds of information can be kept in the
   Directory, we should look at how the Directory stores this
   information and how a Directory users accesses the information. There
   are two components of this model: a Directory User Agent (DUA), which
   accesses the Directory on behalf of a user, and the Directory System
   Agent, which can be viewed as holding a particular subset of the DIB,
   and can also provide an access point to the Directory for a DUA.

   Now, the entire DIB is distributed through the world-wide collection
   of DSAs which form the Directory, and the DSAs employ two techniques

   to allow this distribution to be transparent to the user, called
   "chaining" and "referral". The details of these two techniques would
   take up another page, so it suffices to say that to each user, it
   appears that the entire global directory is on her desktop. (Of
   course, if the information requested is on the other side of the
   world, it may seem that the desktop directory is a bit slow for that
   request...)

3.2 The functionality of X.500

   To describe the functionality of X.500, we will need to separate
   three stages in the evolution of X.500: 1) the 1988 standard, 2)
   X.500 as implemented in QUIPU, and 3) the (proposed) 1992 standard.
   We will list some of the features described in the 1988 standard,
   show how they were implemented in QUIPU, and discuss where the 1992
   standard will take us. The QUIPU implementation was chosen because
   a) it is widely used in the U.S. and European Directory Services
   Pilot projects, and b) it works well. For a survey of other X.500
   implementations and a catalogue of DUAs, see [Lang].

3.2.1 Functionality in X.500 (88)

   There are a number of advantages that the X.500 Directory accrues
   simply by virtue of the fact that it is distributed, not limited to a
   single machine. Among these are:

   * An enormously large potential namespace.
     Since the Directory is not limited to a single machine, many
     hundreds of machines can be used to store Directory entries.

   * The ability to allow local administration of local data.
     An organization or group can run a local DSA to master their
     information, facilitating much more accurate data throughout
     the Directory.

   The functionality built into the X.500(88) standard includes:

   * Advanced searching capabilities.
     The Directory supports arbitrarily complex searches at an
     attribute level. As the object classes a specific entry
     belongs to is maintained in the objectClass attribute, this
     also allows Directory searches for specific types of objects.
     Thus, one could search the c=US subtree for anyone with a last
     name beginning with S, who also has either a fax number in the
     (313) area code or an e-mail address ending in umich.edu.
     This feature of X.500 also helps to provide the basic
     functionality for a Yellow Pages service.

        * A uniform namespace with local extensibility.
          The Directory provides a uniform namespace, but local
          specialized directories can also be implemented. Locally
          defined extensions can include new object classes, new
          attributes, and new attribute types.

        * Security issues.
          The X.500 (88) standards define two types of security for
          Directory data: Simple Authentication (which uses passwords),
          and Strong Authentication (which uses cryptographic keys).
          Simple authentication has been widely implemented, strong
          authentication has been less widely implemented. Each of
          these authentication techniques are invoked when a user or
          process attempts a Directory operation through a DUA.

     In addition to the global benefits of the X.500 standard, there are
     many local benefits. One can use their local DSA for company or
     campus wide directory services; for example, the University of
     Michigan is providing all the campus directory services through
     X.500. The DUAs are available for a wide range of platforms,
     including X-Windows systems and Macintoshes.

  3.2.2 Functionality added by QUIPU.

     Functionality beyond the X.500 (88) standard implemented by QUIPU
     includes:

        * Access control lists.
          An access control list is a way to provide security for each
          attribute of an entry. For example, each attribute in a given
          entry can be permitted for detect, compare, read, and modify
          permissions based on the reader's membership in various groups.
          For example, one can specify that some information in a given
          entry is public, some can be read only by members of the
          organization, and some can only be modified by the owner of
          the entry.

        * Replication.
          Replication provides a method whereby frequently accessed
          information in a DSA other than the local one can be kept by
          the local DSA on a "slave" basis, with updates of the "slave"
          data provided automatically by QUIPU from the "master" data
          residing on the foreign DSA. This provides alternate access
          points to that data, and can make searches and retrievals
          more rapid as there is much less overhead in the form or
          network transport.

3.3 Current limitations of the X.500 standard and implementations.

   As flexible and forward looking as X.500 is, it certainly was not
   designed to solve everyone's needs for all time to come. X.500 is not
   a general purpose database, nor is it a Data Base Management System
   (DBMS). X.500 defines no standards for output formats, and it
   certainly doesn't have a report generation capability. The technical
   mechanisms are not yet in place for the Directory to contain
   information about itself, thus new attributes and new attribute types
   are rather slowly distributed (by hand).

   Searches can be slow, for two reasons: a) searches across a widely
   distributed portion of the namespace (c=US, for example) has a delay
   which is partially caused by network transmission times, and can be
   compounded by implementations that cache the partial search returns
   until everyone has reported back, and b) some implementations are
   slow at searching anyway, and this is very sensitive to such things
   as processor speed and available swap space. Another implementation
   "problem" is a tradeoff with security for the Directory: most
   implementations have an administrative limit on the amount of
   information which can be returned for a specific search. For
   example, if a search returns 1000 hits, 20 of those might be
   displayed, with the rest lost. Thus a person performing a large
   search might have to perform a number of small searches. This was
   implemented because an organization might want to make it hard to
   "troll" for the organization's entire database.

   Also, there is at the moment no clear consensus on the ideal shape of
   the DIT, or on the idea structure of the object tree. This can make
   it hard to add to the current corpus of X.500 work, and the number of
   RFCs on various aspects of the X.500 deployment is growing monthly.

   Despite this, however, X.500 is very good at what it was designed to
   do; i.e., to provide primary directory services and "resource
   location" for a wide band oftypes of information.

3.4 Things to be added in X.500 (92).

   The 1988 version of the X.500 standard proved to be quite sufficient
   to start building a Directory Service. However, many of the new
   functions implemented in QUIPU were necessary if the Directory were
   to function in a reasonable manner. X.500 (92) will include
   formalized and standardized versions of those advances, including

   * A formalized replication procedure.

   * Enhanced searching capacities.

   * Formalization of access control mechanisms, including access
     control lists.

   Each of these will provide a richer Directory, but you don't have to
   wait for them! You can become part of the Directory today!

4: WHAT X.500 CAN DO FOR YOU TODAY

4.1 Current applications of X.500

   X.500 is filling Directory Services needs in a large number of
   countries. As a directory to locate people, it is provided in the
   U.S. as the White Pages Pilot Project, run by PSI, and in Europe
   under the PARADISE Project as a series of nation-wide pilots. It is
   also being used by the FOX Project in the United States to provide
   WHOIS services for people and networks, and to provide directories of
   objects as disparate as NIC Profiles and a pilot K-12 Educators
   directory. It is also being investigated for its ability to provide
   resource location facilities and to provide source location for WAIS
   servers. In fact, in almost every area where one could imagine
   needing a directory service (particularly for distributed directory
   services), X.500 is either providing those services or being expanded
   to provide those services.

   In particular, X.500 was envisioned by its creators as providing
   directory services for electronic mail, specifically for X.400. It is
   being used in this fashion today at the University of Michigan:
   everyone at the University has a unified mail address, e.g.
   Chris.Weider@umich.edu. An X.500 server then reroutes that mail to
   the appropriate user's real mail address in a transparent fashion.
   Similarly, Sprint is using X.500 to administrate the address space
   for its internal X.400 mail systems.

   Those of us working on X.500 feel that X.500's strengths lie in
   providing directory services for people and objects, and for
   providing primary resource location for a large number of online
   services. We think that X.500 is a major component (though not the
   only one) of a global Yellow Pages service. We would also like to
   encourage each of you to join your national pilot projects; the more
   coverage we can get, the easier you will be able to find the people
   you need to contact.

5. For Further Information

    For further information, the authors recommend the following
    documents:

        Weider, C., and J. Reynolds, "Executive Introduction to Directory
        Services Using the X.500 Protocol", FYI 13, RFC 1308, ANS, ISI,
        March 1992.

        Lang, R., and R. Wright, Editors, "A Catalog of Available X.500
        Implementations", FYI 11, RFC 1292, SRI International, Lawrence
        Berkeley Laboratory, January 1992.

        Barker, P., and S. Hardcastle-Kille, "The COSINE and Internet
        X.500 Schema", RFC 1274, University College London, November 1991.

        Hardcastle-Kille, S., "Replication Requirements to provide an
        Internet Directory using X.500", RFC 1275, University College
        London, November, 1991.

        Hardcastle-Kille, S., "Replication and Distributed Operations
        extensions to provide an Internet Directory using X.500", RFC
        1276, University College London, November 1991.

        Hardcastle-Kille, S., "Encoding Network Addresses to support
        operation over non-OSI lower layers", RFC 1277, University College
        London, November 1991.

        Hardcastle-Kille, S., " A string encoding of Presentation
        Address", RFC 1278, University College London, November 1991.

        Hardcastle-Kille, S., "X.500 and Domains", RFC 1279, University
        College London, November 1991.

6. Security Considerations

    Security issues are discussed in section 3.

7. Authors' Addresses

    Chris Weider
    Advanced Network and Services, Inc.
    2901 Hubbard G-1
    Ann Arbor, MI 48105-2437

    Phone (313) 663-2482
    E-mail: weider@ans.net

    Joyce K. Reynolds
    Information Sciences Institute
    University of Southern California
    4676 Admirality Way
    Marina del Rey, CA 90292

    Phone: (310) 822-1511
    EMail: jkrey@isi.edu

    Sergio Heker
    JvNCnet
    Princeton University
    6 von Neumann Hall
    Princeton, NJ 08544

    Phone: (609) 258-2400
    Email: heker@nisc.jvnc.net