NEEDS

Nordic Enhanced Educational Directory Service

<needs@uninett.no>

http://www.katalog.uninett.no/needs/

July 25, 2002

Contents

1 About this document

This document is a summary of the main aspects of the "Nordic Enhanced Educational Directory Service" (NEEDS) project. NEEDS has been funded by *Nordunet2*, which is a research programme financed by the Nordic Council of Ministers and by the Nordic Governments [1].

The project started in April 2001, and was due to finish in April 2002. However, it was delayed by three months, and final end of the project was set to June 2002.

Short texts have been written throughout the project period. This document tries to collect the main parts of the technical aspects, and wrap them up in one single document.

2 Overview of NEEDS

Two main goals of NEEDS have been [1]:

- To develop and deploy a common Nordic index based directory infrastructure to facilitate searching for persons in the Nordic academic community
- Provide information and documentation (guidelines) to interested persons in the Nordic academic communities

Main focus has been on how to connect existing directories.

2.1 Connecting LDAP servers

Connecting LDAP servers can be done (the "old way") by including referrals to all of them in one central LDAP server. When a client does a search in the central server it gets back all the referrals and it has to repeat the search in all those distributed servers. Figure 1 shows the "NEEDS way" of doing this. When a client does a search it usually gets back a subset of all the referrals, based on what kind of hits the search in the index resulted in.

Referrals to LDAP servers that are known to give no hits are not returned by the central index server. In most cases the number of servers searched are significantly reduced, and thus the time spent searching would be reduced. Still you have the problem of a "chain not stronger than the weakest link": The search is not done until the slowest response is received, but this solution scales better than the "old way" of connecting the servers.

[1] `http://www.nordunet2.org/`

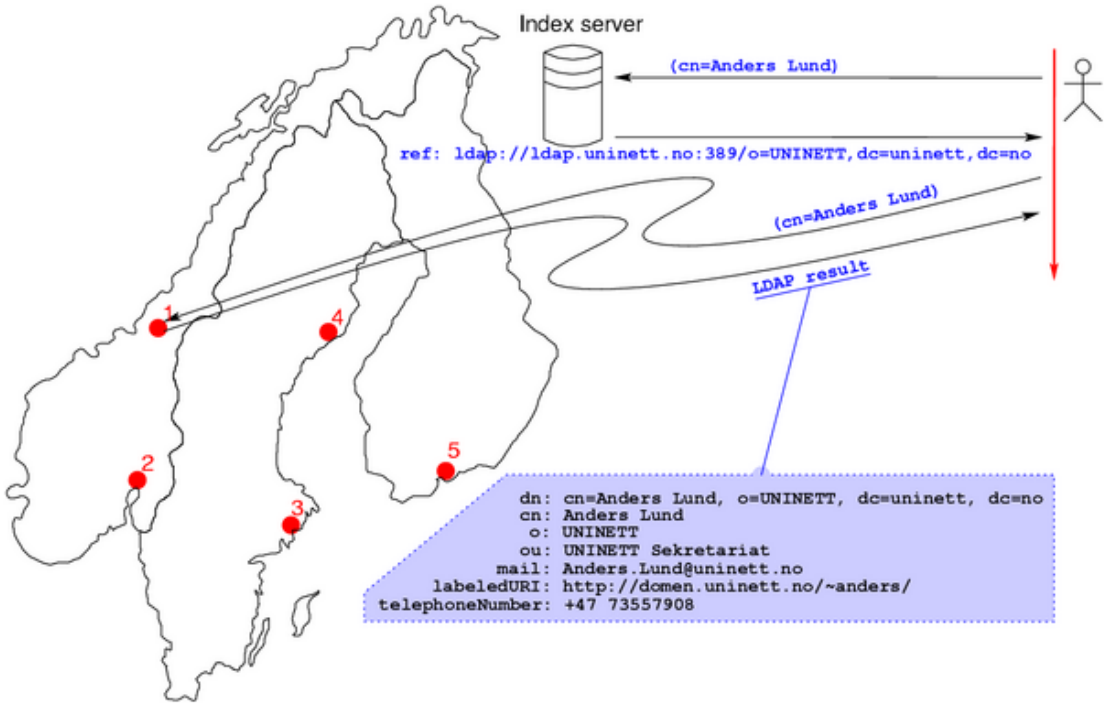

Figure 1: Connecting LDAP servers using a central index server.

2.2 Software used

NEEDS is using software developed by Roland Hedberg (Catalogix [2]). He has developed a index server called `lims`, and to produce data used in the index server `tags` (to produce TIO, which is described in section 5, from LDIF) and `gather` (to harvest/crawl information from operational LDAP servers) have been developed. Roland's comments:

`lims` with its companions `tags` and `gather` is a combination of a LDAP server and a search engine. By the use of `gather` and `tags` you can collect information from LDAP servers and pass them on to the "search engine" which then to any LDAP client appears as a normal LDAP server except for the fact that it never returns information about objects it only returns referrals.

The whole concept is based on the work done by the IETF FIND working group on the Common Indexing Protocol. There are two basic concepts in this work; one is the belief that it would be possible to define a protocol by which one could send any type of index information between index producers and servers and the other is the concept of query routing.

[2] `http://www.catalogix.se/`

Query routing simple means that a client through the use of a sequence of servers/routers can be guided to a information server that has the sought for information.

What are the benefits:

- One well-known access point to publish.
- No need for harmonizing namespaces.
- Given the right index, a query should normally only reach a very limited number of information servers.
- The decisions on access to the information are kept at the source of the information.

Drawbacks

- You have to agree on what should be in the index, which attributes, how to tokenize and so on. The project that has used `lims` so far has used the Tagged index Object as defined in RFC 2654 [2].
- LDAP clients has to be able to handle potentially large number of references.
- The use of the index server adds a finit amount of time to every query.

2.3 The LDAP directories

Before we describe the NEEDS solution in a little more detail, we would like to say something about "how to structure your directory". In the end the users of the NEEDS service will usually end up doing a search in one (or more) of the LDAP servers participating in the total distributed directory service, and how this information should be structured is an important issue. These servers and this information is owned by the end institutions, which in our case are usually universities or similar academic institutions. The next two sections will go through this topic.

3 LDAP data model

Figure 2 shows a sketch of a possible *Directory Information Tree (DIT)*. Using this figure we can define some important keywords to be used when discussing how to structure your information stored in a directory. These definitions are from RFC 2251 [3]:

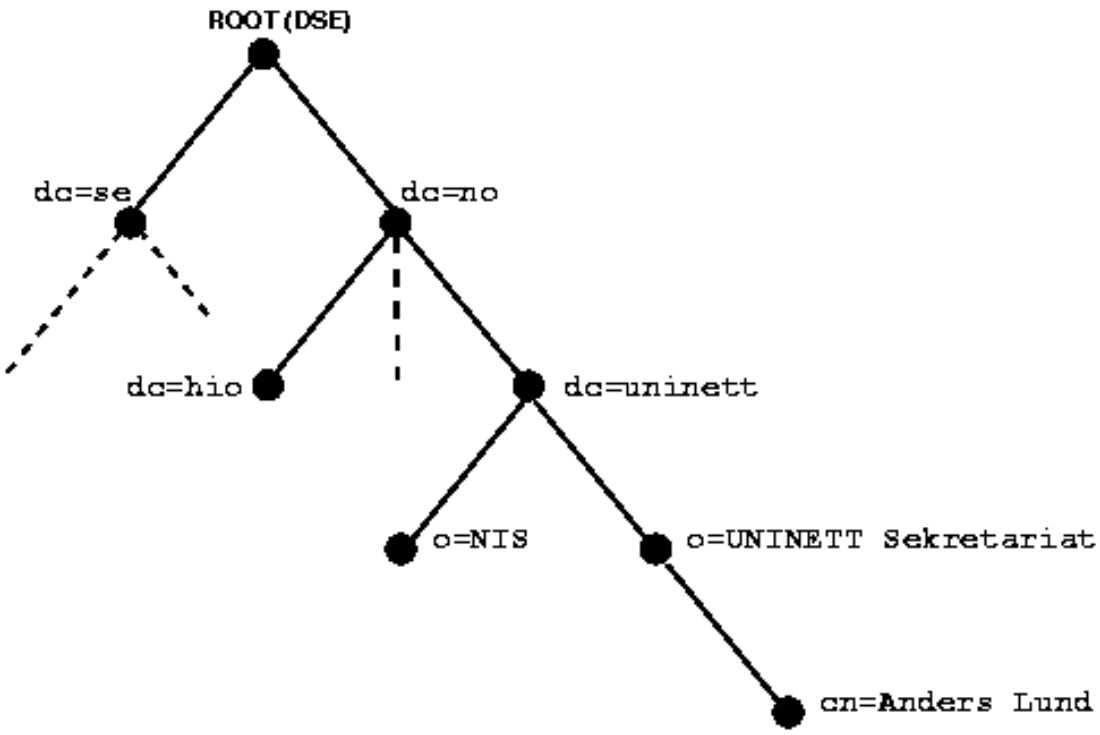

Figure 2: Sketch of a *Directory Information Tree (DIT)*

Entry includes *attributes* and *attribute values*. The directory information tree is made up of these entries.

RDN or *Relative Distinguished Name*, is formed by one or more attribute values from an entry. This name must be unique among all its siblings. In the figure "o=UNINETT Sekretariat" is a possible RDN.

DN or *Distinguished Name*, is "the concatenation of the relative distinguished names of the sequence of entries from a particular entry to an immediate subordinate of the root of the tree". In the figure "o=UNINETT Sekretariat, dc=uninett, dc=no" is one such DN.

Naming Context is defined to be "the largest collection of entries, starting at an entry that is mastered by a particular server, and including all its subordinates and their subordinates, down to the entries which are mastered by different servers".

DSE (DSA-specific Entry) is defined to be the root of the tree, and is not part of any naming context.

DSA is an X.500 term for the directory server.

4 Structuring your directory

There are many ways of structuring your directory information tree and the information stored. "Understanding and deploying LDAP directory services" [4] gives detailed information about this and other issues.

4.1 Considerations

Why is it important to have a good structure of your DIT? Structuring your namespace is important to:

- be ready for future enhancements, such as
  - increased number of entries, but still keep good organization of the data
  - supporting interconnection with global infrastructure of directories
  - supporting automatic systems/applications by offering a structured way of accessing your data
- give users good search interfaces
  - when searching locally within your own organization
  - when searching global interconnected directories
- ease of data maintenance, which could be done by dividing your DIT into different subtrees, and distribute the administration of these trees
- more flexible implementation and maintenance of access control
- presentation of your data to a user browsing the directory could be improved by organizing your data in a well structured hierarchy

There could be more, but these are perhaps the main considerations you have to make.

4.2 Guidelines

It's hard to give absolute solutions to the question "how to structure your DIT". However it is possible to give some guidelines that could be used when establishing your directory service. These guidelines are mainly collected from "Understanding and deploying LDAP directory services" [4] and "A Recipe for Configuring and Operating LDAP Directories" [5].

4.2.1 Naming your root

First of all it is important to choose a root/suffix of your directory. A common mistake is to think only of local users when doing this, instead of thinking of your directory as part of a larger global, interconnected set of directories. It is common today to suggest that directory administrators should use "domain component naming" instead of the original "X.521 naming". Main reason for this is the lack of administration of the "X.521 name space". Domain names are however administrated, and are globally unique. Some software products also requires this kind of naming.

An example:
The domain name of the organization "UNINETT" is `uninett.no`. Traditional "X.521 naming" would name the root as `o=uninett, c=no`, but instead we suggest using the "domain components", i.e. naming your root `dc=uninett, dc=no`.

Using "dc naming" also simplifies the process of finding information using SRV records [6]. SRV records have no information regarding the name of your directory root. This information is needed when searching for persons, and "dc naming" makes it easier.

4.2.2 Namespace hierarchy

Once your root is decided, you should go on designing your namespace hierarchy. It's often said that you should make your namespace as flat as possible. The flatter the "less likely names are to change" [4] (in this context "names" are DNs), and it's also easier to move an object within the organization (you just have to change one component of the name of the object to do so).

Other considerations should however be made when designing your namespace. As mentioned in section 4.1 ease of data maintenance could be done by dividing your DIT into different subtrees, thus making a deeper namespace hierarchy. If on the other hand your organization has a central point of administration, different users get their username/uid from them, and such administration assures that these identifications are unique. If you can trust your users to have unique IDs it is possible to have a flat namespace.

Better access control could also be implemented if you create a more hierarchical tree. If you organize your tree in a hierarchy you don't have to give each object an explicit access control.

4.2.3 Naming your objects

DN and *RDN* were defined earlier in this document. Any object in the tree has an unique DN, and any RDN must be unique among all its siblings. (This also

explains why you can have a flatter namespace if your users have unique IDs.) To achieve unique naming of your objects you could consider

1. Using multivalued RDNs
2. Creating a new subtree, thus expanding the DN of the objects
3. Assigning unique IDs to your objects and use this as your RDN

Using multivalued RDNs "tend to give long, complicated names that change frequently" [4] (also true when creating a new subtree), and this solution is also discouraged because some directory implementations don't fully support it.

The last solution listed (assigning unique IDs) is a better solution, but could be hard to accomplish. If often requires a central administration of your person data, which is not always true at for example a university or college. Using a unique ID as RDN could give a flatter namespace, but if this solution is not working you should consider adding another level to your namespace hierarchy instead of using multivalued RDNs.

Some examples of various RDNs which could be used, with some keywords regarding these solutions:

userid : Could be more readable for users searching your directory, might not be unique across the whole namespace, often not everyone at the organization has such an ID.

serial number : could be student number, usually unique across the whole namespace, not very human readable, could be privacy issues if you're using existing serial numbers.

common name : human readable and easy to remember, not always unique across the whole namespace.

name + serial number : results in long complicated RDNs, some software products don't support multivalued RDNs, could change more often than other solutions.

4.2.4 Attribute values in your entries

Your directory entries should include attributes from standardized schemas. Instead of creating your own schema from scratch you should inherit attributes from well defined standard schemas, and just add necessary new attributes in your local schema.

To improve search capabilities your objects should include information that could be used when searching for a user in a global directory. If for example a person is located at "faculty AA" at "university A", this information should be included in the entry. This could be done by including attribute values for the attribute types "o" and "ou" in your person entries. This solution also improves the search capabilities for users searching in your local directory, even if you have a flat namespace.

If you implement a flat namespace you could consider another tree where you store your information about the organization. This solution facilitates good browsing capabilities for end users looking for information about your organization hierarchy. In this way you could keep the information about your users in a flat namespace, but organize the organizational information in another structured hierarchy. DNs of the objects in this tree tend to keep more unchanged than the names of person objects, and should be more suitable to store in a hierarchy.

5 Tagged Index Objects

The specification of *Tagged Index Objects* (TIOs) can be found in RFC 2654 [2]. From the abstract of this RFC:

"This document defines a mechanism by which information servers can exchange indices of information from their databases by making use of the Common Indexing Protocol (CIP). This document defines the structure of the index information being exchanged, as well as the appropriate meanings for the headers that are defined in the Common Indexing Protocol."

5.1 Format of TIOs

The format used when exporting data from LDAP servers is called *LDAP Data Interchange Format* (LDIF, RFC 2849 [7]), but from this format we generate TIOs which are then used in the central index server. When going from LDIF to TIO information is lost, but we are left with enough information to know where to search for the original information.

The different attribute values in the LDIF file are tokenized according to a "tokenization scheme", as listed in Table 1. In the NEEDS project we have decided to tokenize at least these attributes:

- DNS: "cn", "o", "ou", "givenName" and "objectClass"
- FULL: "sn"

Token Type	Tokenization Characters
FULL	none
TOKEN	white space, "@"
RFC822	white space, ".", "@"
UUCP	white space, "!"
DNS	any character note a number, letter, or "-"

Table 1: Tokenization schemes used in TIOs.

As a result, for example "cn=Anders Lund" is split into "Anders" and "Lund".

In Appendix A you can see an example of what the conversion from LDIF to TIO looks like. Each TIO has a unique *Data Set Identifier* (DSI) which normally is a *Object Identifier* (OID). In our example we have used a OID assigned by UNINETT for the TIO of the UNINETT organization (`1.3.6.1.4.1.2428.50.1`).

5.2 Distribution of TIOs

In principle TIOs should be produced where the LDAP servers are run and maintained, and then collected using some kind of distribution mechanism. This is an issue that we have discussed in the NEEDS project, but a working solution is not in place. Instead ad hoc solutions are used to harvest TIOs (for example FTP or HTTP could be used to fetch TIOs, or the operators of central index servers could fetch LDIF files and generate TIOs themselves).

In Appendix B the requirements of how TIOs should be distributed are included. Work on an implementation of a system called "TIOnet" was started during the project period, but due to lack of time available this work has not concluded. To fulfill the requirements of the privacy laws (see also section 8 later in this document), and to establish contracts with the producers of TIOs, some kind of distribution mechanism needs to be set up. This is an unresolved problem in the NEEDS project.

6 Running the NEEDS service

In the previous sections we have shortly described the different "building blocks" of the NEEDS service. Using these elements we set up a central index server to serve the end users.

6.1 The overall picture

The basic NEEDS service consists of two steps of operation:

1. Producing and collecting *Tagged Index Objects*
2. Loading TIOs into central index server

When operational a user might:

1. Send a LDAP query to the central index server
2. Receive some kind of LDAP referral back if he/she gets a hit
3. Issue a new LDAP query, using the referral received from the central index server
4. Receive some kind of search result from the final LDAP server

as shown in Figure 1.

6.2 Generating TIOs

Using `tags` we can create TIOs from LDIF files. In Appendix C the manual pages of `tags` are included. What attributes to index has already been mentioned in section 5, and the format of the configfile of `tags` is specified in Appendix D.2.

To tokenize correctly the "attribute-section" of the configurefile should include the information shown in Table 2.

cn	DNS	DirectoryString	2.5.4.3	32768
sn	FULL	DirectoryString	2.5.4.4	32768
ou	DNS	DirectoryString	2.5.4.11	32768
o	DNS	DirectoryString	2.5.4.20	32768
givenName	DNS	DirectoryString	2.5.4.42	32768
objectClass	FULL	OID	2.5.4.0	0

Table 2: Tokenization schemes used in NEEDS.

After a configurationfile is written we can index the information given in the LDIF file. To produce the TIO in Appendix A we would run:

```
tags -D 1.3.6.1.4.1.2428.50.1 \
-R ldap://ldap.uninett.no/o=UNINETT,dc=uninett,dc=no \
-c lims.conf < UNINETT.ldif > UNINETT.tio
```

6.3 Running the index server

The `lims` program can load the different TIOs, which have been produced by `tags`, and then users can search in this information using LDAP queries. Appendix D.1 and D.2 include the manual pages of the `lims` program and its corresponding configurationfile (the same as `tags` uses).

Usual way of starting `lims` would be by running:

```
lims -c lims.conf *.tio
```

if you store your TIOs in files with suffix `tio`. What port your index server will listen to is specified in the configurationfile. Default portnumber for LDAP is 389, but one could of course use something else.

6.4 Simple search

Now that we have a running index server, with an LDAP interface, we can issue searches using any LDAPv3 compliant client. Using the command line tool `ldapsearch` from the OpenLDAP [3] software package we can demonstrate this by running for example:

```
ldapsearch -h ldap.uninett.no -p 3891 -b "dc=no" "cn=Anders*"
```

The result is a referral which is returned from the index server:

```
ref: ldap://ldap.uninett.no:389/o=UNINETT,dc=uninett,dc=no
```

Using option `-C` we can tell `ldapsearch` to chase the referral and retrieve the information from the LDAP server:

```
ldapsearch -C -h ldap.uninett.no -p 3891 \
-b "dc=no" "cn=Anders*" cn mail o ou
```

(returning only the attributes cn, mail, o and ou), which gives us:

```
dn: cn=Anders Lund, o=UNINETT, dc=uninett, dc=no
cn: Anders Lund
cn: Anders
o: UNINETT
ou: UNINETT Sekretariat
mail: Anders.Lund@uninett.no
```

The index server is only returning the referral.

[3] `http://www.openldap.org/`

6.5 More advanced search

Using more advanced LDAP search filters we can do more fine-grained searches, but this depends upon what kind of attributes we have indexed.

We can use the UNINETT organization as an example. This organization consists of two organizational units:

- UNINETT Sekretariat
- UNINETT FAS

As already written, we have indexed both attributes cn and ou in the NEEDS service, and this could be used when searching for people within the UNINETT organization. If we issue the search:

```
ldapsearch -C -h ldap.uninett.no -p 3891 \
-b "dc=no" "cn=A*" cn mail o ou
```

we get several hits. Only two of these work in UNINETT Sekretariat, and these two entries can be found using a more advanced search filter:

```
ldapsearch -C -h ldap.uninett.no -p 3891 -b "dc=no" \
"(&(cn=A*)(ou=UNINETT Sekretariat))" cn mail o ou
```

6.6 Different search interfaces

Several different search interfaces can be used to search for information in the index server, as long as the client uses LDAP. A convenient interface for ordinary users is a "web interface". NEEDS offer a web interface, as shown in Figure 3, which can be downloaded from the main homepage (open source code written in PHP [4]). Other interfaces can be:

- E-mail clients with integrated LDAP support.
- WAP enabled devices

7 NEEDS and proxies

The main idea in NEEDS is that a user should be able to search for information about a person, without knowing which organization the person is associated

[4] `http://www.php.net/`

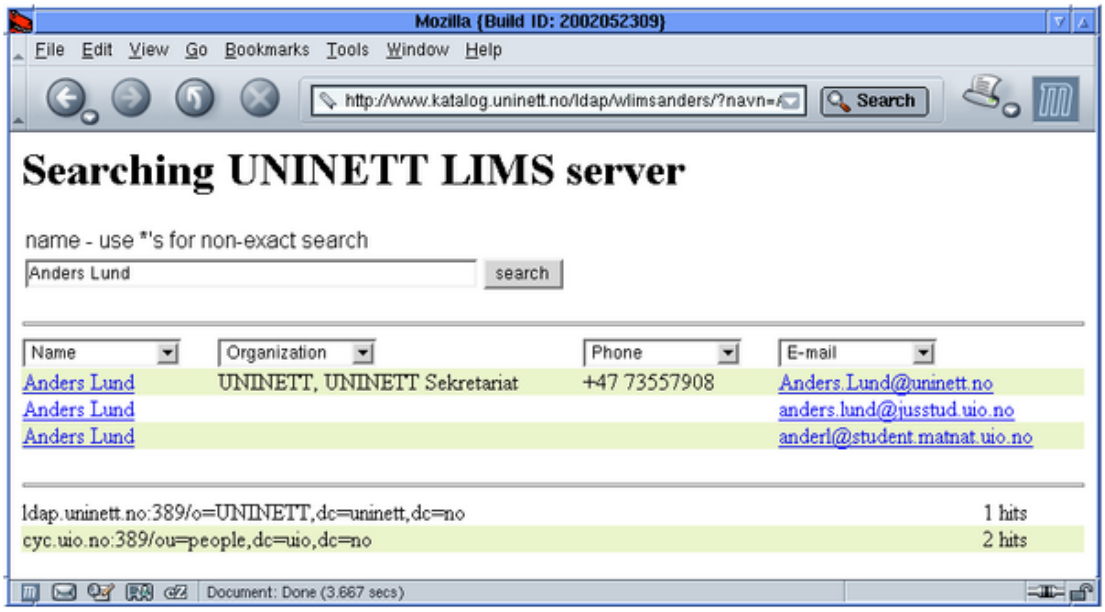

Figure 3: Searching the index server using a web interface.

with. If say only the name of the person is known, there is no way to know which LDAP servers might contain the information. NEEDS uses indexes built from data in all LDAP servers participating in NEEDS, to know which servers are likely to contain the information. To search using the NEEDS system one performs an LDAP search in one of the NEEDS indexing servers, and the server returns LDAP referrals to all servers that are likely to give hits. Referrals are not supported in LDAPv2, thus the NEEDS system requires an LDAPv3 client. An LDAPv3 client would typically chase all the referrals without involving the user, the user only gets to see the final search results. The servers that the NEEDS server points to, may also give referrals, and the client would also chase those. An LDAPv3 client could alternatively present the referrals to the user, and make the user decide whether they, and possibly which, should be chased.

NEEDS also offers a web interface that can perform searches. This is useful for users that prefer to use the web, or do not have access to an LDAPv3 client. When using the web interface, the web server is acting as an LDAPv3 client, so LDAPv3 is still used.

There are a number of people still using LDAPv2 clients. The LDAPv2 client cannot make use of NEEDS directly. It could however be possible to access NEEDS servers through an LDAP proxy that uses LDAPv2 and LDAPv3 to communicate with client and server respectively. It would also need to chase referrals. The LDAPv2 client would only communicate with the proxy. To the

client the proxy looks like a normal LDAPv2 server which contains all the data that exists in all the LDAP servers that take part in NEEDS. Another issue is character sets. LDAPv2 clients expect a specific character set, and the proxy would have to convert the data received from the LDAPv3 servers, which is UTF-8, to the character set expected by the client. The client might expect UTF-8, but clients would often expect ISO 8859-1, T.61 or something else. There is no standard way for the client to specify which character set it wants, or for the server to tell what character set is used. An easy solution would be to have separate proxies for each character set, and configure clients to use a proxy supporting the preferred character set.

The biggest problems with such a proxy are perhaps performance and scalability. One search from a single client might last a relatively long time, tens of seconds is common. One would typically chase all referrals in parallel, so the time depends on the slowest LDAP server. In reality this can be more complex since referred servers can also return referrals. During the search there will be an open TCP session to the client. There will also be a TCP session to the NEEDS server part of the time, this might be shared for multiple clients. For each referral returned by the NEEDS server there will be opened a TCP session, these will all be opened at the same time, but each can be closed as soon as the respective search is finished, so if there is only a few slow servers, most of the sessions will be closed quickly. The behavior here is much the same as for an LDAPv3 client chasing referrals, but there might be a large number of LDAPv2 clients using the proxy at the same time, and then the number of TCP sessions might become a problem. Memory and CPU usage on the proxy may not be that much of a problem on a modern computer, but the search would in most cases, depending a bit on the clients performance and network connectivity, take longer time than if the client chased the referrals itself. If character set conversion is done, that will of course also affect the CPU usage. The biggest problem with character set conversion is perhaps that some schema awareness is necessary to know which data to convert and which to not.

Many LDAPv3 implementations are now available and the LDAPv2 IETF standards will probably change status to historic soon. We expect that most LDAPv2 users will move to LDAPv3 in the near future, although some are tied to legacy applications that only support LDAPv2. A proxy would be useful to those who for some reason use LDAPv2 clients, the alternative is to use a separate LDAPv3 client or a web interface when the NEEDS service is needed.

Today it is not worth the effort to develop such a proxy just for NEEDS. But it might be of interest to test such proxies if any exists. Before this is considered one should fully deploy the NEEDS service, and through those experiences be sure that it works well for LDAPv3, and that the service is of sufficient interest to the LDAPv2 users. By then it is likely that very few LDAPv2 users remain.

8 Privacy issues

Exchanging TIOs and running a central index server leeds to a series of questions regarding privacy issues that needs to be resolved. Within the NEEDS project a short report has been written by Walter M. Tveter [8], in which the different aspects of this issue are discussed. From the introduction:

"This paper examines the Directive 95/46/EC and the laws implementing it in the Nordic countries, with a view to its significance for directory services.

Where each entity is located and counducts it's business in just one country, one can use national law, and any questions will be decided by the states national regulatory authority. In more complex cases, such as the NEEDS project, it is important to analyze the consequences of the mixing of different laws, as will happen if an operator operates in another country.

A lot of the terms in both the Directive and in the different national laws are as of now unclear. A thing to be aware of is that the final say in interpretation is not the different supreme courts in the Nordic countries, but the EU-court. One may see different interpretations of terms in different countries the first years until things are clarified by the EU-court.

The basis of the discussions in this paper will be the Directive 95/46/EC. Since this in many ways is the framework of the different national legislations, and since they do not vary that much from it, the rules are presented in the form they have in the directive, and then national deviations from this are commented.

The different national legislations are viewed in light of their laws. Regulations, preliminary legislative texts and directions have so far only superficially been examined."

Drafts of agreements between end organizations and *Nathional Research Network* (NRN), and between NRNs, that participate in exchanging TIOs are also included. In Appendix E a refined version af a draft "agreement concerning access to and usage of personal data" between an end organization and a NRN is included.

9 Future use

Another kind of future use (in addition to the common "white pages") of a service like NEEDS and software like `lims` could be to let automated systems search for information used when authorizing and authenticating users. One example of this

is web authentication, using the Apache [5] web server and a LDAP authentication module. When doing authentication the module will use some kind of username and password, submitted by the user, and then:

1. Search the central index server for the submitted username.
2. Use the referral it gets back from the index server to do a bind to authenticate, using the password submitted by the user.

Using LDAP and structuring your data according to schemas, you have laid the groundwork for such services.

[5] `http://www.apache.org/`

References

[1] Stig Venås. Nordic enhanced educational directory service. `http://www.katalog.uninett.no/needs/full_application-2001-02-22.html`. Full application.

[2] R. Hedberg, B. Greenblatt, R. Moats, and M. Wahl. A tagged index object for use in the common indexing protocol. `ftp://ftp.nordu.net/rfc/rfc2654.txt`. RFC2654.

[3] M. Wahl, T. Howes, and S. Kille. Lightweight directory access protocol (v3). `ftp://ftp.nordu.net/rfc/rfc2251.txt`. RFC2251.

[4] Timothy A. Howes, Mark C. Smith, and Gordon S. Good. *Understanding and deploying LDAP directory services.* Macmillan Network Architecture and Development, 1999.

[5] Michael R. Gettes. A recipe for configuring and operating ldap directories. `http://www.georgetown.edu/giia/internet2/ldap-recipe/`.

[6] A. Gulbrandsen, P. Vixie, and L. Esibov. A dns rr for specifying the location of services (dns srv). `ftp://ftp.nordu.net/rfc/rfc2782.txt`. RFC2782.

[7] G. Good. The ldap data interchange format (ldif) - technical specification. `ftp://ftp.nordu.net/rfc/rfc2849.txt`. RFC2849.

[8] Walter M. Tveter. Privacy aspects of the needs project. `http://www.katalog.uninett.no/needs/NEEDS_privacy.pdf`.

A LDIF to TIO example

A short, simple example of a LDIF file and a corresponding TIO.

A.1 UNINETT.ldif

```
dn: cn=Anders Lund, o=UNINETT, dc=uninett, dc=no
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Anders Lund
cn: Anders
sn: Lund
givenName: Anders
o: UNINETT
ou: UNINETT Sekretariat

dn:: Y249U3RpZyBWZW7DpXMsIG89VU5JTkVUVCwgZGM9dW5pbmV0dCwgZGM9bm8=
cn:: U3RpZyBWZW7DpXM=
cn: Stig
sn:: VmVuw6Vz
givenName: Stig
o: UNINETT
ou: UNINETT Sekretariat
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: geoPosObject
```

A.2 UNINETT.tio

```
Content-Type: application/index.obj.tagged;dsi=1.3.6.1.4.1.2428.50.1;
base-uri="ldap://ldap.uninett.no/o=UNINETT,dc=uninett,dc=no"

version: x-tagged-index-1
updatetype: total
thisupdate: 1022239986
BEGIN IO-schema
cn: DNS
sn: FULL
ou: DNS
o: DNS
givenName: DNS
objectClass: FULL
END IO-schema
BEGIN Index-Info
cn: 0/ven%c3%a5s
-0/stig
-1/lund
-1/anders
sn: 0/ven%c3%a5s
-1/lund
ou: 0-1/uninett
-0-1/sekretariat
o: 0-1/uninett
givenName: 0/stig
-1/anders
objectClass: 0-1/top
-0-1/person
-0-1/organizationalperson
-0-1/inetorgperson
-0/geoposobject
END Index-Info
```

B TIOnet requirements

Requirements for the NEEDS TIO distribution mechanism

1. Background

The common indexing protocol [RFC2651], [RFC2652], [RFC2653] and [RFC2654] specifies both a data format (TIO) and transport protocols for index information for (among other protocols) LDAP. The NEEDS project uses TIO objects described in [RFC2654] and the LIMS ldap index server to provide a distributed index of ldap servers. Similar project for instance FEIDE and TF-LSD have similar objectives.

Using the protocols for CIP transport described in [RFC2653] is one alternative solution to the problem of distributing the TIO objects. There are other possible solutions to this problem and this document describes the requirements of the NEEDS project on the TIO distribution mechanism.

2. Requirements

2.1 Updates and TIO identification

2.1.1 Full vs partial updates

NEEDS does not require partial updates of TIO objects. That is to say all TIO objects distributed to index servers are expected to be full updates. In order to uniquely identify TIOs for the purpouse of replacing one TIO with another regardless of the LDAP url where referrals are sent, each TIO must have a unique and fixed DSI. Further- more an empty TIO must be equivalent to deleting the corresponding TIO.

2.1.2 TIO Expiration

Index objects in the index servers must be kept in close synchronization with the original data. To achieve that the TIOs distribution mechanism must be able to inform index servers about

expiration time associated with the data in a given TIO object. This expiration time must be used by the index server to remove TIOs which have not been updated in time. I.e in the NEEDS project, absent data is better than erroneous data.

2.1.3 Loop detection

Since the structure of the set of index servers and index object providers may not be tree-like the TIO delivery mechanism must be able to detect loops.

2.1.4 Scoping

The infrastructure for TIO distribution may be shared among several partially over- lapping index infrastructures. For instance some of the index data found in the NEEDS index servers may be included in a European network. To facilitate this it should be possible to include information about scope (for instance "world", "local", "eu", "nordic") in the TIO distribution mechanism.

2.2 Security

All TIO objects must be transported over a channel which provides integrity and privacy. Scoping raises some security issues, however it is believed that leaving scoping up to the receiving party may be sufficient initially. However the TIO distribution mechanism should be able to support some form of secure scoping.

C tags manpage

TAGS(1) TAGS(1)

NAME

tags - Creates Tagged Index Object files from LDIF files.

SYNOPSIS

tags [-t type] [-d debuglevel] [-c configfile] [-s char type] [-D DSI] [-R Refs] [-v] < inputfile > outputfile

DESCRIPTION

Tags is a program that takes as input a LDAP Data Inter change Format (LDIF) file as specified in RFC 2849 and produces as output a Tagged Index Object File as defined in RFC 2654. It's made to work together with lims(1) by the same author.

OPTIONS

The following options are recognized:

-d debuglevel

Turn on debugging as defined by debuglevel. Presently does not really distinguish between dif ferent level but might do so sometime soon.

-c configfile

Tells tags where the configuration file can be found. The format of the configuration file is described in lims.conf(5). If no file is specified tags expects to find the configuration file at /etc/lims/lims.conf.

-s chartype

tags accepts two different character sets in the input LDIF file, it can either be UTF-8 or ISO-8859-1. If UTF-8 is used (this is the default) then according to RFC any attribute values that contains non-ASCII characters MUST be base64

encoded. If on the other hand ISO-8859-1 is specified then this limitation is not necessary. This option is a historical artifact and may go away in the future.

-D DSI This is where you define the Data Set Identifier (DSI) which should be worldwide unique for the dataset you are indexing. Normally a DSI is a ObjectIdentifier.

-R Refs
Apart from the DSI you also have to specify where one can find a network accessible 'database' which contains the original information from which the LDIF information was constructed. Normally these refs are LDAPURLs (RFC 2255), but there is nothing stopping you from defining some other type of URL. If you want to specify more than one URL you can do that, the different URL then has to be separated by a space character. Because of this if a URL contains a space character it has to be escaped, tags expects this to be done the HTML way, that is '%20' instead of <space>.

-v Print the version of the program. The program will close down after doing this.

EXAMPLE

Typically you will use tags in this way:

```
tags -c lims.conf -D 1.2.752.58.46.1 \
-R "ldap://ldap.catalogix.se/dc=catalogix,dc=se" \
< catalogix.ldif > catalogix.tio
```

Or if have more than one ldapserver that handles the same dataset.

```
tags -c lims.conf -D 1.2.752.58.46 \
-R "ldap://ldap.catalogix.se/dc=se ldap://ldap.sunet.se/dc=se" \
< se.ldif > se.tio
```

SEE ALSO

lims(1), lims.conf(5)

AUTHOR

Roland Hedberg, roland@catalogix.se

D lims manpages

D.1 lims

```
LIMS(1)                                                    LIMS(1)


NAME
       lims - Stand-alone Index based LDAP Daemon

SYNOPSIS
       lims [-d debuglevel] [-c configfile] inputfiles

DESCRIPTION
       Lims  is a program that takes as input Tagged Index Object
       (TIO) files as defined in RFC 2654.  It  will  listen  for
       LDAP  connections  on  a  specified  port ( default 389 ),
       responding to the LDAP operations it receives  over  these
       connections.  Lims differs from other LDAP servers in that
       it is a read-only server and that it only  returns  refer
       rals.

       Upon  startup,  it normally forks and disassociates itself
       from the nvoking tty. If the -d flag is given,  lims  will
       not disassociate from the invoking tty.

       The  TIO  files needed as input can be produced by tags(1)
       by the same author.


OPTIONS
       The following options are recognized:

       -d debuglevel
              Turn on debugging as  defined  by  debuglevel.   If
              this  option  is  specified,  lims will not fork or
              disassociate from the invoking terminal. Some  gen
              eral  operation and status messages are printed for
              any value of debuglevel.  Debuglevel is taken as  a
              bit  string,  with each bit corresponding to a dif
              ferent kind of debugging operation.
```

-c configfile
Tells lims where the configuration file can be found. The format of the configuration file is described in lims.conf(5). Default is /etc/lims/lims.conf.

EXAMPLE

Typically you will use lims in this way:

```
lims *.tio
```

To start with a alternate configuration file, and turn on copius debugging which will be printed to standard error, type:

```
lims -c lims.conf -d 255 *.tio
```

SEE ALSO

tags(1), lims.conf(5)

AUTHOR

Roland Hedberg, roland@catalogix.se

Catalogix 25 April 2002 1

D.2 lims.conf

```
lims.conf(5)                                              lims.conf(5)


NAME
       lims.conf  -  configuration  file  for  the  tio  producer
       tags(1) and the stand-alone LDAP daemon lims(1).

SYNOPSIS
       /etc/lims/lims.conf

DESCRIPTION
       This configuration file contains configuration information
       for  both the lims(1) daemon and the tio producer tags(1).
       The lims.conf file contains a  series  of  sections.  Each
       section  with a tag and then a number of lines with infor
       mation pertaining to that section.

       Blank lines and comment lines beginning with a #  charcter
       are ignored.

       The specific configuration options available are discussed
       below.

TOKENTYPE
       For a full background  on  tokenization  you  should  read
       RFC2654.  So I am not going to go through it all in detail
       here. Basically it's all about splitting strings into sub
       strings(  tokens ). Where this splitting occurs depends on
       which characters one uses as tokenization characters. If a
       email  address  is  to  be  split into it's components one
       might use '@' as  the  tokenization  character.  A  finer
       grained  tokenization  will  require  '.' to also be used.
       Hence with the use of '@' and '.' as tokenization  charac
       ters  the  email  address  "roland@catalogix.se"  would be
       split into the following  three  tokens  "roland",  "cata
       logix" and "se".

       Different  data  requires  different tokenization, even if
       they are of the same type there might be reasons for doing
```

it differently. So in this section the different tokeniza tion types (the sets of tokenization characters) are defined.

The format of the definition is :
<name> <char1> <char2> .... <charN>

Is char is not a printable ascii character, that is out side the range 0x20 - 0xff it has to be presented as 0xNN where NN is the hexcode of the character. char also has to be in the UTF-8 set of characters. If the character is a multibyte UTF-8 character is must be specified as 0xNNNN, 0xNNNNNN or 0xNNNNNNNN depending on whether it is a 2,3 or 4-bytes character.

Multicharacter tokenizations 'characters' can also be specified. They must be specified in the same way as multibyte UTF-8 characters that is by specifying them as 0xNNMMOOPP...

SCOPE

In the case where there are indexing domains served by sets of lims servers that overlap, there might be reasons for not allowing the propagation/usage of index information that has been received from LDAP servers within one domain, in another domain. The use of scope is a simple way of telling the lims server which index object it should use.

CONTEXT

It's not yet very common that LDAP clients ask the LDAP server which naming contexts it serves. But if it does the contexts defined in this section is what the lims server returns as it's naming contexts. You can specify as many as you like.

TIMEOUT

How long the lims server should wait before it will close the connection to a inactive client. This is to disallow clients hanging about, just in case they get the need to ask a query.

PORT

Normally LDAP servers listens on port 389, there might be very good reasons for having the lims daemon listing on another port.

UNICODE FILE

When dealing with matching on unicode strings, the soft ware has to normalize the unicode strings before any com parision can be attempted, This file contains information that is necessary for the functionality.

ATTRIBUTE

A list of attribute definitions, each one ont the format:

<name> <tokenizationtype> <attributesyntax> <oid> <maxsize>

Name must be a LDAP attribute name

tokenizationtype must be one of the tokenizationstypes defined earlier in the file

attributesytax has to be one of directorystring or IA5 these two corresponds to "Directory String" and "IA5 String" as defined in the ITU-T standards.

oid is the object identifier of the LDAP attribute.

maxsize is the maximum size of a attribute value of the attribute. This is really the maximum.

LOG

Where tags should write the logs. If nothing is specified tags will attempt to write to /var/log/lims.log.

SEARCHBASECHECK

Has to possible values: strict or lax. If strict is defined (this is the default) then the searchbase will be check against the distingusihed name defined in the referral URL and if it is not less specific that referral will not be returned.

If a search has the searchbase dc=SE and the referral URL has the searchbase c=NO it will not be returned. On the other hand if it is dc=catalogix,dc=SE it will be

returned.

If the check is defined to be lax then no such check is performed. The obvious benefit of this is of course that in some instances you would like a set of LDAP servers to appear outward as if the represent one community even if they by administrative of technical reasons all are using naming contexts that are in totally other parts of the tree.

SSMIN

The minimum number of characters that are allowed in a substring search filter (default is 2). Substring searches are costly and in some cases completely unnecessary. If you don't want to allow substring searches at all define this to be '0'.

MAXREFERRALS

The maximum number of referrals that is returned. It is worth noticing that the client has no way to specify how many referrals he is prepared to handle. The size limita tions he can specify all has to do with attributes not referrals. The is no default.

EXAMPLE

```
[tokentype]
TOKEN @ 0x09 0x0A 0x20
RFC822 @ . 0x09 0x0A
UUCP ! 0x09 0x0A 0x20
DNS ! #  $ % & ? ( ) + , . / : ; < = > ? @ [ ] ^ _ ‘ { | } ~ 0x20 0x5c 0x
22 0x09 0x0D 0x0A
FULL

[scope]
nordunet

[context]
c=fi
dc=se
dc=no

[timeout]
30
```

```
[port]
1389

[unicodefile]
/usr/local/etc/UnicodeData.txt

[attribute]
cn DNS DirectoryString 2.5.4.3 32768
sn FULL DirectoryString 2.5.4.4 32768
c DNS DirectoryString 2.5.4.6 32768
l DNS DirectoryString 2.5.4.7 32768
ou DNS DirectoryString 2.5.4.11 32768
o DNS DirectoryString 2.5.4.20 32768
gn DNS DirectoryString 2.5.4.42 32768
mail RFC822 IA5 0.9.2342.19200300.100.1.3 256
dc DNS IA5 0.9.2342.19200300.100.1.25 32768
objectClass FULL OID 2.5.4.0 0

[log]
lims.log

[searchbasecheck]
strict

[ssmin]
2

[maxreferrals]
25
```

SEE ALSO

lims(1), tags(1)

AUTHOR

Roland Hedberg, roland@catalogix.se

E Draft agreement

Agreement concerning the access to and usage of personal data about employees and students of

.....

for specific use in the electronic directory services of the Nordic Enhanced Educational Directory Service (NEEDS) project.

This agreement exists between

..... (ORG)

and

..... (NRN)

1.0 Scope
1.1 This agreement regulates the access to and usage of information about employees and students of ORG for the above mentioned electronic directory service, between the above parts.

1.2 This agreement covers all usage of such information, at all times.

1.3 This agreement exists between ORG as a controller and NRN as a processor in the sense these terms are given in the "Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data", with the understanding of both parties that ORG as the controller bears the responsibility for securing the necessary grounds for the processing of personal information as described in this agreement.

2.0 Duties of ORG
2.1 ORG shall provide NRN access to the data through a channel which provides integrity and privacy. ORG shall ensure that the data supplied is as correct as is possible, using reasonable resources.

2.2 All information to be used by NRN shall be provided by ORG in

accordance with the provisions in Appendix A, and in close colaboration with the NRN.

3.0 Duties of NRN
3.1 NRN collects data from ORG through a channel which provides integrity and privacy. The information shall be collected, and the NRN's data updated in accordance with the provisions in Appendix A.

3.2 NRN shall provide the directory service to end users without prejudice, without charge and at all times of day.

3.3 NRN shall send a report log to ORG stating when its index server has collected and updated its information concerning ORG every three months.

4.0 Liability and damages
4.1 Any liability for providing access to information shall be minimized by the clear informing of any and all end users, that neither NRN nor ORG will guarantee, or be held responsible for the accuracy or correctness of the data.

4.2 Any and all liability for the providing of incorrect information where this information is available updated and corrected from ORG, will be on the part of NRN.

5.0 Forwarding data
5.1 NRN can forward, exchange or mirror information from ORG only to national research networks that shall use the information solely for purposes relating to the NEEDS project.

5.2 To perform such an action, NRN shall provide that it through contractual means has acquired the right to retract allready given information, should the grounds for giving it (powers derived from ORG) vanish.

5.3 Whenever NRN wishes to forward, exchange or mirror its information, notice shall be given to ORG at least 14 days in advance.

5.4 Given such notice as described in 5.2, ORG can, for any reason, deny NRN the ability to do this.

5.5 If NRN has forwarded, exchanged or mirrored information, ORG can

at any time, for any given reason, demand this to be stopped, and for NRN to retract any given information.

6.0 Security issues
6.1 All handling, publishing and processing of information must be done in accordance with the provisions in Appendix A.

7.0 Alterations of the agreement
7.1 Alterations in this agreement shall be made with regard to, and if possible in accordance with, other agreements between NRN and other controllers concerning the NEEDS project.

8.0 The duration of the agreement, termination of the agreement
8.1 This agreement can at any time be terminated by any of the two parties, with three months written notice.

Date/Signature (on behalf of ORG)	Date/Signature (on behalf of NRN)

(Appendix A)

Handling, publishing and processing of information in the Nordic Enhanced Educational Directory Service (NEEDS) project

The "Nordic Enhanced Educational Directory Service" (NEEDS) consists of two main parts of operation:

1. Distribution of tagged index objects (TIOs) to be used in central index server
2. Running an updated central index server

There should also be a search interface for end users, but part 2 includes support for LDAP. This means that you could use different LDAPv3 clients for searching, and in this way publish information. NEEDS will also deliver a web interface which could be used.

The index server includes referrals to LDAP servers which should be version 3 compliant.

1. Distribution of TIOs

Key words for this part of the NEEDS service:
* All TIO objects must be transported over a channel which provides integrity and privacy.
* All TIO objects distributed to index servers are expected to be full updates.
* Distributing an empty TIO must be equivalent to deleting the corresponding TIO.
* The distribution mechanism must be able to inform index servers about expiration time associated with the data in a given TIO object.

2. Running the index server

The index server contains tagged index objects. In the NEEDS project it has been decided that these attributes should be indexed and made searchable:
* cn
* sn
* givenName
* o
* ou
* objectClass (the primary ones, "person", "role", etc.)

Key words for this part of the NEEDS service:
* Information contained in the index servers is refreshed in accordance with the update frequencies of the original directory server, and should be refreshed in a timely manner, so that the data is not outdated.
* NEEDS does not require partial updates, i.e. all information should be refreshed.