#19

IndeX.500

David Chadwick

This paper was prepared by David Chadwick in his role as consultant to DANTE in the area of Directories - May 1996.

IndeX.500

David Chadwick

Introduction

This paper is a result of the talk on Whois++ given by Alan Emtage to the NameFLOW-Paradise meeting at ULCC in September 1995. The similarities between the two services, X.500 and Whois++, became obvious during the presentation, as did the main distinction, and advantage, which Whois++ might at first sight appear to have over X.500. This is the ability of Whois++ to create centroids and pass these up to index servers, thereby allowing a Whois++ reference hierarchy to be built up from any attribute types, rather than simply from the naming attributes as used in X.500.

This paper presents a way of building up X.500 centroids and index servers, using standard X.500 (88) protocols, and so should allow the NameFLOW-Paradise community to easily gain experimental evidence of the advantages, if any, of such an approach.

Whilst the X.500 protocols do not need to be changed, either the DUAs or the DSAs will need to be enhanced in order to be able to correctly use the index entries. Two alternative strategies are supported, one that requires the DUAs to be enhanced, and one that requires the DSAs to be enhanced.

Terminology

This paper uses the following terminology:

> Index Entry - an entry which contains indexing information about the entries held in a particular DSA. Index entries will typically be created by the administrator of a DSA.
>
> Centroid DSA - a DSA which collects together index entries from other DSAs. The collection may be performed by actually copying index entries from other DSAs, or by dynamically building them using Search operations.

David Chadwick works for the IT Institute of the University of Salford. He produced this paper in his role as consultant to DANTE. His e-mail address is D.W.Chadwick@iti.salford.ac.uk

Index DSAs

A previous paper by Paul Barker [1] suggested the creation of Index DSAs as a means of improving the performance of X.500 Searches. However, Paul's proposed solution required the support of a replication protocol - either 93 shadowing [2] or RFC1276 [3] - by the DSAs. This paper supports most of the reasons suggested by Paul for having Index DSAs, but proposes a simpler solution that more closely mirrors that of Whois++. The primary difference is that:

> whereas Paul's Index DSAs hold copies of entries along with their searchable attributes (implying that the DNs of all the indexed entries are held in the Index DSAs);
>
> the Centroid DSA proposed here, holds the names of server DSAs and the lists of attribute values that they hold. This more closely matches the design of Whois++ centroids.

A Centroid DSA is a DSA that holds a special type of entry, of object class index (and its subclasses). An index entry is initially created by a server DSA, as an index of one or more of its attributes, and is then copied by a Centroid DSA.

Information content of Index Entries

The information required from an index entry is closely related to the information content of a Continuation Reference. This can be summarised as:

> the name and address (Access Point) of the DSA being referenced
>
> the context prefix of the naming context being referenced.

Note. The information required from an index entry in a multi-protocol directory model might be: the protocol needed to contact the directory server being referenced, and protocol specific information needed by the referenced directory server.

In addition, an index entry needs to contain the set of attributes that it indexes. These are the attributes that will be searched as a result of user queries.

The simplicity of this design is that all the attribute syntaxes defined for an index entry i.e. distinguishedNameSyntax, presentationAddress Syntax (or printableStringSyntax ?) and those of the attributes being indexed, already exist. The formal definition of the index object class is given in Appendix 1.

Naming Index entries

There are potentially many different ways of naming index entries. This is probably the most difficult decision to make, since their is no single correct answer. The name can be drawn from the information content of the index entry, or it can be completely unrelated to the information above. Examples include:

use the distinguished name of the DSA that the index points to;

use the context prefix of the DIT subtree that is indexed;

use an arbitrary prefix name, such as O=index, which points to part of the local DIT where all the indexes are held;

use an attribute which differentiates between the types of indices.

As an aid to making the decision, it should be remembered that index entries are held in two different places - in the server DSA that initially creates the index, and in a Centroid DSA that collects the indexes.

This paper suggests the following naming mechanism for index entries, and that the same naming mechanism is used in both the server DSA and the Centroid DSA.

In the server DSA that creates the index

1. When an index entry is created, it is held directly beneath the context prefix entry of the naming context that it is an index to.

2. Since a naming context can have multiple indexes to it, each index entry will have an RDN that uniquely identifies it. The RDN will be composed of the CommonName attribute only, e.g. CommonName=SurnameIndex.

In the Centroid DSA that copies the index

1. When an index entry is copied to a Centroid DSA, it is held in exactly the same place. This may mean that the Centroid DSA has to create new non-leaf nodes in its DSA Information Tree.

(Note that by including the context prefix name in the name of each index entry, there is no need to store the context prefix as an attribute within the index entry.)

Of course, both the server DSA and the Centroid DSA can be the same DSA, i.e. the server DSA can use its own index.

Merging Index Entries

If a Centroid DSA wants to merge together the index entries that it has collected, this is seen to be primarily of benefit to other Centroid DSAs at higher levels of the DIT. Merging index entries together(as opposed to simply collecting index entries and leaving them *as is*) can be done in one of two ways: merge the index entries keeping the original pointers to the server DSAs, or merge them and point to the Centroid DSA that did the merging i.e. create a hierarchy of Centroid DSAs.

The former method will add complexity to the attribute syntaxes, and hence implementation effort. The tuples 'searchable attributes : context prefix : DSA access Point' will need to be kept together, and so merging indexes in this way will require more complex attribute syntaxes to be defined and implemented.

On the other hand, merging indexes and changing the reference information to point to the Centroid DSA that did the merging will cause a significant performance penalty during use, since an extra link in the query chain will be needed in order to find the actual entries that are indexed. For this reason, merging of indexes is not recommended. An example of a merged index is given in Figure 1.

Using Index Entries

Index entries will need to be understood and recognised in order for them to be of use to the X.500 service. Either the DUAs or the DSAs (or both) will need to be enhanced in order to be able to correctly use the index entries. Two alternative strategies are supported, one that requires the DUAs to be enhanced, and one that requires the DSAs to be enhanced.

Strategy A - Enhance the Centroid DSAs

With this strategy, Centroid DSAs understand that index entries are special entries. When information is retrieved from an index entry, it is turned into a ContinuationReference before returning it to the requestor. (Alternatively the Centroid DSA could act upon the continuation reference and chain a request to the referenced DSA.)

Strategy B - Enhance the DUAs

With this strategy, DUAs understand that index entries are special entries. When information is retrieved from an index entry it should be treated as a ContinuationReference, and a subsequent request issued.

An example of index entries

Figure 1, shows (part of) the DSA Information Trees for 3 DSAs. *ABC DSA* holds the O=ABC naming context, *XYZ DSA* holds the O=XYZ naming context, and *GB DSA* holds the C=GB DSA. The administrators of both ABC and XYZ DSAs have decided to create two indexes, namely, an Organisational Units index and a Surnames index. The former index will hold a list of the organisational units held in the DSA, and the latter will hold a list of surnames. Both of these lists will be held in the Organisational Unit Name or Surnames attribute respectively.

Suppose that the ABC DSA holds two organisational units (ITI and Electrical Engineering) with organisational people of surname=Chadwick in the ITI department, surname=Chadwick in the Electrical Engineering department, and surname=Larmouth in the ITI. The surname index entry would hold the following information:

DSA Access Point: DSA Name: C=GB, O=ABC DSA; Presentation Address: string representation of the PSAP of ABC DSA
Surname: Chadwick, Larmouth
CommonName: SurnameIndex

An index entry which indexes on the organisational unit name attribute would hold the following information:

DSA Access Point: DSA Name: C=GB, O=ABC DSA; Presentation Address: string representation of the PSAP of ABC DSA
Organisational Unit Name: ITI, Electrical Engineering
CommonName: OrgUnitIndex

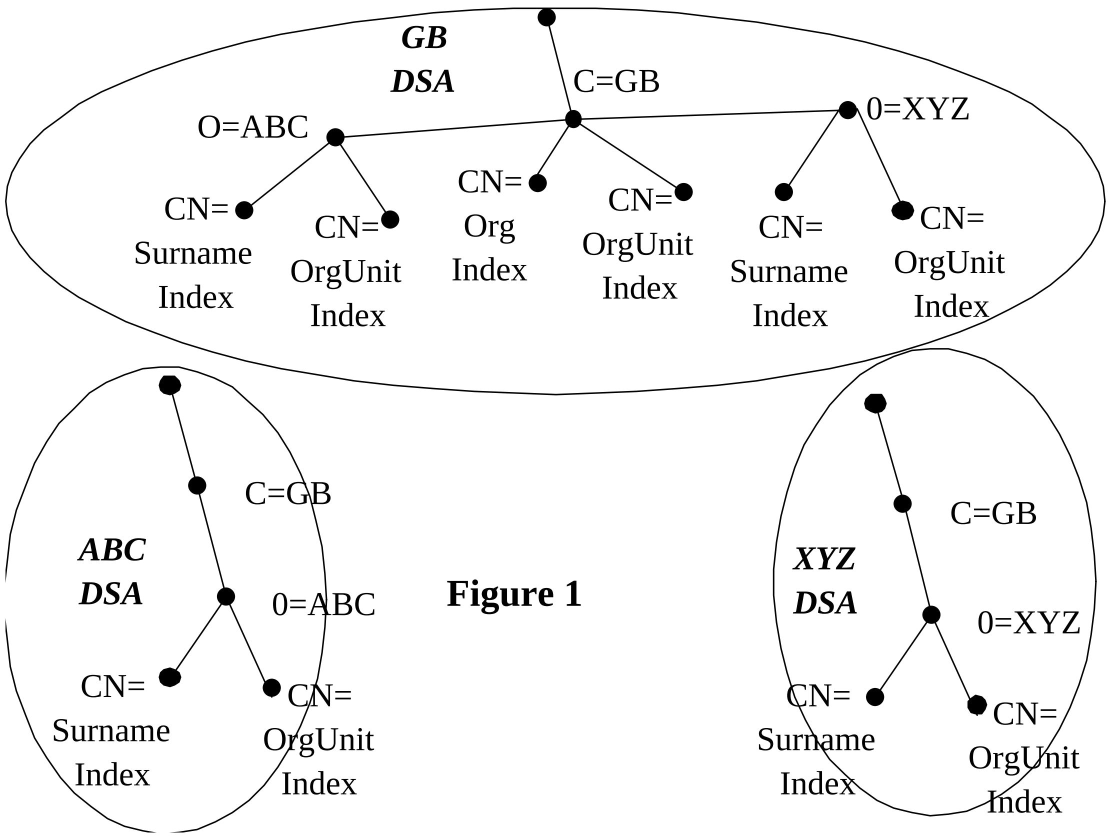

Figure 1

Alternatively, a single ‘surname and OU’ index could have been created holding the following information:

DSA Access Point: DSA Name: C=GB, O=ABC DSA; Presentation Address: string representation of the PSAP of ABC DSA
Organisational Unit Name: ITI, Electrical Engineering
Surname: Chadwick, Larmouth
CommonName: OUSurnameIndex

The GB DSA administrator decides that he would like to copy the indexes from the subordinate DSAs, and this he does, creating 4 index entries, two below the O=ABC node, and 2 below the O=XYZ node. The GB DSA is thus able to determine the organisational units and the surnames held in both of its subordinate DSAs. In addition, the GB DSA administrator decides to merge the OrgUnit indexes to create a list of all of the organisational units within GB. In this case the DSA Access Point in the OrgUnit index will now point to the GB DSA. The GB DSA administrator also decides to create his own index of organisations held in the GB DSA, and this is held in the OrgIndex entry. The root DSA (if it exists) is able to copy any of the 6 indexes held by the GB DSA. Probably the most important index for the root DSA to copy is the OrgIndex from each of the country DSAs. In this way the root DSA will learn about all the organisations in the world.

Finally, note that it is always possible for indexes to be copied *down* the DIT as well as up, so that for example, in Figure 1, the ABC DSA could copy the OrgIndex entry from the GB DSA, thus determining which organisations are present in the GB DIT.

Useful Indexes

Probably the most useful index to have is a world wide index of all organizations. Each country DSA would be responsible for creating its own index of organizations, and this information could be copied by the root DSA, for it to pass to all the other DSAs.

The next most useful indices would be at the country level, and these would be indexes of organisations and surnames (pointing to the particular org DSAs).

Finally, local DSAs might find it useful to have copies of some of the indexes stored elsewhere in the DIT, and to have a superior reference to the country DSA (so that searches of the root can be more quickly carried out).

Creating a Centroid DSA

Step 1.
Server DSAs create their own index entries. This can be achieved by the administrator performing a whole subtree Search on each naming context held by the DSA. The parameters of the Search command are:

service control: chaining prohibited
base object: the context prefix of the naming context
subset: whole subtree
filter: present = Attribute type to be indexed
search aliases false
selection: Attribute value to be indexed

The index entry is then created below the context prefix entry, and the Access Point of the server DSA is added to the entry.

Step 2A.
Server DSAs do not know about the Centroid DSA. The Centroid DSA administrator should Search the global DIT for entries of object class index, starting from the root of the DIT. This allows the Centroid DSA to create its index entries. The specifics of the Search command are:

service control chaining prohibited
base object is null
subset is whole subtree
filter is object class = index
search aliases is false
selection is all

This will return a set of index entries, plus a set of continuation references to other DSAs, which can be followed to pick up additional index entries.

Step 2B.
Server DSAs know about the Centroid DSA. The server DSA administrators tell the Centroid DSA administrator about the index entries that they hold. The Centroid DSA then either reads the index entries (and periodically polls them for updates) or enters into shadowing agreements to replicate the index entries.

Using a Centroid DSA

By now, the reader should realise that a Centroid DSA holds a set of index entries, and that each index entry holds a list of attribute values held by the referenced DSA. Index entries can be as simple or as complex as the creator requires. For example, an index entry could simply hold a list of the organisation names held by a DSA, or could hold the favourite drinks, telephone numbers and surnames of organisational people held by the DSA. When searching a Centroid DSA, the Search request depends upon whether the DUA understands index entries or not.

If the DUA is aware of index entries, then the following parameters are used for the Search operation:

> base object: set to root (for worldwide indexes), C=AA (for country level indexes), or C=AA, O=ZZ (for organisational level indexes)
> subset: full subtree
> filter :set to match (a subset of) the attributes held in the index entries, as driven by the user, AND object class equals index
> selection: set to dsaAccessPoint (i.e. dsaName and presentationAddress)

The DUA will then use the returned information to initiate requests to the referenced DSAs.

For example, one could search a Centroid DSA for organisation names, using any standard filter (substrings, equality or approximate matching), and the result would be the names of the server DSAs that hold organisation names matching the filter. Alternatively one could search a centroid DSA for a surname = xx and a favourite drink of Australian Shiraz or a telephone number of 1234. Any DSA holding that combination of attribute values would be returned (but note that that combination of attribute values are not necessarily held in the same organisational person entry! For this reason it might make more sense to build index entries that index one attribute type only.

If the DUA does not understand index entries, then a normal Search request will be issued as now, and any returned continuation references will be acted upon. Index entries should not be returned (see Migration Plan).

An example of using index entries

A user who does not know that the author works at the University of Salford, could search a Centroid DSA, looking for surname=Chadwick. The user would determine if the scope should be the whole world (base object = null) or just a particular country (base object is C=nn), and the DUA would formulate the Search of indexes. The DUA can direct the query to its home DSA, and this DSA will route the request in the normal way to its superior reference (that is assuming, of course, that the user's home DSA does not hold the indexes). The Centroid DSA will search through all of its index entries, and will get a match on the C=GB, O=University of Salford, CN=SurnamesIndex (and on any other index that holds a Chadwick value), and will either:

> if it recognises index entries (via their object class), return a partial result containing a Continuation Reference to each DSA holding a matched index entry;
>
> if it does not recognise index entries, return the DN of the index entry and the DSA access point. Note that the index attributes are NOT returned, as these could be huge - the DUA did not request it.

The DUA can then either use the Continuation References as is, or create them from the index entries, and thereby send queries directly to the DSAs that it knows hold Chadwick entries. Note that this method fulfils the same design criteria as the Whois++ design i.e. that false positives might be returned, but false negatives will not be returned. In other words, the Centroid DSA will not omit the names of DSAs that might have the desired information, but may return the names of DSAs that don't have the required information.

Migration towards Index Entries

There are 4 alternative cases to consider as index entries are introduced in the NameFLOW-Paradise directory:

> neither the DUA nor the DSA understand that index entries are special
>
> the DUA understands that index entries are special, but the DSA does not
>
> the DSA understands that index entries are special, but the DUA does not
>
> both the DUA and DSA understand that index entries are special.

This implies that we can have Centroid DSAs that collect together index entries, but they do not understand that index entries are special. If a DSA does understand index entries it will return continuation references in their place. If the DUA understands index entries it will formulate special Search queries to deal with them, and will create continuation references from the responses. If both DUA and DSA understand index entries, the DUA will formulate special Search requests, but will only receive continuation references. If neither DSA nor DUA understand index entries then the DUA could receive 'peculiar' results from an index entry to a Search operation.

Points for further study

i) How much technical work is needed for a DUA to support the searching of index entries and the processing of the results, and how much technical work is needed for a Centroid DSA to support index entries?

i) How much administrative work is needed to set up the index entries and Centroid DSAs?

Acknowledgements

The author is grateful for the many useful comments that he has received, which have allowed this paper to evolve. Particular thanks are due to DANTE for funding this work, and to Vincent Berkhout, Colin Robbins, Paul Barker, Mark Wahl, Tim Howes and Chris Weider who provided valuable technical comments.

References

[1] Barker, P. "X.500 Index DSAs". Paper presented at NameFLOW-Paradise meeting at Salford University, March 1995.

[2] ISO / IEC 9594-9 | ITU-T Rec X.525 "The Directory: Replication", 1994

[3] Hardcastle-Kille, S. "Replication and Distributed Operations Extensions to Provide an Internet Directory using X.500". RFC1276, 1991

Changes made since the first version

It was suggested that the indexing information should be created and stored initially by the indexed DSA, rather than the centroid DSA. The benefits of this are that organizations can tightly control the indexes that they create and export, and can update them when appropriate. They can also use the indexes themselves if their DSA has been enhanced.

It was also suggested that indexes can be transferred to the centroid DSA by either shadowing or scanning. Both of these mechanisms are supported.

Nexor suggested that a DSA should be enhanced to understand index entries, whereas the original paper suggested that only DUAs should be enhanced. Now both are catered for.
Several colleagues suggested that a DSA might want to have more than one index for the information that it holds, and so this has been added. Also the original method of naming index entries was seen to be too restrictive. This has been altered.

It was also suggested that indexes of indexes should be supported, thereby allowing a hierarchy of Centroid DSAs to be created. This is supported in two ways, by merging indexes into super-indexes, and also by collecting indexes into subtrees of indexes within a Centroid DSA.

Finally, it was suggested that the indexing mechanism should be made more general to cater for the case where an X.500 directory holds indexes to information in a non-X.500 directory. Whilst this change has not been made to the current specification, I would envisage that it can be accomplished along the lines of the extended V3 X.509 certificates (currently being profiled by the PKIX group). The altName component allows for alternative names such as Email addresses and Web addresses to be used as well as distinguished names. I envisage that the DSA name and context prefix attributes of an index entry, which are currently defined as distinguished names, could become altNames.

ASN.1 Definitions of Index Object Classes and Attribute Types

The *index* object class is the superclass of all index entries.

```
index      OBJECT-CLASS        ::= {
           SUBCLASS OF         {top}
           MUST CONTAIN        {dsaName |
                               presentationAddress }
           MAY CONTAIN         {contextPrefix }
           ID                  c1 }
```

The *dsa name* attribute is used to hold the distinguished name of the server DSA being indexed by this index object.

```
dsaName    ATTRIBUTE           ::= {
           SUBTYPE OF          distinguishedName
           ID                          a1 }
```

Index Subclasses - an example, the organisation index
93 definition using multiple inheritance–

```
orgIndex   OBJECT-CLASS        ::= {
           SUBCLASS OF         {index |
                               organization }
           ID                  c2}
```

88 definition without multiple inheritance–

```
orgIndex   OBJECT-CLASS        ::= {
           SUBCLASS OF         {index }
           MUST CONTAIN        organizationName}
           MAY CONTAIN         OrganizationalAttributeSet
           ID                  c2}
```

Similarly, index subclasses can be created for organisational persons, application entities etc. as required.