alters an EDB file, it writes the current (UTC) time in string format as the version string.

Generally following the header are a sequence of blank line separated entries. The concept of a null EDB file that contains just a header is allowed but discouraged. However it is sometimes useful as a temporary measure when creating a database.

An entry consists of a set of attributes, each attribute begins on a new line of the file. Section 3.2 discusses attributes in more detail and Chapter 10 describes the syntaxes used by all the attributes QUIPU recognises.

The first line of an entry is the relative distinguished name (RDN) of the entry. The subsequent line contains the non distinguished attributes.

14.1.2 Object Class Attribute

Of all the attributes an entry may have, the `objectClass` attribute is one of the most important from the configuration point of view. It defines the set of mandatory and optional attributes that must and may be present in the entry. For example, the object class `person` insists that there is a `surname` attribute, and there may optionally be a `telephomeNumber` attribute. QUIPU knows about all the standard object classes and attributes, those defined by COSINE/Internet [BK91] and those defined by QUIPU itself (see Appendix D). The full set of object classes and attributes a DSA knows about is defined by the "oidtables" which are explained in Section 14.11.

An entry can belong to more that one object class. For example, an entry representing an organisation might have the following object class attribute:

```
objectClass = organization & quipuNonLeafObject
```

And a person within that organisation might use the following object class definition:

```
objectClass = organizationalPerson & quipuObject & thornPerson
```

Every entry in a QUIPU DSA should belong to either the `quipuObject` or `quipuNonLeafObject` object classes[1], as this allows attributes the DSA needs to be added to an entry.

[1] An entry may belong to `ExternalNonLeafObject` instead IF it is actually represented in the DIT by a non-QUIPU DSA — See Section 14.6.

Further, object classes posses the notion of `class inheritance`. This means that an object class can be defined as a subclass of a previously defined object class with additional refinements. As a subclass, the newly defined object inherits all the semantics of its superclass, in addition to having additional semantics.

For example, the directory defines an object class called `person`. This object class defines the attributes which a person in the real world might have. It may be useful to refine this somewhat to talk about persons who have network access. So, we need a new object class, e.g., `netPerson`. This can be defined in a straight-forward fashion:

- The object class `netPerson` is a subclass of the object class `person` which *may* contain an additional attribute, `netMailbox`.
- The syntax of an `netMailbox` is a simple string of printable characters which is not case sensitive when performing comparisons.

It is a QUIPU requirement that every entry that is not a leaf of the DIT should belong to the object class `quipuNonLeafObject`.

This class has one mandatory attribute:

`masterDSA:` identifies the directory entity which is responsible for maintaining the master EDB for the children of this entry. The value is a Distinguished Name.

There is typically a single master for a particular entry in the tree. Hence, this value is usually single-valued. When an entry is to be modified, the directory must contact the entity responsible for the MASTER EDB for that entry in order to perform the modification.

This class has two optional attributes:

`slaveDSA:` identifies any directory entities which have authoritative copies of the EDB for the children of this entry, and are prepared to resolve operations on that EDB file for a remote DSA. The value is one or more Distinguished Names.

`treeStructure:` identifies the object classes which may exist immediately below this entry. The value is one or more object classes. See Section 14.7.2 for full details of how to set this attribute.

Since a fundamental assumption of the directory is that reads (queries) occur much more frequently than writes (updates), it is common to have several entities containing authoritative copies of an EDB. By keeping copies locally, queries can be answered with less latency.

14.1.3 Database Structure

All the local information held by a QUIPU directory is held in an in-core database, this is loaded from disk when the DSA starts.

The data on disk is held in a UNIX tree of EDB[2] files that map the DIT. At every level in the DIT for which the DSA holds data, there is a single file called *EDB*. The top level of the DIT is stored in the UNIX directory defined by the `treedir` variable in the *quiputailor* file. For example, the setting

```
treedir     /usr/etc/quipu-db/
```

would define that the top level of the DIT would be found in this UNIX directory. If you hold a copy of the root EDB file, it will be found here.

If an entry defined in an *EDB* file has children stored locally, then the *EDB* file for the children will be found in a sub-directory whose name is the string encoded relative distinguished name of the entry. For example, underneath the root, there are typically countries such as "c=GB". The data for this will be held in the file *c=GB/EDB*, or to give the fullpath name */usr/etc/quipu-db/c=GB/EDB*.

This mapping continues all the way down the DIT hierarchy, so for example, if an *EDB* file has an entry whose RDN is "ou= Computer Science", then if the entry for "ou= Computer Science" has sibling entries and these are stored locally, they can be found in the file *ou=Computer Science/EDB* relative to the directory that contains the EDB file with the "ou=Computer Science" entry.

NOTE that the case sensitivity of the sub-directory naming is one of the few areas within QUIPU where string matching is case sensitive. The case of the attribute type is taken from the definition of the attribute in the oid tables, whereas the case of the attribute value is the same as that found in the EDB file. Spacing is also important. There should be no spaces either

[2]Throughout this section the term EDB is used in the generic sense, and includes *EDB.gdbm* files if the TURBO_DISK compile option is used

side of the "=" sign, and only one space between each word contained in the name.

When an entry is modified, a new *EDB* file is rewritten to disk. The old *EDB* is renamed *EDB.bak* to provide a limited backup.

14.1.4 Long Distinguished Names

There is a problem with the above method for naming UNIX sub-directories with some versions of UNIX — particularly System V. In these systems directory names are limited in length. Some versions of UNIX will not allow space characters in filenames (in any case they are hard to manage because of all the quoting required).

To allow for this, `any` distinguished name can be given a "mapping name", which will be used as the UNIX sub-directory name. For example the entry for "o=University College London", may be mapped onto the name "UCL" and thus stored in the file *UCL/EDB*. The mapping names are specified in a file called *EDB.map*, found in the same directory as the EDB file holding the entry to be mapped. So if the file *c=GB/EDB* contains an entry for "o=X-Tel Services Ltd", then the file *c=GB/EDB.map* may be used to map this onto "X-Tel".

The syntax of the *EDB.map* file is:

```
<Distinguished Name> "#" <Mapped name>
```

so for the example used above, the file will contain:

```
o=X-Tel Services Ltd#X-Tel
```

Only RDNs you want to map need to be in this file. If a name is not found in the mapping file, then the long directory name will be used.

When a DSA needs to create a sub-directory (e.g., after an add operation) it will use the relative distinguished name for each sub-directory, unless the name is longer than the maximum number of allowed characters (usually 15). In this case, the DSA will generate a shorter mapped name and write this to the *EDB.map* file. A generated mapped name is based on the UNIX *mktemp* procedure call and produces names such as "XTelServia01950".

14.2 Setting Up an Initial DSA

These instructions are assuming that you are trying to set up a DSA with the following characteristics:

- It is the first DSA in an Organisation.
- It is not the first DSA in the Country.
- It holds a copy of the root EDB.

This is found in the example:

others/quipu/quipu-db/organisation

There are two other examples which might also be used as illustrations for national DSAs and organisational DSAs not holding a copy of the root EDB. These are in :

others/quipu/quipu-db/national

and

others/quipu/quipu-db/non-root.

Note that if you are going to be running a DSA in the United States, then you should skip this section and refer to the document [Ros90b], which is provided in the ISODE documentation set (look in the source tree area for the directory *doc/whitepages/admin/*). This document describes turn-key installation mechanisms for DSAs in the United States.

To start a DSA with one of these example databases go into the relevant database directory and type:

```
$(SBINDIR)ros.quipu -t ./quiputailor
```

The `-t` flag tells QUIPU to use the *tailor* file *./quiputailor* rather than the default file (*quiputailor* in the ISODE `ETCDIR` directory).

This will cause the DSA to print some logging information onto the screen, followed by the message

```
DSA c=GB@cn=toucan has started on localHost=17003
```

```
cn=Toucan
presentationAddress= localHost=17003
edbinfo= #cn=giant tortoise#
description= Demonstration DSA
description= Bird with large colourful bill.
objectClass= quipuDSA & quipuObject
manager= c=GB@o=X-Tel Services Ltd@cn=Camayoc
acl= others # compare # attributes # userPassword
userPassword = toucan
quipuVersion= quipu 6.8 #3 (trellis) of Mon Feb 4 09:26:47 GMT 1991
supportedApplicationContext= QuipuDSP & X500DSP & X500DAP
```

Figure 14.2: Example DSA Entry

The default setup assumes you have TCP/IP access and starts a DSA on the IP address of the local machine (127.0.0.1). If you do not have TCP/IP, you will need to change the address the DSA will attempt to listen on. The next section delves into the world of addresses!

The presentation address of the example DSA is found in the file

others/quipu/quipu-db/organisation/c=GB/EDB

The first entry in this file is the entry for a DSA called "c=GB@cn=Toucan", which is the name of the DSA we are trying to start (as defined by the "mydsaname" entry in *quiputailor*). The entry is shown in Figure 14.2. The attribute `presentationAddress` defines the address that the DSA is going to listen to the network on. If you need to listen on a different address, you should change the value of the attribute to the address you want to listen on.

Having started your DSA you should be able to connect to it by invoking DISH. If you are using the default DSA address, and are using the default *dsaptailor* file, then invoking DISH without arguments is sufficient. If you are not using the default address or *dsaptailor* file, then you will need to edit *dsaptailor* in the ISODE `ETCDIR` directory. You should add an entry

```
dsa_address  toucan  <presentation address>
```

Note that `<presentation address>` should have *exactly* the same value as the `presentationAddress` attribute in the DSAs entry in the EDB file. Now to contact the DSA use

```
dish -c toucan
```

Once connected to the DSA, try issuing the command:

```
list "@c=GB@o=University College London@ou=Computer Science"
```

You should get a list of four names back:

```
1. commonName=Colin Robbins
2. userid=quipu%commonName=Colin Robbins
3. commonName=Steve Kille
4. commonName=Michael Roe
```

If this happens you have a working DSA. (The entry numbered 2 is an example of an RDN with multiple values !)

14.2.1 Presentation Addresses

This section is a brief introduction into presentation addresses, and may be all you need to know. For the brave, more details are given in *Volume Two* of this manual and [Kil89a].

The example address used in the previous section for the demonstration DSA was "`localHost=17003`". This is an address using the TCP/IP transport stack, and is the TCP loop back address. You should *not* use this address in any DSA involved in the pilot DIT for reasons explained at the end of the section.

If you want to use the TCP/IP transport stack (using RFC1006) there are two possibilites for an address. If you are connected to the Capital-I Intenet network, you can use addresses of the form

```
Internet=128.16.5.31+17003
```

"128.16.5.31" is the IP address of your machine, which can usually be found in the */etc/hosts* file on your system. You can use the DNS name of your host instead, but the ISODE will replace this with the IP address when possible to aid portability of addresses. "17003" is the TCP port number the DSA will listen on.

If you want to use TCP/IP on a local Ethernet[3], that is not connected to the Capital-I Internet, you will need to define a local network community.

[3]Ethernet is a trademark of the Xerox Corporation.

This process is described in full in *Volume Two* of this manual and should result in addresses of the form

```
LOCAL-ETHER=128.16.5.31+17003
```

Do *not* use `Internet=` addresses **unless** you are connected to the Internet.

Now onto the X.25(80) community. If you have IPSS access, you can use addresses of the form

```
Int-X25(80)=23421920030045
```

where 23421920030045 is the DTE of you host. If you plan to have more than just a DSA (e.g., *iaed* or *tsapd*) listening on X.25 on the same machine, we advise the use of a two-digit subaddress (45 in the example). The DTE should have the DNIC included; full details of tailoring DNICs are given in *isotailor*(5n). The X.25 parameters PID and CUDF can be specifed if needed, see [Kil89a] for details.

If you do not have connectivity to the IPSS network, as in the TCP/IP case, you will need to define a local community. The ISODE has knowledge of two such X.25 communites built in. In Europe, the IXI network should be addressed using an address of the form

```
IXI=20433450210398
```

In Great Britain, the Janet network should be addressed using an address of the form

```
Janet=00002100102998
```

Your DSAs address should contain a component for every network you have access to. Multiple components can be linked using the "`|`" symbol; for example:

```
Internet=128.16.5.31+17003|IXI=20433450210398
```

Using this addressing mechanism it is possible to use transport, session and presentation selectors; however, in the context of a QUIPU DSA they are unnecessary. Hence we advise against their use; again details can be found in [Kil89a].

Correctly defining your DSA's address is important. A paper [BR89] describes the problem in detail. Briefly, X.500 was defined assuming a single

global network. Unfortunately the world is not yet like that: there are at least two major communities. QUIPU DSAs know about this, and make a careful choice between use of referrals and chaining in an attempt to make sure all operations suceed. If your DSA is incorrectly addressed, other DSAs may make the wrong assumption about your DSA, and so your DSA will not be able to contact certain parts of the DIT, and they will not be able to contact you. For a similar reason you should not use the "localHost" macro for addreses, as other DSAs will think they can connect to the DSA in question.

14.2.2 Choosing a Name for Your DSA

Every QUIPU DSA *must* have an entry in the DIT, hence your DSA will need a unique **distinguished** name. This entry is used by other DSAs to identify your DSA, and so is needed if other DSAs are going to be able to see your DSA. The examples are tailored to start a DSA called "toucan", and will be sufficient to get an example DSA started, but it is not unique, so will not be of much use when you want to start adding your own data, and want to connect into the global DIT. *Note* that the DN has to be unique, but not the RDN component, thus both "cn=Wombat" and "c=GB @ cn=Wombat" are allowed!

There are other two aspects to consider in choosing a name for your DSA.

Firstly, it is a QUIPU convention that DSAs should be named after endangered South American wildlife, and that the entry for the DSA should contain a description of the animal or plant in question. Some example animal names are shown in Table 14.1; names of wild plants can also be used! A more comprehensive list of animal names can be found in the IUCN's "Red Book" [IUC82]. This is not a just a game, there is a serious point. The authors believe it is wrong to name DSA "UCL.DSA" or similar, as this binds the DSA to the organisation too tightly, especially at the higher levels of the DIT.

Secondly, the entry for your DSA must be visible to other DSAs that do not know how to contact your DSA. So, the *master* copy of your DSA's own entry must be held at least one level higher up in the DIT than the part of the DIT it holds as `MASTER` data. For example the DSA which holds "c=GB @ o=University College London @ ou=Computer Science", must be held at the "c=GB @ o=University College London" level (e.g., "c=GB @

o=University College London @ cn=wombat") or above. Such naming also helps prevent loops as described more fully in [Kil89b].

Typically, you will not hold the master EDB file containing your DSA locally. However, you do still have control of the entry as if you held the master copy. This is explained in more detail in Section 14.4.

In practice DSAs should be named fairly high up the tree. Each country should have at least two DSAs named at the root level. Each Organisation should have at least two DSAs named at the national level.

You should use DISH to find out if the name you want is already taken. For example, if you are creating a DSA for an organisation in Great Britain, you might use:

```
% dish -c "Giant Tortoise"
Welcome to Dish (DIrectory SHell)
Dish -> search @c=GB -filter objectClass=dsa -nosize
```

This will print out the list of names already in use.

14.2.3 Setting up Your DSA

Having chosen a name, you will need to tell your DSA its name, and make sure it can find an EDB file for its own entry.

A DSA finds it own name from the "mydsaname" variable defined in the file *quiputailor* (see Section 14.3 for details). An example "quiputailor" entry would be:

```
mydsaname: "c=GB@cn=a dsa name"
```

Having read this name from "quiputailor", a DSA will try to find the corresponding entry, in this case by looking in the *c=GB/EDB* file.

If you hold the EDB that the entry should be in, simply make sure the entry is in that EDB and then it will be found. If you do not hold, and do not want to hold the EDB in which your DSA is named (e.g., your DSA is called `c=GB@cn=toucan` but you do not hold the EDB `c=GB`), then you should (normally) supply a cached copy of the EDB which contains only the entry for the DSA, and not all the other entries the full EDB would have. If you do not supply it, your DSA will have to rely on other DSAs before it can start.