10.3.3 Photo

COSINE/Internet Attributes
photo
personalSignature

Photos are a special case of "ASN.1" (see Section 10.5), but the output format is different. Your attention is drawn to Section 14.7.4 which discusses storing large attributes in a separate file.

With this version of QUIPU, the attribute should be encoded as a bit-string, in a two-dimensional G3FacsimilePage encoding (as per recommendation T.4). However, in the future this will migrate to an X.400 bodypart, containing a G3FacsimilePage, with the advantage this can then be one- or two-dimensional. The current photo decoders will recognise the new format.

10.3.4 Audio

COSINE/Internet Attributes
audio

The data is in the form of a u-law encoded sound file. If you have a Sun Microsystems Sun 4 workstation, the demonstration "play" utility can be used to play the attributes. (A modified version of "play" that ignores the file header is better — such as that made available by J. Michael Bauer of the University of Calgary. QUIPU-support can supply details if required.)

This is an interim demonstration attribute; the format is temporary and may be replaced when a suitable standard audio encoding mechanism is found.

10.3.5 Data Quality

COSINE/Internet Attributes
singleLevelQuality
subtreeMinimumQuality
subtreeMaximumQuality

```
<dataQuality>   ::=  <compKeyword> "#" <attrQuality>
                     "#" <listQuality>
                     ["#" <description> ]
```

```
<attrQuality>   ::= <levelKeyword> "+" <compKeyword>
<listQuality>   ::= <list> "$" <list> <listQuality>
<list>          ::= <attribute> "+" <attrQuality>
<compKeyword>   ::= "NONE" | "SAMPLE" |
                    "SELECTED" | "SUBSTANTIAL" |
                    "FULL"
<levelKeyword>  ::= "UNKNOWN" | "EXTERNAL" |
                    "SYSTEM-MAINTAINED" | "USER-SUPPLIED"
```

These attributes are used to define the quality of data at a single level of the DIT, or even the whole subtree.

10.3.6 DSA Quality

COSINE/Internet Attributes
dSAQuality

```
<dsaQuality> ::=  <DSAkeyword> ["#" <description>]
<DSAkeyword> ::=  "DEFUNCT" | "EXPERIMENTAL" |
                  "BEST-EFFORT" | "PILOT-SERVICE" |
                  "FULL-SERVICE"
```

Used to define the quality of a DSA.

10.4 MHS Attribute Syntaxes

Attribute	Syntax
mhsORaddresses	ORAddress
mhsDLMembers	ORName
mhsDLSubmitPermissions	DLSubmitPermissions

The Syntaxes shown above are not currently recognised by QUIPU, and are thus handled as raw ASN.1 (see Section 10.5). However, the PP MHS System has QUIPU compatible syntax handlers that can be easily added to the DISH program, and provides a tool for handling entries of the "distribution list" object class. PP is available under conditions similar to the ISODE; for details you should contact "PP-Support@cs.ucl.ac.uk".

10.5 ASN.1

As the preceding sections in this chapter have mentioned, not all syntaxes have a string representation defined by QUIPU, so are represented by the raw ASN.1.

An example would be:

```
photo= {ASN}0308207b4001488001fd...
```

where `0308...` is a hexadecimal representation (encoded using the "Basic Encoding Rules") of the ASN.1 defined attribute.

Attributes stored as ASN.1, will usually be matched correctly, with the following exceptions:

- There is an IMPLICIT Set in the ASN.1. The DSA will not detect the set, and so will not know to match components in arbitrary order.
- If special matching rules apply, for example, special rules to determine equivalence of telephone numbers. Such rules would need to be represented by code in the DSA.

Chapter 11

Introduction to Security Features

11.1 Passwords

When you bind to a DSA, to get write access you need to tell the DSA who you are — you do this by supplying your distinguished name. So that the DSA can authenticate you, you also need to supply your QUIPU password. Anyone knowing your password can act as you and alter your data. Therefore your password should be well chosen and kept safe.

11.1.1 Choosing a Password

- Do not use the same password for QUIPU as you do for login. It is always bad practise (but convenient!) to use the same password on different systems. The manager of your QUIPU DSA can discover your QUIPU password. This doesn't affect the security of QUIPU (as your DSA manager can alter local data anyway), but the QUIPU manager is probably not allowed access to your login account.

- Do not choose an obvious word.

  Your password should **not** be any of the following:

  - Your name, initials, date of birth, place of work or similar personal details.

  - Any girl's name in any language.
  - The name of a film star, television personality, and so forth.
  - The name of a character from a science-fiction or fantasy novel.
  - Computer jargon e.g., "foobar".
  - Any of the above, spelt backwards.
  - Any word from your system's on-line dictionary. Ideally, your password should not be a word at all (it could be the concatenation of two unrelated words, for example).

- Change your password from time to time.

11.1.2 Taking Care of Your Password

- Configuration Files. It is possible to put your QUIPU password in a configuration file in your home directory (see the later Chapters on user interfaces). If you do this, you must make sure that the configuration file (.quipurc) is not publicly readable (by using the UNIX *chmod* command, for example).

- Access Control Lists. Your QUIPU password is itself stored in QUIPU, as the *userPassword* attribute of your entry. You should make sure that the access control list for your entry grants public *compare* but not *read* access to this attribute. (The attribute must be publicly comparable so that QUIPU can check that you have presented the right password when you start a session). The next section will explain how to set up access control lists.

- Do not give your password to other people.

  If several people need to be able to modify the same data, the access control lists can be set up so that this is possible without them sharing passwords.

- Do not write your password down!

11.2 Discretionary Access Control

"Discretionary access aontrol" is any means by which users can (at their discretion) give other users access to data which they control. In QUIPU, access control lists are used to provide discretionary access control.

11.2.1 Model

Each node in the DIT held by a QUIPU DSA has a QUIPU access control list (ACL). The ACL is divided into three parts:

1. Child ACL. This controls who may discover or change which entries are placed immediately below the node in the DIT.

2. Entry ACL. This controls who may access the entry placed at the node.

3. Attribute ACL. For every attribute of the entry, there is an attribute ACL which controls who may access that attribute. To keep the representation compact, each entry has a default attribute ACL. This is used for all the attributes of the entry that have not been explicitly given a different attribute ACL.

Each entry, child, or attribute ACL (henceforth called an object ACL) consists of a list of (access selector, access level) pairs. It associates every distinguished name with an access level, according to the following rule:

> Take every pair where the selector (left hand side) matches the name. The associated access level is the maximum of the corresponding right-hand sides. If no selectors match the name, the associated access level is *none*.

The levels of access are as follows:

`None:` No accesses to the object are allowed.

`Detect:` A DUA can detect that the protected object exists.

`Compare:` A filter (e.g., test for equality to some value) may be applied to the object.

`Read:` The contents of the object may be read.

`Add:` The contents of the object may be added to, but not removed from.

`Write:` The contents of the object may be modified in any way.

The possible Access Selectors are as follows:

`Entry:` Matches the entry itself only.

`Other:` Matches everything.

`Prefix <name>:` Matches <name> and everything below it in the DIT.

`Group <name>:` Matches <name>. (This is not what the "group" selector was originally designed for, but it is currently what it does!)

The attribute syntax used to represent this is defined in Section 10.2.1, with the ASN.1 definition shown in Figure 11.1.

11.2.2 Detect Access

QUIPU treats the access level *none* as though it were *detect*. This means that it is often possible to detect the presence of protected data, even if you have no access to it.

The reason for this is that it is very difficult for the directory to pretend that data isn't there; carefully chosen queries can catch it out. Access control mechanisms that can be by-passed are very dangerous; they give a false sense of security. Accordingly, we have decided not to implement "undetectable data".

11.2.3 Effect of ACLs on Operations

This section explains which ACLs are checked for each of the X.500 operations.

`List:` The child ACL of the target must give at least *read* access. In addition, each child will only be listed if its entry ACL gives at least *read*. Later versions of QUIPU may also require that the attribute ACL of the distinguished (naming) attribute of the child give at least *read* access.

`Search:` To search the immediate descendants of a node, that node's child ACL must give at least *read* access. In addition, each child will only be searched if its entry ACL gives at least *compare* access. The filter "present" may be applied to any attribute. Other basic filters evaluate to *maybe* unless the relevant attribute ACL gives at least compare access. If an attribute within an entry does not have public read access, normally subtree searches on that attribute will fail. This is intentional, as it would be difficult to provide a consistent picture of the DIT, when unauthenticated DSP links are involved in the search. This restriction is lifted if, and only if, the entire subtree to be searched (using master or slave EDB files) is held within one DSA, and the association to that DSA is an authenticated DAP association.

`Read:` The entry ACL of the target must give at least *read* access and the attribute ACL of each attribute read must give at least *read* access.

`Compare:` The entry ACL of the target must give at least *compare* access. The attribute ACL of the tested attribute must give at least *compare* access.

`Modify:` The entry ACL of the target must give at least *add* access if attributes or values are to be added, and at least *write* access if they are to be removed. For each attribute changed, the attribute ACL must give at least *add* access for an attribute or value to be added, and at least *write* access for an attribute or value to be removed.

`Add Entry:` To add an entry below a node, that node's child ACL must give at least *add* access.

`Remove Entry:` To remove an entry, the child ACL of its parent must give at least *write* access. No rights to the entry itself are required.

`Modify RDN:` The child ACL of the target's parent must give at least *write* access. If the operation needs to add an attribute (value), the target's entry ACL must give *add* access and the attribute ACL of the attribute must give *add* access. If the operation needs

to remove an attribute (value), the target's entry ACL must give *write* access and the attribute ACL of the attribute must give *write* access.

11.2.4 Example Use of ACLs

A node representing a user might be given the following ACL:

- childACL is not applicable, and so omitted.
- EntryACL is {other, read} + {self, write} + {group=<Manager>, write}, so that only the user or a manager can change the entry. Using the ACL syntax, this is expressed as:

```
acl= other # read # entry
acl= self # write # entry
acl= group # <Manager Name> # write # entry
```

- DefaultAttributeACL is {other, read} + {self, write}, which leads to publicly readable attributes modifyable by the user. Using the ACL syntax, this is expressed as:

```
acl= other # read # default
acl= self # write # default
```

- The attribute ACL for ACL is {other, read} + {group = <manager name>, write}, so that only the manager can change the ACL. Using the ACL syntax, this is expressed as:

```
acl= other # read # attributes # acl
acl= group # <Manager Name> # write # attributes # acl
```

- The attribute ACL for Password is {self, write} + {other, compare} so that the user can change the password, DSAs can check the password, and only the user can read it.

```
acl= self # write # attributes # userPassword
acl= other # compare # attributes # userPassword
```

A node representing an organisation or organisational unit might be given the following ACL:

- childACL is {other, read} + {group=<manager name>, write}. Everybody can search the members of the organisation, but only the manager is allowed to add or delete members. This is represented as an attribute with:

```
acl= other # read # child
acl= group # <Manager Name> # write # child
```

- EntryACL is {other, read} + {group=<manager name>, write}. This is represented as an attribute with:

```
acl= other # read # entry
acl= group # <Manager Name> # write # entry
```

- DefaultAttributeACL is {other, read} + {group=<manager name>, write}. This is represented as an attribute with:

```
acl= other # read # default
acl= group # <Manager Name> # write # default
```

Every entry in the QUIPU DIT must have an acl attribute. If you do not supply one, the the default is added. The default ACL is often printed as

```
acl=
```

with no value. The default ACL is everybody read everything, but self can write, in long form this is expressed as:

```
acl= self # write # entry
acl= self # write # child
acl= self # write # default
acl= others # read # entry
acl= others # read # child
acl= others # read # default
```

For many entries this is sufficient.

It can be seen that this scheme gives a great deal of flexibility, without the addition of any protocol elements. The encoding is designed so that the volume overhead is not excessive for sensible access policies.