9.3.3 The friendlyNames File

This file maps attribute names onto user-friendly names, for use in displaying the "Current Directory Position". The current defaults map the names onto empty strings so that the bare values are shown, e.g.:

```
The World
GB
Brunel University
```

The format of the file is

```
attribute list : friendly name
```

where an attribute list can be one or more comma-separated attribute names.

9.3.4 The filterTypes Files

POD searches are based upon a complex filter; e.g., the default filter for the type *Person* is

```
objectClass=person AND (cn~=* OR sn~=* OR title~=*)
```

Where "*" represents a value supplied at search time.

The directory *filterTypes/* contains a set of files, each with a prefix *Type_*, describing the set of such abstract types used in POD. The set of contained files may be edited or added to.

The syntax used in these files is shown in the example below:

```
<filter_type> ::= <filter_name> <filter>
<name> ::= "name:" <asciistring>
<filter> ::= <filter_item> | <assertion>
<assertion> ::= "(" <filt_type> <filter> <filter>
                <filter_list> ")"
<filter_list> ::= <filter> <filter_list> | <filter> | NULL
<filter> ::= <filter_item> | <assertion>
<filter_item> ::= "(" NUMERIC_OID <match_type> <value> ")"
<filt_type> ::= "&" | "|"
<match_type> ::= "=" | "~=" | "%="
<value> ::= "*" | STRING
<comment> ::= "#" STRING
```

Where the symbol " =" represents approximate match, "%=" represents substring match and "=" represents exact match. The "*" character for value represents a value supplied at search time.

As an example, the file representing the default type "Person" is:

```
#Composition of type Person
name:"Person"
( & (2.5.4.0="2.5.6.6") # objectClass = person
    ( | (2.5.4.3~=*) # cn (common name)
(2.5.4.4~=*) # sn (surname)
(2.5.4.12~=*))) # title
```

9.3.5 The typeDefaults File

The *typeDefaults* file defines the relationships between each of the type defined in the *filterTypes* directory. This is best described by the following example line from the provided *typeDefaults* file:

```
2.5.4.10:Person, Place, Department: Person
```

The first field defines the type of entry to which this line applies, the OID shown here is the one for *organizationName*. The second field lists the POD search types which are available when visiting an entry of the specified type. So this line specifies that the types *Person*, *Place* and *Department* are available when visiting an entry of type *organizationName*. The third field defines the default search type to be used when visiting an entry of the specified type.

Chapter 10

Attribute Syntaxes

All attributes recognised by QUIPU have an associated "string" representation which is used to store the data in the DSA. This chapter describes all the currently recognised syntaxes.

For most of the syntaxes, a BNF description is given, using the following base description:

```
<a> ::= any of the 52 upper and lower case IA5 letters
<d> ::= any IA5 digit 0-9
<k> ::= any of the 52 upper and lower case IA5 letters,
              IA5 digits, and "-" (hyphen)
<p> ::= any IA5 character in ASN.1 PrintableSting
<CRLF> ::= IA5 Newline
<letterstring> ::= <a> | <a> <letterstring>
<numericstring> ::= <d> | <d> <numericstring>
<keystring> ::= <k> | <k> <keystring>
<printablestring> ::= <p> | <p> <printablestring>
```

The BNF description of attributes can be found in Appendix B, but this does not repeat the BNF used for syntaxes given below.

10.1 Standard Syntaxes

This section describes the Attribute Syntaxes defined by X.500.

There are many attributes that are represented by a sequence of printable characters. There are various ways in which the contents of the string is

restricted, each is represented by a different syntax, which are described below.

10.1.1 PrintableString

Standard Attributes	QUIPU Attributes
serialNumber	execVector

The printable string characters are:

`A` through `Z`
`a` through `z`
`0` through `9`
`'` (apostrophe)
`(` (left parenthesis)
`)` (right parenthesis)
`+` (plus-sign)
`,` (comma)
`-` (hyphen)
`.` (period)
`/` (solidus)
`:` (colon)
`?` (question-mark)
space

The value can be any character listed above, with matching as for CaseExactString.

When matching, multiple white spaces and tabs are treated as a single space character.

Approximate matching is supported for this syntax.

The only standard attribute of this type is `serialNumber`, this is used in the `device` objectClass, and represents the serial number of the device. QUIPU defines an `execVector` attribute which is used by the *iaed* program as the vector to pass to a program when starting it.

10.1.2 CaseExactString

No attributes currently use this syntax. Matching is as for PrintableString.

The value for this syntax can be one of two types, either a printable string, or a T.61 string represented as follows:

```
<StringValue> ::= "{T.61}" <T61String> | <printablestring>
<T61String>   ::= Any character defined to be in
                  T.61 String.
```

If a T.61 string is used, where possible the character is displayed using the equivalent ASCII character. Characters in T.61 string that cannot be represented by an equivalent ASCII character are quoted, using "`\xx`" where "`xx`" is the hexadecimal value of the character.

However, if your terminal has the ability to display characters using an ISO 8859-1 font (for example some of the X Window System fonts) then a large number of the T.61 characters can be displayed (and so are not quoted in hex!). You will need to tell the DUA that you have the ability to display the ISO 8859-1 characters. This can be achieved in one of three ways:

- In the system-wide dsaptailor file, you can add

  ```
  ch_set ISO8859
  ```

  or

- in your .quipurc file you can add

  ```
  dsap: ch_set ISO8859
  ```

  However not all the DUA interfaces read this files (DISH does!) or

- set the environment variable CH_SET to "ISO8859", e.g., using "csh"

  ```
  setenv CH_SET ISO8859
  ```

To enter T.61 characters is a little more tricky. In T.61 accented characters are represented by two octets, the first indicating the accent and the second the base character to be accented. Note that some combinations of accent and character do not have an equivalent in ISO 8859-1, and hence cannot be displayed on an ISO 8859-1 device. Table 10.1 shows some of the characters that can be represented, for the accented characters only the first octect is shown, the second octect can typically be any character, "o" is used in most of the examples.

Hex code	Description	Character
c1	grave accent	ò
c2	acute accent	ó
c3	circumflex	ô
c4	tilde	õ
c5	macron	ō
c7	single dot	ȯ
ca	ring	å
cb	cedilla	ç
cc	underscore	o̲
cd	umlaut	ö
cf	caron	ǒ
a1	inverted exclamation mark	¡
a3	pound sign	£
a4	dollar sign	$
a6	hash	#
a7	paragraph sign	§
ab	left quotation mark	“
b0	ring	å
b1	plus/minus	±
b5	greek letter mu	μ
b6	pilcrow	¶
b8	division sign	÷
bb	right quotation mark	”
bc	quarter sign	$\frac{1}{4}$
bd	half sign	$\frac{1}{2}$
be	three-quarters sign	$\frac{3}{4}$
bf	inverted question mark	¿
e0	greek letter omega	Ω
e1	AE	Æ
e8	L stroke	Ł
e9	O stroke	Ø
ea	OE	Œ

Table 10.1: T.61 Character Codes

For instance, to enter the T.61 string "Galápagos Penguin," you should use

```
{T.61}Gal\c2apagos Penguin
```

The acute accent is represented by the "`\c2`" and appears over the following character — "`a`" in this case.

10.1.3 CaseIgnoreString

Standard Attributes
knowledgeInformation
commonName
surname
localityName
stateOrProvinceName
streetAddress
organizationName
organizationalUnitName
title
description
businessCategory
postalCode
postOfficeBox
physicalDeliveryOfficeName

COSINE/Internet Attributes
userid
textEncodedORaddress
info
favouriteDrink
roomNumber
userClass
host
documentIdentifier
documentTitle
documentVersion
documentLocation
durName
wkdName
protocolProfile
objectID
friendlyCountryName
buildingName

The syntax is the same as CaseExactString, except that when matching "characters that differ only in their case are considered identical".

Approximate matching is supported for this syntax.

The use of some of the attributes is now described, the more obvious ones are omitted.

`knowledgeInformation:` A description of knowledge mastered by a DSA.

`localityName:` Used to identify the geographical area or locality in which the object is physically located e.g.,

```
London
```

`stateOrProvinceName:` describes the state in which the `locality` is found; e.g.,

```
New York
```

`title:` An objects job title, e.g.,

```
Technical Manager
```

`businessCategory:` describes the business of the object, e.g.,

```
networking
```

This is used to find people sharing the same occupation.

`physicalDeliveryOfficeName:` describes the geographical location of the physical delivery office which services the postal address of this object; e.g.,

```
Troy
```

`friendlyCountryName:` A "nice" name for countries, as opposed to the two letter codes enforced by use of `CountryName`, for example

```
Great Britain
```

10.1.4 CaseIgnoreList

No standard attributes currently use this syntax.

The CaseIgnoreList syntax consists of a sequence of CaseIgnoreString values as shown by the BNF below.

```
list = <list_component> | <list_component> "$" <list>
list_component = [ "{T61}" ] <string>
```

When the list is displayed to a user, the "$" is replaced with a newline. Ordering is preserved. For example, the entry in an EDB file

```
caseIgnoreAttribute= this is a $ multi line $\
attribute definition
```

would be shown to a user as

```
caseIgnoreAttribute= this is a
                     multi line
                     attribute definition
```

QUIPU only allows equality matching for this syntax.

10.1.5 CountryString

Standard Attributes
countryName

This syntax is treated as a PrintableString, with matching rules as for CaseIgnoreString, and the restriction that the string must be one of the codes defined by ISO 3166.

10.1.6 IA5String

QUIPU Attributes
control
quipuVersion

COSINE/Internet Attributes
aRecord
mDRecord
mXRecord
nSRecord
sOARecord
cNAMERecord

This syntax is handled as PrintableString, except a wider range of characters are recognised (i.e., any character in IA5 string, characters such as NewLine are quoted using the same mechanism as described for T.61 string in Section 10.1.2), with matching rules as for CaseExactString.

10.1.7 OctetString

No attributes use this syntax directly.

Characters that cannot be printed as an ASCII are represented using the quoting mechanism described in Section 10.1.2.

10.1.8 NumericString

Standard Attributes
x121Address
internationaliSDNNumber

The value is simply a numeric string (digits 0 through 9 only)

```
<NumericValue> ::= <numericstring>
```

The two attributes are used thus:

`x121Address`: defines the X.121 Address of an object as defined by the CCITT Recommendation X.121.

`internationaliSDNNumber`: An International ISDN Number as defined by CCITT Recommendation E.164.

10.1.9 DestinationString

Standard Attributes
destinationIndicator

Behaves as a `<printablestring>`, with CaseIgnoreString matching rules.

The only attribute of the type `destinationIndicator` is used to define the addressee as required by the Public Telegram Service.

10.1.10 TelephoneNumber

Standard Attributes	COSINE/Internet Attributes
telephoneNumber	homePhone

PSI Attributes
mobileTelephoneNumber
pagerTelephoneNumber

The value should be a string describing the phone number of the object using the international notation; e.g.,

```
+1 518-283-8860
```