Pod User Manual

POD is a Directory User Agent developed for X-windows version 11 release 4, using the Athena widget set. It is intended to be an interface for the 'naive' user. If you want to use the full power of X.500 you should use the DISH interface described in the QUIPU User Guide.

1. Types of Widget

POD makes use of various simple on-screen devices called widgets. The most important of these are described in the following sections.

1.1. Buttons

A button is a rectangular screen area which when selected will activate some specific function. A button is selected by pointing at it with the cursor and then clicking with the first mouse button.

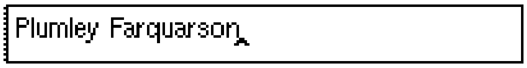
Most buttons contain a label and sometimes graphics which denote the button's function, as can be seen in the example button shown. In addition POD buttons are denoted by a change to a pointing hand cursor or by a border becoming highlighted.

A button box is simply a collection of buttons.

1.2. Dialogue Boxes

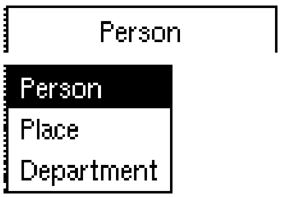
A dialogue box is a means of supplying textual information to an application, and can be thought of as a mini text editor. The dialogue boxes used in POD follow the command structure of the EMACS editor.

Dialogue boxes, like most others widgets, only become active (or have the input focus) when pointed at by the cursor.



1.3. Menus

A menu is a vertically arranged collection of buttons that appears on selection of a menu button.



2. Using POD

POD is invoked with the following command.

```
pod [-t <tailor file>] [-T <oidtable>] [-c <dsa address>]
      [-u <quoted user name>] [-p <password>]
```

The command line options have the following effect:-

-T Use an alternative oidtable.

-t Use the specified tailor file instead of *<ETCDIR>/dsaptailor*

-c Bind to the specified DSA.

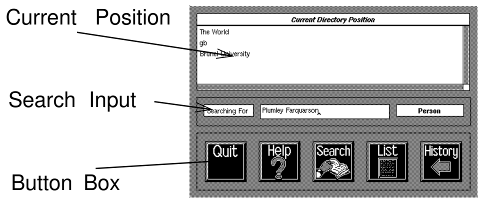
-u Bind as the specified user.

-p Bind with the given password.

POD is made up of a number of separate windows, described in the following sections.

2.1. The Main Window

POD’s main window is comprised of three sections: a current position display, a search input box and a button box. Each of these is described below.



2.1.1. Current Directory Position

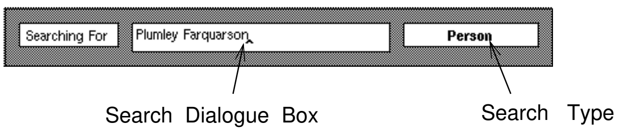
All operations are performed relative to a base directory entry. The name of this entry is displayed in the main window under the title "Current Directory Position".

To move to and read one of the entries shown select the appropriate line of text with the first mouse button, e.g "GB" in the illustration above. To list entries under an entry select with the *middle* mouse button. To move only, select with the *right-hand* mouse button.

2.1.2. Searching for Entries

The search input area contains a dialogue box and a menu button.

The dialogue box is used to enter a value describing an entry to be searched for, for example if searching for the entry for "Damanjit Mahl" a suitable value would be "D Mahl".



The menu button specifies the type of entry being searched for (the search type), in the above example this would be ‘Person’. The search type can be changed by pulling down the menu attached to the type button and selecting from the contained list of types.

Searches are activated by clicking on the ‘Search’ button or by pressing the <RETURN> key.

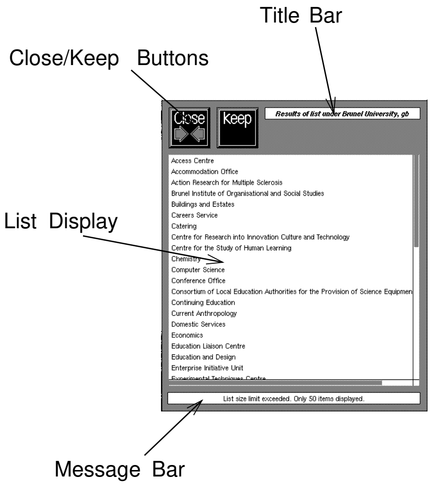
2.1.3. Main Functions

The buttons displayed in POD’s main window are:

- Quit - Quit from POD
- Help - Invoke interactive help
- Search - Search for specified entry
- List - List entries under current position
- History - Display history of visited entries

2.2. The List Window

A list window displays a list of named directory entries, which can be returned as a result of list, search or history operations. It comprises a pair of buttons, a list display and two message bars.



The upper message bar contains the information regarding the source of the displayed list, e.g. ‘Result of List under Brunel University’. The lower message bar contains errors and constraint messages.

2.2.1. List Display

To move to and read a listed entry simply click on that entry’s name. Non-leaf entries cause a move and read, whereas leaf entries only cause a read.

To list entries under one of the displayed entries select the appopriate entry using the *middle* mouse button. To simply move to a listed entry select with the *right-hand* mouse button.

2.2.2. Close and Keep Buttons

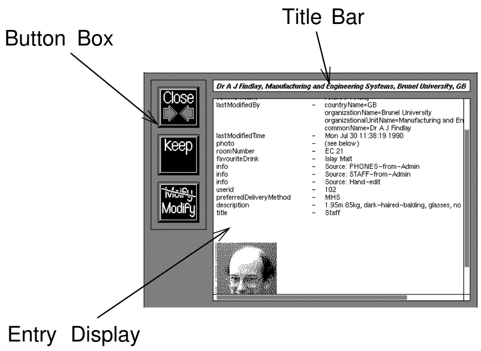
The close button can be used to close the list window. Note that this does not iconify the window but rather deletes it and all contained information.

The keep button can be used to make the information contained in the list window semi-permanent. In the default case, results from ensuing list/search operations overwrite any ‘unkept’ list windows, thereby limiting the number of windows containing temporary information to one. If the keep button is selected, displayed information becomes semi-permanent, and is only removed if the close button is selected.

The text in a keep button changes from ‘Keep’ to ‘Kept’ upon selection, in order to distinguish from list windows containing transient data.

The List window that displays the session history does not have a keep button.

2.3. The Read Window



The read window displays selected parts of a directory entry. It is comprised of a title bar, a button box and an entry display window.

The title bar displays the entry’s name. If selected this bar places the complete distinguished name of an entry into the primary X cut buffer. Ordinarily this is of little use, though can help when modifying attributes that require a distinguished name as a value.

The text window displays the body of the entry, which currently contains textual information and may contain a (single) fax image stored in the photo attribute.

The close and keep buttons perform a similar function to those described above for the list window.

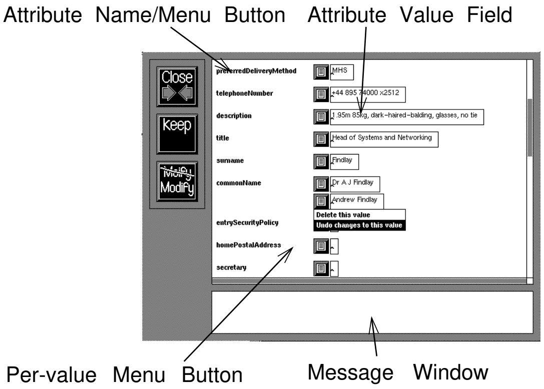
The modify button activates a window containing facilities to modify the entry being read.

2.4. The Modify Window

The modify window allows the user to modify the attributes of an entry. Each attribute value displayed in the text window is contained in it’s own dialogue box. Pointing to a dialogue box will allow editing of the contained value.

Each value has an associated menu button (denoted by the menu icon) which allows various operations to be performed on a per-value basis, e.g. delete value, undo changes etc.

In addition each attribute label, e.g. ‘commonName’, is itself a menu button and allows various operations on all values associated with that attribute, e.g. delete all values, undo all changes to this attribute, add a new value field.



Undo operations only undo all changes since the last succesful modify operation or else return all affected values to their original states.

The modify button attempts to make the modification entered into the text window. A message reporting the success (or otherwise) of a modification request is displayed in the lower right message window.

2.5. Error and Status Messages

From time to time, during or after directory operations, POD will provide error or status reports in a popup that appears in the top left corner of the screen. Errors sometimes require the user to click on the error window before normal operation can resume.

3. Configuration of POD

POD can be configured on a system-wide or per-user basis. POD installs system default files into a directory called *xd/duaconfig/*
under ISODE’s ETCDIR.

Per-user configuration files other than *.duarc*, must be held under a directory *.duaconfig/* contained in a user’s home directory.

The POD configuration files are:

<CONFIGDIR>/readTypes,
<CONFIGDIR>/duaconfig/friendlyNames,
<CONFIGDIR>/duaconfig/filterTypes/Type_*,
<CONFIGDIR>/duaconfig/typeDefaults.

3.1. The .duarc file

The *.duarc* file is analogous to the *.quipurc* file for DISH, though less extensive in the number and flexibility of options provided. It has the following format:-

```
flag: value
```

The following flags are currently recognised:-

- username

  The name of the user to bind as.

- password

  The password to use when binding. For this reason care should be taken to ensure your *.duarc* file is not publicly readable.

- service

  A list of default service control flags as described the Quipu User Guide.

- isode

  This can be followed by one of the ISODE tailor options described in Volume One of the ISODE Manual.

- history

  This can be followed by a number that specifies the number of entry names to be maintained in the history buffer.

- prefergreybook

  For those who prefer grey book mail address format. This can take values of TRUE or FALSE.

- readnonleaf

  This tells POD whether or not to read non-leaf entries. Values are TRUE or FALSE.

3.2. The readTypes file

The file *readTypes* contains a list of attributes that are to be read for entry display. The format of this file is;

```
"quotedAttributeName"  numericOid
```

an example line then being;

```
"photo"        0.9.2342.19200300.100.1.7
```

This format was chosen as it is the same format as the output from the OIDDUMP utility provided with ISODE.

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.
The format of the file is:-

```
attribute list : friendly name
```

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

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
```

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. Thus 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.