```
[-sequence <name>] [-nosequence]
```

are used to control sequences. The `-nosequence` flag is use to completely remove the concept of sequences (so no numbers are printed by list or search).

To get a different sequence (hence renumber from 1 again), you can change this to mysequence with a

```
-sequence mysequence
```

flag.

The special keyword `reset` is used to reset the current sequence, thus causing future operations to be renumbered from 1. This is different to setting a new sequence, as the old sequence values are removed.

Occasionally you will want to search an entry defined in one sequence, putting the results in another. For this purpose the special token `result` is recognised, thus

```
search 4 -sequence xxx -sequence result yyy
```

will search entry 4 as defined by sequence "xxx", but place the results in sequence "yyy".

4.3 Service Controls

All the commands described in Sections 4.1.2 through 4.1.11 have additional flags to control the type of service provided. The advice to a novice user is let these flags take their default values.

The flags recognised are listed below:

`-(no)preferchain`: Advise the DSA (not) to chain the operation if required — The DSA **IS** allowed to ignore the advice!

`-(no)chaining`: (Prohibit) Allow the use of chaining.

`-(dont)usecopy`: (Prohibit) Allow the use of a cached or slave copy of the data. Note that this refers to data cached in the DSA; there are separate flags (already described) for data cached in the DUA.

`-(dont)dereferencealias`: (Do not) dereference aliases if found in the path of a query.

`-low`: Flag the query as low priority.

`-medium`: Flag the query as medium (default) priority.

`-high`: Flag the query as high priority.

`-timelimit` *n*: Set the time limit to *n* seconds.

`-notimelimit`: Do not specify a time limit.

`-sizelimit` *n*: Set the size limit to *n* entries.

`-nosizelimit`: Do not specify a size limit.

`-(no)localscope`: (Do not) limit operation to local scope.

`-(no)refer`: (Do not) automatically follow referrals issued by the DSA.

`-strong`: Ask the DSA to use strong authentication when sending the results.

4.4 Tailoring

There are two levels of tailoring that control the operation of the DISH DUA. First of all there is the system wide tailor file *dsaptailor* (found in the ISODE `ETCDIR` directory, usually */usr/etc/*), and then users private tailor file *.quipurc*.

The file *dsaptailor* is described in Section 13.2.

4.4.1 .quipurc

The file *.quipurc* is read by DISH on start up, and has the following format:

```
flag:  value
```

The following flags are currently recognised:

`username`: The name of the user to be used when binding.

`password`: The password to use when binding. For this reason care should be taken to ensure you .quipurc file is not publicly readable.

`cache_time`: How long to keep the DISH process alive after unbinding (specified in minutes).

`connect_time`: How long to keep a connection open, if there is no activity (specified in minutes).

`service`: A list of default service control flags (as defined in Section 4.3).

`sequence`: Add the supplied DN parameter to the default sequence.

`dsap`: This can be followed by one of the *dsaptailor* options described in Section 13.2.

`isode`: This can be followed by one of the ISODE tailor options described in Volume 1 of the ISODE manual.

`notype`: The argument is expected to be an attribute type. This attribute will not be displayed by showentry by default.

`type`: The only valid argument for this entry is "unknown," and says that unknown attributes should be shown; by default, they are not shown.

`certificate`: Used for strong authentication.

`secret_key`: Used for strong authentication.

<`dish command`>: A list of default flags to be used when "command" is used. For example:

```
showname: -compact
```

says that `showname` should always be invoked with the `-compact` flag set.

<`nickname`>: A nickname entry, for example

```
cjr: "c=GB@o=X-Tel Services Ltd@cn=Colin Robbins"
```

This can then be used in distinguished name arguments, for example

```
showentry cjr
```

will show the entry for "Colin Robbins".

4.5 Remote Management of the DSA

```
dsacontrol [-[un]lock <entry>] [-dump <directory>]
           [-tailor <string>] [-abort]
           [-restart] [-info] [-refresh <entry>]
           [-resync <entry>] [-slave [<entry>]]
```

If the DISH program is bound to the DSA as the directory manager (See Section 4.1.13 on binding), then it may be used (primitively) to manage the DSA. This feature is discussed in [Kil89b] and Section 14.3.1 of this manual.

`(un)lock` used to (un)lock subtrees of the DIT held locally. This (allows) prevents users to modify the DIT.

`dump` causes a copy of the local DIT to be dumped into the specified directory. Note that when the DSA dumps its in-core database, it does so relative to the possibly remote UNIX directory from which the DSA was started, not the directory that DISH was invoked in. So it is best to specify path names in full to ensure the database is dumped where you expect it to be.

`tailor` is used to pass tailor command to a running DSA. The format of "string" is the same as a line in the *quiputailor* file. For example:

```
dsacontrol -tailor "dsaplog level=all"
```

will turn the DSA logging on full.

`refresh` tells the DSA to reread subtree "entry" from disk. As with all dsacontrol commands, entry must be the full DN of the target entry, the DISH concept of current position is not used. Also, if relating to a disk operation, the CASE of the DN must be correct as well.

`resync` tells the DSA to re-write the subtree entry to disk.

`slave` tell the DSA to update its SLAVE EDB files. The keyword root is used to identify the root EDB file. The keyword *shadow* is used to tell the DSA to update any entries it has shadow copies of. See Section 14.10 for details.

`abort` tells the DSA to stop. `restart` tells the DSA to stop, then restart.
`info` asks the DSA to return some statistics about its database.

The error messages returned from a DSA are somewhat obscure when using dsacontrol. There are four things DISH may report:

1. `DONE`: the operation has been successfully completed;
2. `Scheduled`: the operation has been scheduled, and will be completed as soon as possible;
3. `Access rights insufficient`: you are not bound as the DSA manager;
4. `Unwilling to perform`: the DSA was unable to perform the task, this generally means the argument to dsacontrol was wrong.

4.6 Caching in the DUA

All results returned from read and list operations are cached in memory in the DUAs. The cache is then used if possible to answer queries. The cache is kept for *cache_time* minutes as specified in the *tailor* files.

Disk caching has not been implemented yet, but it is intended to extend the in-core caching, so that the cached data can be written to disk. When the DUAs start up, the user will have the option to read this cache. Similar support for private data may be provided.

4.7 Running DISH from the Shell

Each of the DISH commands mentioned in this chapter can be applied directly to the shell, without having to first invoke DISH. This has the advantage that the result can be piped through programs like *more* thus giving the users flexibility in how they view the data.

Chapter 12 describes how to install this extra feature of DISH.

Each time a new shell is invoked and a DISH command entered, a new DISH will be invoked. This is sometimes not required. To prevent this from happening an environment variable DISHPROC can be set, and should be the TCP address to use for communication, for example:

```
127.0.0.1 12345
```

where 127.0.0.1 is the IP address, and 12345 is the TCP port. The two example scripts show how the TCP port may be generated if required under `sh`. For `csh` you might use

```
if (${?DISHPROC} == 0) then
        setenv DISHPROC "127.0.0.1 `expr $$ + 10000`"
endif
```

4.7.1 DISHinit

A program called *dishinit* can be use to bind to the directory as the manager, read certain information and create a default `.quipurc` file for a specified user. This is a useful utility for new databases. If the `dsaptailor` variable `dishinit` is set to `on`, dishinit will be invoked by DISH whenever is cannot find a .quipurc file.

`dishinit` needs three parameters to be set in order to bind as the manager, and search the local subtree. These parameters are defined at the top of the dishinit script and are:

`manager:` The DN of the manager of the local DSA.

`password:` The manager's password (the script should be protected with UNIX file protection).

`position:` The DN of the local sub-set of the DIT. This is where the entries for users will be looked for

4.7.2 Example Scripts

Two example scripts have been supplied as part of ISODE, and can be found in the *others/quipu/uips/dish* directory.

`dsaping:` Contacts all the DSAs at a given level in the DIT, and gathers some statistics.

`dsalist:` Produces a list of all the known DSAs in the DIT.

4.7.3 Files

When the multiple command version of DISH is compiled on a machine that does not support sockets, name pipes are used to communicate with the a background process.

The file /tmp/dish<pid> is used by DISH to write results to the calling processes, where the pid is the process id of the calling process.

The file /tmp/dish–<tty>–<uid> is used by calling process to send commands to DISH, where *tty* is the users terminal number, and *uid* their user id.

Chapter 5

SID

SID (Steve's interface to DISH) is a directory user agent, designed to be incredibly easy to use. It is implemented as a simple set of scripts to the DISH DUA.

There are a lot of flags to DISH. This can make it awkward to use, and in general shows too much directory-specific information. FRED has a centralised view of the world, and does not give the advantages of a shell based interface.

SID is a simple approach of giving appropriate aliases to access DISH commands. It is targetted for searching for people and organisations. It may be useful "as is" to others, and also give an indication as to how DISH may be used.

5.1 Quickstart

WARNING Before you start to try SID, you must have the variable DISHPROC set in your environment. The mechanism suggested in Section 5.6 is a good way to achieve this. To experiment, csh users can type the incantation:

```
% setenv DISHPROC "127.0.0.1 `expr $$ + 1000`"
```

To find users within your Organisation is very easy. The default setup is optimised for your particular Organisational Unit. Simply use the command

psearch with a single argument to identify the user looked for. This will identify all users with this key as a substring. For example:

```
% psearch steve
countryName=GB
organisation=University College London
department=Computer Science

1   cn=Steve Kille
telephoneNumber    - +44-1-380-7294 (work)
telephoneNumber    - 01-350-2888 (home)
2   cn=Stephen Titcombe
telephoneNumber    - +44-1-387-7050 x 3674/5
telephoneNumber    - Term = 01-388-4741
3   cn=Stephen Sackin
4   cn=Steve Wilbur
telephoneNumber    - +44-1-380-7287
5   cn=Stephen Usher
telephoneNumber    - +44-1-387-7050 x 3674/5
6   cn=Kevin Steptoe
7   cn=Tomasz Stepniak
8   cn=Stefan Penter
telephoneNumber    - +44-1-387-7050 x 3674/5
9   cn=Stephania Loizidou
10  cn=Steve Hodges
telephoneNumber    - +44-1-387-7050 x 3674/5
11  cn=Steve Davey
telephoneNumber    - +44-1-387-7050 x 7280 or 7289
12  cn=Steve Chan
telephoneNumber    - +44-1-387-7050 x 3674/5
13  cn=Steven Britton
telephoneNumber    - +44-1-387-7050 x 3715
14  cn=Stephen Beckles
telephoneNumber    - +44-1-387-7050 x 3674/5
15  cn=Steven Bacall
telephoneNumber    - +44-1-387-7050 x 3674/5
telephoneNumber    - dd 01-387-6978 (Campbell House)
```

Each match will have an associated number. To show a given entry, just

use the command **showentry**, with the number identifying the entry as the single argument. For example, to show entry 1 (Steve Kille):

```
% showentry 1
c=GB@o=University College London@ou=Computer Science@cn=Steve Kille
Address            - 35 Elspeth Road
                     London
                     SW11 1DW
userClass - csresstaff
photo - (See X window, pid 598)
roomNumber         - G24
favouriteDrink     - Pinta - Brakspears
info               - Directory worker
rfc822Mailbox      - S.Kille@cs.ucl.ac.uk
textEncodedORaddress - /Pn=Stephen.Kille/Ou=cs/O=ucl/ \
                     Prmd=uk.ac/admd=gold 400/c=gb/
userid             - steve
telephoneNumber    - +44-1-380-7294 (work)
telephoneNumber    - 01-350-2888 (home)
description        - New Description
surname            - Kille
name               - Steve E. Kille
name               - Stephen Kille
name               - Steve Kille
```

If you are running the X Window System a picture will be displayed if it is present.

If you wish to search all of your Organisation, simply give a second argument to **psearch**. For example, to search UCL give "ucl" as shown:

```
% psearch baker ucl
Searching for People matching "baker" under:
countryName=GB
organisation=University College London

4   ou=French Lang and Lit@cn=F Baker
telephoneNumber  - 3077
5   ou=Economics@cn=V Bashkar
telephoneNumber  - 2286
6   ou=Biochemistry@cn=J Basra
telephoneNumber  - 2185
7   ou=Bartlett School@cn=H Bowker
```