technical memorandum Daresbury Laboratory

DL/SCI/TM79E

THE COMPLETE GUIDE TO THE DARESBURY INFO-SERVER

by

P. GRIFFITHS, SERC Daresbury Laboratory

Issue 1.1 April 25, 1991

June, 1991

Science and Engineering Research Council
DARESBURY LABORATORY
Daresbury, Warrington WA4 4AD

ISSN 0144-5677

The Complete Guide to the Daresbury Info-Server

P.Griffiths

Daresbury Laboratory

Issue 1.1 April 25, 1991

Contents

1 Introduction

The Daresbury Info-Server provides users with an **electronic mail** file retrieval facility for data kept in a public domain **archive** at Daresbury. That is to say, a user may send electronic mail to the Info-Server with specific instructions embedded in the body of the message; the server then accepts this mail and performs operations on behalf of the user directed by the commands in that message. Commands usually take the form of requests for files, indexes on directories, various options or administrative trivia.

This document covers two different aspects of the server. Chapter 2 describes the operation of the server as users see it when communicating via electronic mail; chapter 3 details administrative information relevant to people who manage directories exported to the outside world via the server.

To contact the server send mail to:

Info-Server@uk.ac.daresbury

To contact the server administrator send mail to:

Info-Server-Manager@uk.ac.daresbury

New terms and keywords are introduced in the document using a **bold** font. Any word being introduced in this way will have an entry made in the glossary.

The intended uses of this server are:

- The distribution of public domain software, including Unix, Atari and Archimedes specific software.
- The distribution of **Internet RFCs**.
- The distribution of software from **Collaborative Computing Projects** run at Daresbury (includes distribution of remote host databases[1]).

We are always interested in acquiring new software for the archive. If you have anything you think could be of interest then please send a summary via E-Mail to **Info-Server-Manager@uk.ac.daresbury**. Similarly comments and bug reports regarding the server should be sent to the same address.

This document was produced using ArborText Publisher Version 3.0.4A running on a Sun Microsystems SparcStation 1+. Printing was done on a 300dpi DataProducts LZR2665.

[1] See appendix B

2 User Guide

As mentioned in the document introduction the Info-Server responds to commands sent to it via electronic mail. This section describes the set of commands the Info-Server recognizes. Please note that in all mail messages sent to the server the **Subject:** line is unused.

1 Layout of the archive

The archive is based on a Unix style directory hierarchy, the directory separator being the forward slash (/), the root being referred to as just a / on its own. All directory names start with a capital letter, file names are in lower case (i.e. `/Unix/Gnu/binutils.tar.Z` is the file `binutils.tar.Z` in the sub-directory `Gnu` of the main root sub-directory `Unix`). Common conventions are used when naming files, for instance a **compressed** file will have a trailing `.Z` extension, a **tar** file will have a trailing `.tar` extension. Combinations of file types (e.g. a compressed tar archive) will be represented with both extensions, the order of unpacking being read from right to left (i.e. a compressed tar archive will have the extension `.tar.Z` meaning you have to uncompress it before you can run tar on it).

2 Coding

Most source files sent out by the server need to be encoded for transmission across electronic mail. Coding the files avoids problems with gateways translating certain characters and allows binary files (most files in the archive are binary) to be sent out over a medium that only supports ASCII[2]. Choosing a suitable coding method was no easy task as there are a myriad of programmes available. We therefore came up with a check list of what we wanted from a coding method and then looked for something that fitted the bill:

- ☐ The ability to code up binary files in ASCII.
- ☐ An included table of characters used for the coding so that translations occurring could be spotted and corrected.
- ☐ The source had to be public domain so it could be made available via the server.

As a result of this we decided to use a variant on the uuencode/decode program commonly available on Unix machines: Dumas[3] (or uue/uud — we'll refer to it as Dumas). Dumas is a reverse engineered version of uuencode/decode, with the added advantage of including a translation table at the start of each coded file. Dumas will quite happily decode uuencoded files making it the ideal choice (note here that the reverse is not true, uuencode/decode will not decode Dumas files!). Dumas is available from the Info-Server in two forms:

- As the source file `/Bootstrap/dumas.shar`. The server is aware of the significance of the file and will send out the source without performing any coding on it!

[2] American Standard Code for Information Interchange

[3] Named after its author.

- As part of a collection of files living in the `/Bootstrap` directory of the server, i.e. requesting `/Bootstrap/unix` gives you a source file which assumes you are on a vanilla system allowing you to get started from scratch. A BBC Basic version of uud is present as `/Bootstrap/archimedes`

3 Command format

Most commands in the body of a mail message take the form:

```
Request: request-type
Topic: topic within the current request-type
Topic: topic within the current request-type
Topic: topic within the current request-type
Request: request-type
Topic: topic within the current request-type
Topic: topic within the current request-type
Request: end
```

We shall refer to this as a **command sequence**.

Commands are handled by request drivers, which are allocated dependent on the request-type parameter in a `Request:` line. In a single mail message previously allocated drivers are superceded each time a valid `Request:` line is encountered. In all mail messages blank lines are ignored. Table 1 gives a full listing of all the request types recognized, with associated driver (upper and lower case may be freely intermixed in request lines).

Request Line	*Driver*
Request: Admin	admin
Request: Catalogue	index
Request: Digests	source
Request: Help	help
Request: Index	index
Request: Mail-lists	source
Request: Rindex	index
Request: Rfc	source
Request: Sources	source

Table 1 Request types with associated drivers

To put the table into some sort of perspective we can say that all file requests are handled by the source driver, directory listings are handled by the index driver, general information pertinent to the server is supplied by the help driver and administrative duties are handled by the admin driver. Once a driver has been allocated, each `Topic:` line encountered is sent to the driver for processing, the whole being terminated upon encountering a `Request: end` line (end of message is treated as equivalent to a `Request: end` line). Multiple `Request:` lines may be embedded in a single message. For all drivers (i.e. request types) you may specify a topic line of the form `Topic: help` to get request specific help information.

Apart from the known request types the server also recognizes a set of *stand alone* commands (i.e. single line commands requiring no previous `Request` line). Stand alone commands may appear anywhere in the body of a mail message, even interspersed between `Requests` and `Topics`. Table 2 lists the set of stand alone commands available:

Table 2 Stand alone commands

Ping
Quit
Coding: *coding-method*
Line-Limit: *size*
Return-path: *new-path*

For every command sequence sent to the server, a reply file is generated showing how the server interpreted the commands. Upon termination of the command sequence the reply file is sent to the user with a standard header reporting the latest news and information, the following example depicts a typical response generated by the server:

```
Date: Mon, 1 Apr 91 16:27 UT
From: Daresbury-Info-server <Info-Server@UK.AC.Daresbury>
Reply-To: Daresbury-Info-server <Info-Server@UK.AC.Daresbury>
Sender: Daresbury-Info-server <Info-Server@UK.AC.Daresbury>
To: P.G.Griffiths@uk.ac.daresbury
Subject: Re: your requests "Example"
X-Software: Daresbury-Info-server V 6.3.2
Comments: Bug reports to <Info-Server-Manager@uk.ac.daresbury>
```

```
==========================================================================
=          Greetings from the Info-Server at Daresbury Laboratory        =
=                          Running on a SUN-4/370                        =
=                                                                        =
=    THIS SERVER HAS MOVED TO A NEW MACHINE - FILE LOCATIONS, DIRECTORY  =
=    NAMES HAVE CHANGED. The Unix directory is now "/Unix".              =
=                                                                        =
=    NOTE! The UUencoding programs used by the server are UUe/d NOT the  =
= uuencode/uudecode found on standard Unix systems (which proved to be   =
= somewhat unreliable across some gateways).                             =
=                                                                        =
=    The source for UUe/d can be found in '/Bootstrap/dumas.shar' along  =
= with the source for Compress/UnCompress (BSD Public Domain) and many other =
= utilities (dumas will be sent out in shar file format).                =
=                                                                        =
= Info - Bugs - etc: Info-Server-Manager@UK.AC.DARESBURY                 =
=                                                                        =
= JANET:        info-server@uk.ac.daresbury                              =
= Internet:     info-server%uk.ac.daresbury@cunyvm.cuny.ed               =
= EARN/BITNET: info-server%uk.ac.daresbury@UKACRL                        =
= uucp:         info-server%uk.ac.daresbury@ukc.uucp                     =
==========================================================================

Checking mail header...
Original address      : "P.G.Griffiths <P.G.Griffiths@uk.ac.daresbury>"
Return address        : "P.G.Griffiths@uk.ac.daresbury"
Line limit            : 3000
Coding                : UUE

Processing started    : Mon Apr  1 16:25:48 1991

*request: sources
Valid request.

*topic: index
Sending index.

*topic Index /Unix
Sending index.

*topic rindex /Unix/Sun
Sending recursive index.

*topic /Unix/Gnu/binutils.tar.Z
Sending /Unix/Gnu/binutils.tar.Z - total of 5 parts mailed

End of processing     : Mon Apr  1 16:27:23 1991
Total processing time : 95 seconds
```

The following sections first deal with the stand alone commands, followed by an examination of the semantics of each available driver.

4 Stand alone commands

Ping

Syntax: `ping`

Ping causes the server to respond with a dump of the current environment it is running in. You can use this command as a way of getting a simple response from the server, say when you are trying to debug your electronic mail route using the `return-path` command. Upon receipt of a ping request the server responds with the following type of information:

```
Pong:  Daresbury Info-Server <V/6.3.2>
Pong:  Started Fri Mar 22 15:26:17 1991
Pong:  Host name is dlsb
Pong:  Domain name is dl_sun_server
Pong:  Debug at level 0
```

Quit

Syntax: `quit`

Quit aborts any further processing the server does on your mail message. Useful for example when your mail program automatically tags a signature on to the end of messages you send; putting a quit after the end of a command body stops the server trying to interpret your signature as instructions (and sending you help files for the commands it doesn't understand), e.g.

```
Request: Sources
Topic: Index
Request: End
Quit


Yours Paul,

            JANET: P.G.Griffiths@uk.ac.daresbury
```

Coding

Syntax: `Coding:` *`coding-required`*

The coding command allows you to change the type of coding method used by the server when mailing files out to you. For each message processed, the server initially sets the coding method to be uue, however, upon encountering a valid coding line the coding method is changed to the new method specified *for the duration of that message or until another valid coding line is encountered.* At present there are three methods available:

- `uue` (the default).
- `btoa`.
- `off` (no coding whatsoever).

Line-Limit

Syntax: `Line-Limit:` *`new-line-limit`*

The line-limit command tells the server the size of messages you are prepared to accept, the default size being 1200. The prevailing line limit has the effect of causing the server to split up an outgoing message into chunks of size line limit, numbering them accordingly. Messages are split this way to avoid problems when travelling through mailers/gateways that restrict the maximum size of messages they are prepared to handle.

However, if the route that your messages take has no size restrictions you may want to increase the message size. There is a minimum line limit of 250 and a maximum of 6000 use the line limit command to vary the message size between these boundaries. For example, the following request causes the server to send `/Unix/Gnu/binutils.tar.Z` split up in sections of 4000 lines each:

```
Line-limit: 4000
Request: Sources
Topic: /Unix/Gnu/binutils.tar.Z
Request: End
```

Return-path

Syntax: `return-path:` *`new-electronic-mail-address`*

Address translation Before any processing is done by the server an electronic mail return path for the user is calculated. To calculate this path the server first extracts a valid **network domain** address from the `From:` field in the message header, e.g.:

Original From: field	*Translated address*
Paul Griffiths <P.G.Griffiths@uk.ac.daresbury>	P.G.Griffiths@uk.ac.daresbury
Bart Simpson "B.Simpson@edu.ucb.ucbvax"	B.Simpson@edu.ucb.ucbvax
D.Pringle <dp@uucp.nl.ccp>	dp@uucp.nl.ccp

Once a valid network domain address has been extracted the message header is re-scanned to search for any mail **gateways** a message had to pass through to get to the server — this information is gleaned from `Via:` fields. Gateways found are then incorporated into the domain address, producing the final electronic mail address to use when replying to the sender of the message. For example, if we had ascertained the network domain address to be **B.Simpson@edu.ucb.ucbvax** and found that the message came via **uk.ac.nsfnet-relay** the two would be merged to produce the return address **B.Simpson%edu.ucb.ucbvax@uk.ac.nsfnet-relay**.

The return-path command changes the electronic mail address the server uses when sending mail back to you. You may need to use the return-path command when you are having problems getting any response back from the server and suspect that it just isn't getting your electronic mail address right! Beware however, that you must give an address which is valid from the JANET network — if you're not on JANET then you will need to include the appropriate relays in the new address; for example, if you are located on the Internet and your address is **B.Simpson@edu.ucb.ucbvax** you will need to add the gateway uk.ac.nsfnet-relay, giving a resultant address of **B.Simpson%edu.ucb.ucbvax@uk.ac.nsfnet-relay**. Another possible use of the return path command could be to forward mail from the server to a different machine than the one you originally sent the message from — useful for example when you are requesting a large file and just don't have the disk space to hold it locally! The following example depicts an index request, followed by a file request to a different host using the return-path command:

```
Request: Sources
Topic: Index
Return-Path: pg@uk.ac.daresbury.cxa
Topic: /Unix/Gnu/binutils.tar.Z
Request: End
```

5 Unpacking

As hinted in section 2 there is a sub-directory of the main archive called `/Bootstrap` where a collection of files exists that are intended to provide basic **bootstrap** programs to get you going. When any file is requested from the `/Bootstrap` directory the server will not perform any coding on the requested file, rather it will be sent out as it appears on disk (ASCII). When you receive a bootstrap file instructions are included at the head explaining what it contains and how to extract the programs contained therein.

When creating bootstrap files we make as few assumptions as possible about the available software on the target system with the aim that upon receipt of a bootstrap file you should be able to extract a working set of utilities that will allow you to receive and unpack files requested from the server.

Once you have a working copy of the necessary utilities there is still the question of how they are used to build a source file received from the server. Let us assume the following request:

```
Request: Sources
Topic: /Unix/Gnu/binutils.tar.Z
Request: End
```

Upon receipt of this request the server will send out `/Unix/Gnu/binutils.tar.Z` coded using Dumas and fragmented according to the current line limit (see section 4). It is first necessary for you to defragment the separate files into a single uuencoded file. This part is very much dependent on the mail system you are using and the editors available to you; however there is the utility **recon** (in most bootstrap files and as source in `/Bootstrap/recon.c`[4]) that attempts to automate some of the procedure.

Recon takes a number of fragmented files and concatenates them into a single source file. Let us assume that our request for `/Unix/Gnu/binutils.tar.Z` caused the server to send us the file in 5 parts over electronic mail. If we were then to save these five parts in individual files named `binutils.`*`seq`*, where *`seq`* was a number representing the fragment number, we now have the following files:

```
binutils.1
binutils.2
binutils.3
binutils.4
binutils.5
```

Note that the `binutils` part of the name is user definable — it could just as viably be called `noddy.`*`seq`*, the only difference being that the resulting file will be called `noddy.uue`.

[4] I shall attempt to provide a compiled version of recon in bootstrap files for machines where access to a C compiler may be difficult, i.e. in `/Bootstrap/ibm-pc`

We now enter the command `recon binutils`. Recon now concatenates parts 1–5 into the file `binutils.uue` ready for decoding. OK, this seems useful, but isn't it the same as say running the Unix command `cat binutils.[1-5] > binutils.uue`! Well there is a difference, namely that recon searches each fragment file for a special command sequence that marks the start and end of the real data — meaning that it is not necessary for you to strip out mail header information, trailing newlines and other junk. Recon looks for the following command sequences:

```
<*>---START[seq]---
<*>---END[seq]---
```

These command sequences only place one restriction on the user, the sequence number contained in the start and end lines *must* equal the sequence number of the file it is saved in. For example the file `binutils.1` must use the sequence number 1, i.e.

```
Date: Mon, 1 Apr 91 16:25 UT
From: Daresbury-Info-Server <Info-Server@UK.AC.Daresbury>
Reply-To: Daresbury-Info-Server <Info-Server@UK.AC.Daresbury>
Sender: Daresbury-Info-Server <Info-Server@UK.AC.Daresbury>
To: P.G.Griffiths@uk.ac.daresbury
Subject: Sources request for /Unix/Gnu/binutils.tar.Z (UUE format) Part 1/5
X-Software: Daresbury-Info-server V 6.3.2
Comments: Bug reports to <Info-Server-Manager@uk.ac.daresbury>

<*>---START[1]---
UUencoded file...
<*>---END[1]---
```

Recon thus partially automates what was at best a fairly tedious task.

Once you have de-fragmented the source file the next step is to decode it using uud. As mentioned before, Dumas is available in source form as `/Bootstrap/dumas.shar.Z`, or alternatively in one of the available bootstrap files, e.g. a compiled version is in `/Bootstrap/ibm-pc`. Once decoded we will be left with the file `binutils.tar.Z` ready for uncompressing and extraction using tar — Table 3 should help in deciphering file extensions and deciding which package to use on archive source:

Table 3 Common archive utilities

Extension	*Archive utility*	*Source location (if available)*
.tar	tar	`/Unix/Gnu/tar-*.tar.Z`
.Z	compress	`/Unix/Packing/compress.shar`
.arc	pkunpak (IBM-PC)	Not available
.zip	pkunzip (IBM-PC)	Not available

Table 3 (Continued) Common archive utilities

.arc	arc	`/Unix/Packing/arc.tar.Z`
.zip	zoo	`/Unix/Packing/zoo2.tar.Z`
.btoa	btoa	`/Unix/Packing/btoa-*.tar.Z` `/Archimedes/AtoB_BtoA.arc`
.arc	!SparcPlug	`/Archimedes/SparkPlug`
.ps	PostScript	Most friendly neighbourhood laser printers.

The table by no means represents a complete list of all the archive utilities available. Suggestions for inclusions would be most welcome — send details to **Info-Server-Manager@uk.ac.daresbury**.

6 The source driver

The primary function of the source driver is the packing and mailing of files you request. A file is requested by specifying its *full* path on a topic line, e.g. `Topic: /Unix/Gnu/binutils.tar.Z`. This will cause the source driver to ascertain the file's existence, pack it up using uue (see section 2 on coding methods), and then mail it out to the requesting user — the file being sent out in parts according to the current line-limit.

Apart from supplying files the source driver is also capable of providing you with directory listings of the archive. Listings are requested using the **index** keyword on a topic line (`Topic: index`). Index requests take an optional argument specifying a sub-directory to apply the index to, e.g.

```
Request: Sources
Topic: Index
Topic: Index /Unix
Topic: Index /Unix/Sun
Request: End
```

The server also provides an enhanced form of the index command, **rindex**. Rindex instigates a recursive index on the directory specified, i.e. each sub-directory encountered will be processed for an index along with the original. The only restriction on rindex is that it is illegal to attempt an rindex on a root directory (`Topic: Rindex /`) as the overhead of traversing the whole archive is too great.

As can be seen from Table 1 the source driver is responsible for several different request types. There are subtle differences between each request type so we'll examine each one in detail.

Request: Sources

Sources is the primary caller of the source driver. Sources makes the whole of the archive available to the user. Use the command sequence:

```
Request: Sources
Topic: Index
Topic: Rindex /Unix
Request: End
```

to explore what's available to you. Files can then be requested using the pathnames gleaned from the index request, i.e.

```
Request: Sources
Topic: /Unix/Gnu/binutils.tar.Z
Request: End
```

Request: Digests

The digests request makes available to the user an archive of electronic mail digests kept as a subdirectory of the main archive (`/Digests`). Electronic mail digests are available via the sources request type, however the digests request avoids having to specify the `/Digests` directory leader. Currently available digests are:

Digest Name	*Summary*
SunSpots	Comments/Views/Suggestions for users of Sun Microsystems workstations.
InfoUnix	General information on Unix machines
UnixWizards	As Info-Unix, but claims to be for the more experienced user.
GraphUK	UK Graphics users magazine.
InfoMac	Apple macintosh users magazine.
InfoAtari	Atari users magazine.
TexMag TexHax UKTex	Three seperate magazines for people interested in the typesetting package TeX and related LATeX.

Digests are saved in the sub-directory of the main archive using the format:

Component	*File Type*	*Example*
Digest-Name	Directory	`/SunSpots`
Digest-Volume	Directory	`/SunSpots/V8`
Digest-Issue	File	`/SunSpots/V8/Issue124`

A typical request could be:

```
Request: Digests
Topic: Index /GraphUK
Topic: /InfoUnix/v10/Issue91
Topic: /SunSpots/v9/Issue1
Request: End
```

Again use the index keyword on a topic line to explore what's available to you.

Request: Rfc

The Internet is a large, active research project in the US linking many sites together through a combination of Wide Area and Local Area Networks. Reports of work, proposals for protocols, and protocol standards all appear in a series of technical reports called *Request For Comments*, or *RFCs*, all of which are numbered. As with the digests request type, rfc works in a subdirectory of the main archive (`/Rfc`). The `/Rfc` directory holds a subset of the most commonly requested *RFCs*. There is however an extra bonus with the rfc request type — if the requested rfc is not available the server will automatically forward your request to the main RFC server in the States known as the CSNET info-server. The requested RFC will then be sent to you by CSNET. The following example depicts a request for the two rfc's rfc1000 and rfc1234, rfc1000 is available but rfc1234 isn't:

```
Request: Rfc
Topic: rfc1000
Topic: rfc1234
Request: End
```

The server would respond to this request thus:

```
*Request: Rfc
Valid request

*Topic: rfc1000
Sent rfc1000,total of 6 parts.

*Topic: rfc1234
This RFC is not in our archives - I have sent the following
request to the US RFC server "CSNET" on your behalf:
>Request: RFC
>Topic: rfc1234
>Request: End

*Request: End
Valid request
```

All rfc's requested from CSNET are cc'd to the server-manager for inclusion in the RFC archive.

Request: Mail-lists

As part of a related project to the Info-Server, Daresbury runs a Bulletin Board Server (BBS). The BBS accepts an incoming mail message from a user and then redistributes it to a group of users. During this process the BBS appends a copy of the message to a save file. These save files are available from the Info-Server using the mail-list request type. The root directory of mail-lists contains sub-directories, each representing individual lists. Within each sub-directory save files are located, each save file representing a month's activity on the list. Save files use the naming convention `MonthYear`, both being two digit strings, e.g. February 1991's save file is `0291`. The following example shows a couple of index requests plus a save file request for the Twin-Peaks activity for March 1991:

```
Request: Mail-lists
Topic: Index
Topic: Index /Twin-Peaks
Topic: /Twin-Peaks/0391
Request: End
```

7 The index driver

The index driver provides you with information about the current contents of the archive. When called, the index command interrogates a directory and returns you a formatted display of the contents. Comments associated with each file are also included in the display — note that some files don't have associated comments, some for obvious reasons (i.e. it's pointless to give comments to files such as `/Digests/SunSpots/V8/issue12` because the pathname makes it fairly obvious what it is) and some because we haven't got round to writing them yet.

Request: index | catalogue

The index request works in the root directory of the archive; you may specify a particular sub-directory that you are interested in on the `topic:` line (a blank topic line is taken to mean the root directory of the archive). The catalogue request type is synonymous with the index request; it is provided for compatibility with other servers. The following example depicts a typical index request:

```
Request: Index
Topic:
Topic: /Unix
Request: End
```

Request: rindex

The rindex request is identical to the index request except that where index only works in the one directory, rindex will recursively descend any directories found in the directory it is applied to. Because of the large overheads involved in this rindex can only be applied to sub-directories of the main archive — applying it to the root directory of the archive will not fail, but sub-directories will not be decended.

8 The help driver

The help driver distributes help files for all request types. Using help is simplicity itself. The general format is:

```
Request: Help
Topic: request-type
Request: End
```

Where *request-type* is any one of the permissible request line types, e.g.

```
Request: Help
Topic: Help
Topic: Admin
Topic: Sources
Topic: Digests
Request: End
```

9 The admin driver

The admin driver is the anomaly amongst the drivers. Whereas the other drivers have well defined set of commands the admin driver does not. Rather, because its purpose is to enhance and ease the use of the server, there is a diverse set of commands the driver recognizes. To this end you should be aware that the facilities offered by the driver are likely to be extended as and when the need arises, as opposed to the other drivers where the commands available are fairly static. Changes to the driver will be advertised on the server reply file banner, the server mailing list and a more permanent record in revised editions of this guide.

To make best use of the admin driver you should be aware that each user of the info-server has a **user-record**, created and maintained on your behalf by the server. Records are **indexed** using your electronic mail address as the **key** field — if you therefore use the server from two different hosts you will have two separate records (and so on). The primary function of the admin driver is the modification of some of the fields in your user record. Let us first look at the fields in a user record:

Field	*Explanation*	*Initial Value*
Email-address	The electronic mail address identifying the user record.	User's Address
Return-address†	The electronic mail address of the user.	User's Address
Date-first-used-server	The date when the user first communicated with the server.	Creation Date Of Record
Date-last-file-request	The date when the user last requested a file.	NULL
Last-file-requested	The pathname of the last file requested by the user.	NULL
Number-messages-sent	The number of messages the server has sent to the user.	1
Number-files-requested	The number of files the user has requested from the server.	0
On-mailing-list†	A boolean indicating whether or not the user wants to receive mail from the server administrator, i.e. when new files are added to the archive.	FALSE
Coding†	Coding to use when sending out files.	uue
Line-limit†	Line limit to use when sending out files.	1200

The fields marked by a † are modifiable using the admin driver, thereby allowing you to customize the server to suit your individual requirements (the other fields are used for debugging

purposes only — there is no *big brother* watching you!). Note that all commands (except account) make *permanent* changes to the way in which the server responds to you. There is however no limit to the amount of changes you can make so modifications may be undone by reissuing the appropriate command. The driver recognizes the following commands:

Topic: account

This request provides you with a copy of your user record, for example:

```
*Request: Admin
Valid request type.

*Topic: Account
Account as of Wed Apr  3 12:50:34 1991

Email address               : P.G.Griffiths@uk.ac.daresbury
Return address              : P.G.Griffiths@uk.ac.daresbury
Date first used server      : Fri Feb 15 18:34:51 1991
Date of last file request   : Tue Apr  2 16:55:54 1991
Name of last file requested : /Unix/Sun/mtools2.tar.Z
Number of messages sent     : 22
Number of files requested   : 4
On server mailing list      : Yes
Coding on files             : UUE
Line limit on files         : 2000

*Request: End
Valid request type.
```

Topic: mail on | off

This request instructs the server whether or not you wish to be included on the server mailing list. The server manager mails people on the list with information on the server and updates to the archive.

Topic: return-path *new-path*

Inform the server of the return path to use when sending messages back to you.

Topic: coding *coding-option*

Set the coding option for outgoing source files — uue, etc see section 3 for details.

Topic: line-limit *size*

Set the line limit for outgoing files in lines.

3 Administrators Guide

This section is relevant only to people who manage collections of files accessible to the Info-Server. For information on making a directory available to the world via the Info-server contact **Info-Server-Manager@uk.ac.daresbury**.

1 File permissions

Files exported by the Info-Server must have user, group and other read access. Optionally user write access may be granted to allow the owner of the files to update and delete them. For example, if the directory `/Foos` contains the files `.index`, `foobat`, `foobar` and `bartap` then the output of an `ls -al` command on the `/Foos` directory should look like:

```
-rw-r--r--  1 root          175 Mar 12 17:19 .index
-rw-r--r--  1 root        49502 Mar 12 17:19 foobat
-rw-r--r--  1 root         1470 Mar 12 17:19 foobar
-rw-r--r--  1 root         4864 Mar 12 17:19 bartap
```

Here the files are writable by the owner (root) and have user, group and other read access.

There is one special case of the permissions that causes the server to treat a file differently — if that file has its sticky bit set. The presence of the **sticky bit** on a file causes that file to be ignored, or effectively hidden, by the server; that is to say that the file will not appear in any index requests and a user requesting the file will cause the server to respond with a "file does not exist" message. This technique provides a simple information hiding facility for administrators.

Setting of the appropriate file permissions may be done using the Unix command:

`chmod a=r u+w` *`filename`*

To set the sticky bit on a file use the command:

`chmod u+t` *`filename`*

Alternatively use the utility scp instead of chmod, `scp reset` *`filename`*`|all` will correctly set the permissions on files, `scp hide` *`filename`* and `scp reveal` *`filename`* sets and unsets the sticky bit on *`filename`* respectively — see section 4 for a complete description of scp.

2 The .index file

In each directory seen by the server you may have an *optional* index file, `.index`, which documents the files in that directory. The format of an index file is relatively simple and is summarized thus:

```
#
# Comment.
#
filename\
documentation line 1\
documentation line 2\
documentation line 3
#
filename\
documentation line 1\
documentation line 2
#
```

When a user requests an index of a directory the server checks to see if in that directory there exists a `.index` file, and if so the contents are read in and the documentation lines associated with individual files based on the `filename` entry. These comments are then used to document a user's requested index whenever a match on `filename` is encountered. Reading of a `.index` file is a dynamic process, so any changes made will instantly be seen by the server.

It is perfectly legal to only have a subset of the files in a directory appearing in the `.index` file. Any line beginning with a hash (#) is treated as a comment and ignored, the trailing back-slash (\) is treated as a continuation character. Any number of documentation lines can be used, the set being terminated by a line with no trailing backslash. Hashes between documentation entries (as in the above example) are not mandatory, but are recommended as they greatly ease the reading of `.index` files.

For example consider the following `.index` file in the directory `/Foos` containing the files `foobar`, `foobat` and `bartap`:

```
#
# This is my index file for the directory /Foos
#
foobar\
An example foobar program, highly sought after\
in some circles!
#
bartap\
A program to extract foo's from foobar files.
#
```

Upon requesting an index of the `/Foos` directory the user would receive the following reply:

```
******************************************************************************
Index of /Foos
******************************************************************************

<Directories...>
```

```
<Files...>
  foobat                              49502 bytes, Mon Feb 19 18:05:38 1990
  foobar                               1470 bytes, Fri Nov  2 14:16:51 1990
  *
  *  An example foobar program, highly sought after
  *  in some circles!
  *

  bartap                               4864 bytes, Fri Nov  2 14:14:18 1990
  *
  *  A program to extract foo's from foobar files.
  *
```

Notice how `foobat` is included in the listing, but without any documentation. If there were no `.index` file in the `/Foos` directory then an index command would produce the following output:

```
*************************************************************************
Index of /Foos
*************************************************************************

<Directories...>


<Files...>
  foobat                              49502 bytes, Mon Feb 19 18:05:38 1990
  foobar                               1470 bytes, Fri Nov  2 14:16:51 1990
  bartap                               4864 bytes, Fri Nov  2 14:14:18 1990
```

Which, as you can see, is fairly un-informative.

To see the effect of a `.index` file use the `index` command of the scp utility (see Section 4 for a full description of scp). Scp will then produce an index on the *current directory* equivalent to that produced by the Info-Server.

3 The .banner file

In any directory seen by the server an optional `.banner` file may exist. This file represents a banner to be pre-pended to any files the server sends out from the directory. The `.banner` file is used to include information relevant to all files in a directory. The `.banner` file mechanism provides the administrator with a simple method of including such things as copyright notices, general instructions, latest news etc. on all the files in a directory without any redundancy, for example:

status

Status is used to ascertain the current state of the servers. Status reports back whether or not the servers are currently running, followed by a summary of their current activity taken from individual status files, e.g.

```
Info-Server status at Tue Mar 26 13:57:21 1991
Process: Running
PID:     530
State:   Sleeping for 300 seconds

Bulletin-Board status at Tue Mar 26 13:57:22 1991
Process: Running
PID:     740
State:   Mailing file to CCP4 distribution

Digest-Server status at Tue Mar 26 13:57:23 1991
Process: Running
PID:     1075
State:   Saving InfoUnix Volume 10 Issue 23
```

Appendix A Reference card

Server Address (JANET):	Info-Server@uk.ac.daresbury
Server Manager Address (JANET):	Info-Server-Manager@uk.ac.daresbury

Command Format:	`Request: request-type`
	`Topic: topic within request-type`
	`Request: request-type`
	`Topic: topic within request-type`
	`Topic: topic within request-type`
	`Request: end`

Request Types Available:	`Admin`
	`Catalogue`
	`Digests`
	`Help`
	`Index`
	`Mail-Lists`
	`Rindex`
	`Rfc`
	`Sources`

Stand Alone Commands:	`Ping`
	`Quit`
	`Coding:` *`coding-option`*
	`Line-Limit:` *`size`*
	`Return-Path:` *`new-path`*

Coding Options:	`uue`
	`btoa`
	`off`

Line Limit:	`Minimum: 250`
	`Maximum: 6000`
	`Default: 1200`

Admin Topics:	`account`
	`return-path` *`new-path`*
	`coding` *`coding-option`*
	`line-limit` *`size`*
	`mail on|off`

Example Request:	`Request: Sources`
	`Topic: Help`
	`Topic: Index`
	`Topic: Index /Unix`
	`Return-Path: pg@uk.ac.dl.cxa`
	`Coding: btoa`
	`Topic: /Unix/Sun/lpqtool.tar.Z`
	`Request: End`
	`Quit`

Appendix B CCP4 supplement

The server uses a **decnet** interface to retrieve and distribute the **CCP4** software archive from a remote **DEC VAX** host running **VMS**. When requesting files from the server most operations are identical to those described in this manual; there are a few exceptions detailed below:

- Request type is **dni** (abbreviation for **decnet** interface) or **crystal**.
- The index command sends you a copy of a directory listing generated nightly on the remote host.
- Applying the index command to a file name has no effect i.e. `Topic: index [PUBLIC.CCP4]INTRO.TXT`.
- There is no support for the rindex command.
- Filename specification uses the same convention as that on VAX systems, e.g. `[PUBLIC.CCP4]INTRO.TXT`. Use the index command to obtain a listing of valid filenames.
- When a dni transfer fails the request is placed in an ftp queue. The request is then retried using a progressive time delay algorithm (i.e. after 15mins, then 30mins...). The server will attempt to satisfy requests in the **FTP** queue for a maximum period of one week. If, after the expiration of the week, the request has not been satisfied, the user initiating the request will be sent a failure message and the request purged from the ftp queue; this can happen for example when a file appearing in the index has been deleted from the remote host since the index was updated.

  Here is a sample message:

```
Request: Crystal
Topic: Index
Topic: [PUBLIC.AVERG.MDF_MFF]CHKMDF.DOC
Request: End
```

the typical response would be:

```
*Request: Crystal
Valid request type.

*Topic: Index
Sending index.

*Topic: [PUBLIC.AVERG.MDF_MFF]CHKMDF.DOC
DNI request successful - 1657 bytes in 0 seconds.
Sending file...total of 1 part mailed

*Request: End
Valid request type.
```

Appendix C Glossary

Archive	In the context of this document the archive is a collection of data (files) kept together in a directory hierarchy.
ASCII	(*American Standard Code for Information Interchange*) ASCII is a defined standard for the representation of characters in bytes thereby allowing differing computers to exchange information.
Bootstrap	Bootstrapping is the concept of starting from nothing leading to the achivement of something (pulling yourself up by the ...). For example an operating system load from a floppy disk.
CCP4	Collaborative Computing Project Number 4. A series of computing projects run at Daresbury with collaboration from a number of different sites - number 4 is protein crystallography (a science dealing with the form and structure of crystals).
Command Sequence	A command sequence is taken to signify the set of commands found in an electronic mail message sent to the server.
Compress	A utility that reduces the size of a named files using adaptive Lempel-Ziv coding.
DEC	(*Digital Equipment Corporation*) A computer manufacturer.
Decnet	The proprietary network of the digital equipment corporation.
DNI	(*Dec Net Interface*) An interface that in our case allows the info-server to retrieve files from a remote DEC host.
Electronic Mail	A term used meaning the ability for a user to compose a message on their computer and then transmit it to another user using an addressing scheme. The transmitting between computers is usually done over computer networks.
FTP	(*File Transfer Protocol*) A high level protocol for transferring files from one machine to another.
Gateway	A special purpose, dedicated computer that attaches to two or more networks and routes packets from one another.
Index	When applied to databases an index is a collection of keys uniquely identifying records.

Internet	The collection of networks and gateways, including the ARPANET, MILNET and NSFnet, that use the TCP/IP protocol suite and function as a single, cooperative virtual network.
Key	A key (in this context) uniquely identifies records in databases. So, for instance, in a database containing names, addresses and postcodes, the key would be a combination of name and postcode.
Local Area Network	A local area network is one in which the hosts attached to the network physically reside in the same area and the maximun distance between two hosts should not be greater than a set length, i.e. office, building, site.
Network Domain	A domain is a part of a network naming hierarchy. For instance, UK.AC specifies two domains (separated by a dot "."), AC is a sub-domain of the UK domain.
Recon	A utility provided in the Bootstrap directory (`/Bootstrap/recon.c`) which partially automates the reconstruction of files segmented by the server.
RFC	(*Request For Comment*) The name of a series of notes that contain surveys, mesurements, ideas, techniques, amd observations, as well as proposed and accepted Internet protocol standards.
Scp	(*Server Control Program*) A program providing interaction between the servers and a user - see section 4.
Sticky Bit	In the concept of Unix files the sticky bit refers to a bit that, when set, causes the operating system to retain a copy of that file (if it is executable) in memory after it has been run so that subsequent invocations did not require the program to be reloaded into memory - the concept was applied to programs that were likely to be used often.
Tar	A utility to archive and extract multiple files onto a single tar file archive, called a tarfile.
VAX	A type of computer made by the Digital Equipment Corporation.

VMS	The operating system on DEC VAXs.
Wide Area Network	Wide area networks generally connect local area networks together enabling a computer at one site to talk to another at a remote site. There is no general restriction on distances.