

## About This Remade Document

Title: Level 4 Fileserver Functional Specification, Issue 2  
Last remade: 30-Mar-2025  
Repository: <https://github.com/acheton1984/AcornDocsRemade>

This is not quite the beautifully "remastered" document as seen elsewhere, but it is intended to be very similar to the original in an easy-to-read and searchable PDF format.

## Reconstruction Notes

Source scan: <https://www.stardot.org.uk/forums/viewtopic.php?p=430447#p430447>

- \* Recreated from a scan of the original document.
- \* Font sizes, margins, line spacing, and character spacing are only approximate. Some words may appear on different lines, but the overall layout and pagination are preserved.
- \* The ToC on p2+ now has page numbers and previously-missing entries (eg 2.9 Passwords)
- \* References to 'section n' should link to the relevant section
- \* Spelling errors have been corrected where spotted.
- \* Sheets 57, 58 and 59 are missing from the original scan
- \* A title and remade document history page have been added, along with a link to this repository.

## Disclaimer

The underlying content is reproduced or adapted from the original documents for educational purposes and ease of access.

All rights remain with their respective owners.

---

# Level 4 Fileserver

## Functional Specification

---

Distribution:	COMPANY CONFIDENTIAL
Title:	Level 4 Fileserver Functional Specification
Drawing Number:	0370,160/FS
Issue:	2****LIVE****
Author(s):	Martin Neville-Smith
Date:	22nd August 1990
Change No.:	N/A
Last Issue:	Issue 1

---

### Contents:

1. Introduction
2. The User Environment
3. Interfaces
4. File Service - software features
5. Print Service - software features
6. Management Service
7. Manual Design and Contents
8. Artwork design
9. Product organisation and packaging
10. External dependencies
11. Testing Strategy
12. Future enhancements

# Table of Contents

<b>1. INTRODUCTION</b>	<b>7</b>
1.1 General	7
1.2 Conventions and Abbreviations	7
1.3 The Level 4 System	7
1.4 Level 4 Modules	7
1.5 File service	8
1.6 Print service	8
1.7 Management service	8
1.8 Environment	9
1.9 Hardware requirement	9
1.10 Software requirement	9
1.11 Performance	9
<b>2. The User Environment</b>	<b>10</b>
2.1 Handles	10
2.2 Protocol Block Formats	10
2.2.1 Standard Tx Header	10
2.2.2 Standard Rx Header	11
2.3 Standard Data Types	11
2.3.1 File size	11
2.3.2 Attributes	11
2.3.3 Date	12
2.3.4 Access rights	12
2.3.5 Object type	12
2.4 File Server Function Codes	12
2.5 Privilege	13
2.6 Object names	14
2.7 User identifiers	14
2.8 Disc titles	14
2.9 Passwords	14
<b>3. Interfaces</b>	<b>15</b>
3.1 Save	17
3.2 Load	18
3.3 Examine	19
3.4 Catalogue Header	20

3.5 Load as Command	21
3.6 Find (OPEN)	22
3.7 Close	22
3.8 BGET	22
3.9 BPUT	23
3.10 GetBytes	23
3.11 PutBytes	24
3.12 Read Random Access Information	25
3.13 Set Random Access Information	26
3.14 Read Disc Information	26
3.15 Read Current Users	27
3.16 Read Date and Time	27
3.17 Read 'End-of-file' status	27
3.18 Read Object Info	28
3.19 Set File Information	29
3.20 Delete Object	29
3.21 Read User Environment	30
3.22 Set User Boot Option	30
3.23 User log-off	30
3.24 Read User Information	30
3.25 Read File Server Version Number	31
3.26 Read Disc Free Space	31
3.27 Create Directory	31
3.28 Set File Server Date and Time	31
3.29 Create File	32
3.30 Read User Free Space	32
3.31 Set User Free Space	32
3.32 Read Client UserId	33
3.33 Read Current Users (extended)	33
3.34 Read User Information (extended)	33
3.35 Open object for multiple access	34
3.36 Read User Profile	34
3.37 Set User Profile	35
<b>4. File Service - software features</b>	<b>36</b>
4.1 Logfiles	36
4.1.1 Functions logged	36

4.1.2 Format of logfile	37
4.2 Status window	37
4.2.1 Functions displayed in status window	37
4.2.2 Configuration of status window	37
4.3 Management window	37
4.4 Maximum number of objects in a directory	38
4.5 Multi-threading	38
4.6 Multitask switch	38
4.7 Management tools	39
4.7.1 Commands/functions supported:	39
4.8 Objects per user (OPEN)	40
4.9 File systems supported	40
4.10 Record locking	40
4.11 User types	40
4.11.1 System	41
4.11.2 Normal	41
4.11.3 Locked	41
4.11.4 Fixed	41
4.12 Hidden files and directories	41
4.13 Ownership system	41
4.13.1 General	41
4.13.2 Ownership of more than URD	42
4.13.3 Configuration of access	42
4.14 System Internal Number (SIN)	42
4.15 User Root Directory (URD)	42
4.16 Library (LIB)	43
4.17 Currently Selected Directory (CSD)	43
4.18 Passwords	43
4.19 Maximum number of users	44
4.20 Memory management	44
4.21 RISC OS Desktop interface	44
4.22 Standard system timeout	44
4.22.1 Connect time	44
4.23 Server shutdown	44
4.23.1 Fast shutdown	45
4.23.2 Normal shutdown	45

4.24 Caching of files	45
4.25 Error messages	45
<b>5. Print service - software features</b>	<b>47</b>
5.1 General	47
5.2 Print service - setup/configuration	47
5.3 PostScript printers	47
5.4 Printing before received all data	47
5.5 Header file	48
5.6 Footer file	48
5.7 Multiple physical streams	48
5.8 Multiple logical streams	48
5.9 Support of applications that cannot generate PostScript	48
5.10 Desktop interface	48
5.10.1 Main menu	49
5.10.2 Queue information	49
5.11 Timeouts	49
5.12 PostScript printer messages	50
5.13 Banners	50
<b>6. Management service</b>	<b>51</b>
6.1 General	51
6.2 Fileserver Selection	51
6.3 Display list of users in password file	52
6.4 User profile setup	53
<b>7. Manual design and contents</b>	<b>54</b>
7.1 Warnings	54
7.2 Introduction	54
7.3 Part I	54
7.4 Part II Managing L4FS	55
7.5 Part III Management Tasks	56

The following contents are mentioned in, but missing from, the source document:

- 7.6 Printer Server
- 8. Artwork design
- 9. Product organisation and packaging
- 10. External dependencies
- 11. Testing strategy
- 12. Future enhancements

Remade from a scan of an original document. This PDF hosted at:  
<https://github.com/acheton1984/AcornDocsRemade>

# **1. INTRODUCTION**

## **1.1 General**

The Level 4 system is a set of software modules to enable a RISC OS based computer system to provide network facilities for file serving, print serving and management of these facilities. The network supported is Econet and a variety of file systems can be exported as 'discs' to other users of the Econet system. The system is a development of the Acorn Level 3 file server to incorporate performance enhancements and extra features offered by ARM based computer systems. The RISC OS desktop environment provides an opportunity to make the management aspect of the system as simple and effective as possible and allow access to a wide range of advanced server facilities normally found on larger more expensive computer installations.

This document also deals with the interfaces through which clients (station users and user programs) can interact with the Acorn File Server. The interfaces allow a client to manipulate the file server, to gather information about files and directories, other users of the file server and about the configuration of the file server. This document describes the interfaces to the file server operations and the responses which may be expected from the file server.

## **1.2 Conventions and Abbreviations**

All numbers used are given in decimal except where prefixed by '&' to indicate hexadecimal notation. CR is used as an abbreviation for 'carriage return', character &0D. The first byte in a data packet is always referred to as 'byte 1'.

## **1.3 The Level 4 System**

The Level 4 system comprises of a number of 'modules' each of which provides a particular service. The design allows for additional modules to be added later as the requirements are defined.

The current modules are as follows:

File Service

Print Service

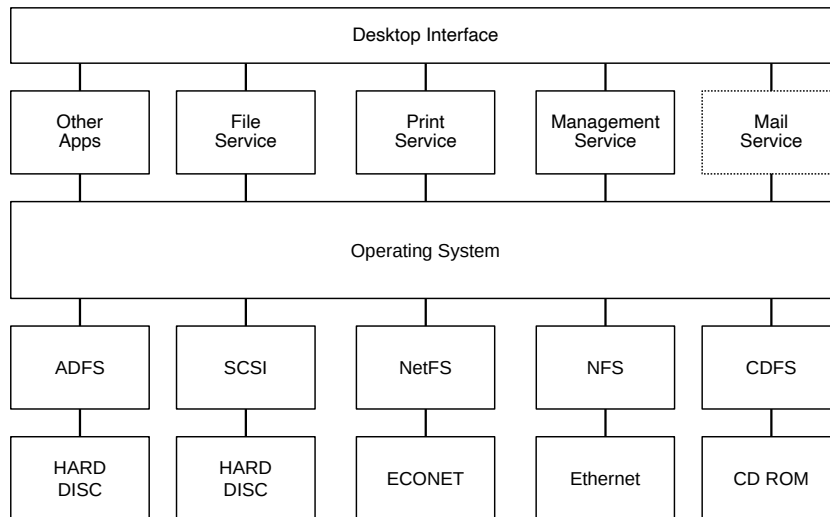
Management Service

The following diagram illustrates how the modules relate to the rest of the system.

## **1.4 Level 4 Modules**

Each module is a separate task running via the RISC OS desktop task manager. They are completely self-contained and do not rely on the presence of any other Level 4 module to provide the functionality required. The modules provide a service, the services are file, print and management. Other modules could be added such as mail, teletext and modem.





## 1.5 File service

This module is known as the 'core module'. It provides the interface to allow multiple clients to communicate with the server via the Econet protocol. It deals with:

- decoding of command line requests (\*command)
- authentication of clients
- access to objects (files and directories) on the file systems exported
- creation, reading and writing of objects
- management of the accounting system
- control of security and access to objects
- examination of directories (catalogue)
- log of client activity

Various functions are provided to support these actions. These are described in full detail in sections 3 and 4.

## 1.6 Print service

Printing is done via this module. Facilities provided are: spooling print jobs, viewing status of print queue, printing when printer free, multiple client printing service. See section 5 for more details.

## 1.7 Management service

Management of users, accounts, server functions are provided by this module. The RISC OS desktop is used to allow a consistent approach to network management. Users are viewed in a similar way to objects in the standard filers and various actions can be carried out in the same way as files are dealt with. This can be performed either on the machine running the file service or from a remote machine. See section 6 for more details.

## **1.8 Environment**

The modules are designed to operate in the RISC OS desktop environment. A mode of operation will allow the file and print service to run outside the desktop thereby gaining the maximum advantage for performance.

## **1.9 Hardware requirement**

The minimum hardware is a 1 Mbyte ARM based computer. A hard disc should be present although a floppy based server is feasible. An Econet module must be fitted and suitable cabling, clocks etc. to allow client machines to communicate with the server.

## **1.10 Software requirement**

A number of relocatable modules are required to support the Level 4 software. These are as follows:

Econet 5.xx

ABCLibrary 2.50

WimpMan 1.00

The host system will require RISC OS 2.00 or later.

The client machine will require NetFS 5.xx if broadcast loading is required.

All other versions of Econet client software will be supported except NFS 3.34.

## **1.11 Performance**

The Level 4 Fileserver, when used in standalone mode, with logfile and status window disabled will be at least as fast as a Filestore, and should be faster in non-Econet limited functions. When using it in Desktop mode it should be possible in a lightly loaded network to perform useful work on the server machine. Note that the introduction of more powerful future machines may not increase server performance significantly, but may make considerable difference to the amount of power available to other desktop applications.

## 2. The User Environment

### 2.1 Handles

A client is identified and authenticated on the file server by its station number and three HANDLES. These handles are created by the file server by opening directories and identify to the file server the environment in which to interpret commands and to look-up filenames presented by the client. The handles are created by the file server when a user logs on and are closed when the user logs off again. The three handles which comprise the user environment are:

The Currently Selected Directory (CSD).

The directory in which to look-up all filenames which do not refer to the root directory.

The User Root Directory (URD).

The directory where a user is placed after logging-on and where he is returned after a \*DIR command.

The Library (LIB).

The directory where unrecognised commands are looked-up if the filename is not found in the CSD.

Usually the client machine software deals with the manipulation of these handles but it is possible for a user to define his own environment by opening several directories and declaring a set of these handles as representing the current environment. This enables the user to execute a command in a number of different environments.

### 2.2 Protocol Block Formats

#### 2.2.1 Standard Tx Header

The format of initial protocol blocks sent to the file server take a standard form. This form is known as the STANDARD TX HEADER.

Byte 1 - Reply Port

Byte 2 - Function Code

Byte 3 - Handle for User Root Directory

Byte 4 - Handle for Currently Selected Directory

Byte 5 - Handle for Library Directory

The Reply Port is the port on which the client station is prepared to receive a response from the file server. The Function Code indicates to the file server which operation to perform. The three handles define the environment for this command as described above. The command is sent to the file server on port &99, this port is called the command port.

### 2.2.2 Standard Rx Header

The format of responses from the file server also take a standard form, known as the STANDARD RX HEADER.

Byte 1 - Command Code

Byte 2 - Return Code

The Command Code indicates to the client what action (if any) the client should take upon receiving this response. The Command Code byte is only valid when the operation performed was a 'Decode Command Line', except where noted.

The Return Code is an indication to the client of any error status which has arisen as a result of attempting to execute the command. A return code of zero indicates that the command step completed successfully otherwise the return code is the error number indicating what error has occurred. If the return code is non-zero then the remainder of the message contains an ASCII string terminated by a carriage return which describes the error.

## 2.3 Standard Data Types

The file server protocols use standard data types for most operations. In all cases multi-byte values are stored low byte first.

All specifications of object and user names may be 8 bit values. Note: &80 is reserved and means end of data in some file server protocols.

### 2.3.1 File size

File servers only support files up to 16 megabytes so all pointer operations, and file length indications use 24 bit quantities, stored low byte first. Econet protocols do not support larger files.

### 2.3.2 Attributes

The attributes for an object are stored in a single byte with the bit set to '1' meaning as follows:

Bit 0 ==> Public read access

Bit 1 ==> Public write access

Bit 2 ==> Owner read access

Bit 3 ==> Owner write access

Bit 4 ==> Owner locked

Bit 5 ==> Is a directory

Bit 6 ==> Directory protected

Note that public lock is always implicitly set

### 2.3.3 Date

The date is stored in two bytes, the first byte contains the day of the month in bits 0 to 4 this value ranges from 1 to 31 depending on the month. The second byte contains the month number in bits 0 to 3, this value is between 1 and 12. The year is encoded as a seven bit number (0 to 127) and is given as an offset from 1981. These seven bits are stored as follows, lower four bits in bits 4 to 7 of byte two, and the upper three bits in bits 5 to 7 of byte one.

### 2.3.4 Access rights

The access rights to a directory are encoded in a single byte. The value zero for Owner Access, or the value &FF for Public Access.

### 2.3.5 Object type

The object type is encoded in a single byte as follows, zero means no object, one means the object was a file, and two means the object was a directory.

## 2.4 File Server Function Codes

A summary of the file server function codes is given below.

Function	Privilege	Logged-on	Description
0	No	Yes *	Decode command line
1	No	Yes	Save file
2	No	Yes	Load file
3	No	Yes	Examine
4	No	Yes	Catalogue header
5	No	Yes	Load as command
6	No	Yes	Open object
7	No	Yes	Close object
8	No	Yes	Get byte
9	No	Yes	Put byte
10	No	Yes	Get bytes
11	No	Yes	Put bytes
12	No	Yes	Read random access arguments
13	No	Yes	Set random access arguments
14	No	No	Return disc name information
15	No	No	Return logged-on users information
16	No	Yes	Return file server date and time
17	No	Yes	End-of-file status
18	No	Yes	Read object information
19	No	Yes	Set object attributes

Function	Privilege	Logged-on	Description
20	No	Yes	Delete object
21	No	Yes	Return user environment information
22	No	Yes	Set user log-on option
23	No	Yes	Logoff
24	No	Yes	Return single user's information
25	No	No	Return file server version number
26	No	Yes	Return disc free space information
27	No	Yes	Create directory, specifying size
28	Yes	Yes	Set file server date and time
29	No	Yes	Create file
30	No	Yes	Read user free space
31	Yes	Yes	Set user free space
32	No	Yes	Read client UserId
33	No	Yes	Return logged-on user's extended information
34	No	Yes	Return single user's extended information
35	No	Yes	Open object for multiple access
36	Yes	Yes	Read user profile
37	Yes	Yes	Set user profile

Function = Function code number

Priv = If the client has to have privilege

Logged-on = If the client has to be logged-on

\* There is no need to be logged-on to decode the \*I AM command.

## 2.5 Privilege

A client may have different levels of privilege, these are explained in detail in section 4.11, User types. The values for each privilege level are as follows:

&00 ==> Locked

&40 ==> Fixed

&80 ==> Normal

&FF ==> System manager

## **2.6 Object names**

Object names are currently limited by the filing system, usually 10 characters. They may contain 8 bit values and are not case sensitive.

## **2.7 User identifiers**

User identifiers are normally 10 characters with a provision for group identifiers ie: <group>.<name> each of 10 characters.

## **2.8 Disc titles**

The disc title is the name that the server exports to network clients. The length is determined by the host file system however it must start with a letter and consists of alphanumeric characters (ie: A-Z, 0-9), “-” and “\_”.

## **2.9 Passwords**

The password file associated with the server holds encrypted passwords, privilege levels, boot options and space allocation information for each user. See Section 4.18.

### 3. Interfaces

This section deals individually with each of the commands and functions available to client software. The exchange of packets is detailed and the format of requests and responses is given. A number of the operations performed by the file server are initiated by the sending of a command line

Client (command port):

Bytes 1-5	Standard Tx Header, Function code = 0 (&00)
Bytes 6+	Command line terminated by CR

File Server (reply port):

Bytes 1-2	Standard Rx Header
Bytes 3+	Command dependent results

The command line syntaxes which the Acorn File Server will accept are as follows: (Commands in bold are new Level 4 commands)

```
*Access <file/directory name> <new access string>
*Bye
*CDir <directory name>
*Delete <file/directory name>
*Dir <optional directory name>
*I am <user name> [<password>]
*Info <file/directory name>
*Lib <directory name>
*Logon <user name> [<password>]
*Pass <old password <new password>
*Rename <old file name> <new file name>
*SDisc <disc name>
```

Management specific commands:

```
*FSShutdown <time>
*Logoff <User name/number>
*NewUser <user name>
*Priv <user name> [<new privilege status>]
*RCopy <source spec> <destination spec> [<options>]
*RemUser <user name>
```

If the command requires more action by the client then the command code in the Standard Rx Header indicates what command line the file server has decoded. The file server will also return any decoded parameters or data which the client will need to complete the command. The possible command codes that



the file server may return are:

0	No Action, command complete
1	reserved (for SAVE operation)
2	reserved (for LOAD operation)
3	reserved
4	*Info
5	*I am
6	*SDisc
7	*Dir
8	Unrecognised command
9	*Lib
10	disc information function code 14 called
11	users information, function code 15 called

Return from \*INFO

File server (reply port):

Bytes 1-2	standard Rx header, Command code = 4
Bytes 3+	Character string of information terminated by negative byte (&80)

The format of the info string is as follows:

Object name padded with spaces (10 characters), space, load address padded with zeros (&20) (8 characters), space, execution address padded with zeros (8 characters), 3 spaces, length padded with zeros (6 characters), 3 spaces, access details, 5 spaces, date (DD:MM:YY), space, SIN padded with zeros (6 characters).

Return from \*IAM

File server (reply port):

Bytes 1-2	standard Rx header, Command code = 5
Byte 3	New URD handle
Byte 4	New CSD handle
Byte 5	New LIB handle
Byte 6	Boot option (Bits 0 to 3 significant)

Return from \*SDISC

File server (reply port):

Bytes 1-2	standard Rx header, Command code = 6
-----------	--------------------------------------

Byte 3	New URD handle
Byte 4	New CSD handle
Byte 5	New LIB handle

Return from \*DIR

File server (reply port):

Bytes 1-2	standard Rx header, Command code = 7
Byte 3	New CSD handle

Return from Unrecognised command

File server (reply port):

Bytes 1-2	standard Rx header, Command code = 8
Byte 3+	command string terminated by CR

Return from \*LIB

File server (reply port):

Bytes 1-2	standard Rx header, Command code = 9
Byte 3	New LIB handle

### 3.1 Save

This is the actual save operation, this protocol is used after the command line has been decoded, either by the file server or the local operating system.

Client (command port):

Byte 1	reply port
Byte 2	function code = 1 (&01)
Byte 3	acknowledge port
Byte 4	CSD handle
Byte 5	LIB handle
Bytes 6-9	file load address
Bytes 10-13	file execute address
Bytes 14-16	file size
Bytes 17+	file name terminated by CR

File server (reply port):

Bytes 1-2	standard Rx header, Command code = 3 if leaf name is returned otherwise it is 0
-----------	---

Byte 3	data port
Bytes 4-5	maximum data block size
Bytes 6+	leaf name terminated by CR (if returned)

The client and file server now enter the 'data exchange phase' of the protocol where the file server acknowledges the receipt of each data packet. If there is no data to be sent (eg a zero length file) then this phase is omitted. If the file server detects an error during the data transfer phase (eg a disc error) then the 'data exchange phase' is allowed to complete but the SAVE operation is aborted. The error status is returned in the return code of the 'final acknowledge.'

Client (data port):

A block of data, up to the maximum data block size

File server (acknowledge port):

A single byte of undefined value

When the final data block has been received by the file server the 'data acknowledge' is replaced by the 'final acknowledge' which is the terminating packet of the protocol.

File server (reply port):

Bytes 1-2	standard Rx header
Byte 3	Attributes
Bytes 4-5	Date

### 3.2 Load

This is the actual load operation, this protocol is used after the command line has been decoded, either by the file server or the local operating system.

Client (command port):

Byte 1	reply port
Byte 2	function code = 2 (&02)
Byte 3	data port
Byte 4	CSD handle
Byte 5	LIB handle
Bytes 6+	file name terminated by CR

File server (reply port):

Bytes 1-2	standard Rx header, Command code is 14 if leaf name is resolved otherwise it is 0
Bytes 3-6	file load address

Bytes 7-10	file execute address
Bytes 11-13	file size
Byte 14	file access
Bytes 15-16	file creation date
Bytes 17+	leaf name terminated by CR this resolves wild- carded filename specifiers (if returned)

If the file is of zero length then the 'data transfer' phase is omitted. If the file server detects an error (eg disc error) then the required amount of data will be sent but its data content is undefined.

File server (data port):

data blocks of undefined size repeated until 'file size' data has been sent.

maximum data block size is currently 4k

File server (reply port):

Bytes 1-2	standard Rx header
-----------	--------------------

### 3.3 Examine

Client (command port):

Bytes 1-5	standard Tx header, Function code = 3 (&03)
Byte 6	argument to examine function
Byte 7	directory entry point (0-255)
Byte 8	number of entries to examine (0-255)
Bytes 9+	name of directory to be examined

The argument passed in byte 6 specifies the format and amount of information to be returned by the file server.

Argument	= 0 ==> all information, machine readable format
	= 1 ==> all information, character string
	= 2 ==> file title only
	= 3 ==> file title and access, character string

The directory entry point gives the entry number within the directory from which to examine. Conventionally the first entry in a directory is entry number zero.

The number of entries to examine specifies how many entries are to be examined, so is usually determined by the buffer space available to the client. A parameter of zero in this case conventionally demands that all entries in the directory from the entry point to the end of the directory be examined.

Where information is returned in character string format individual entries are delimited by zero bytes (&00), the final entry being terminated by a negative byte (&80). Carriage returns may occur within such strings. Where data is returned in machine readable format the entries consist of a defined number of bytes and so there are no delimiters between entries although the final entry is terminated by a negative byte (&80).

File server (reply port):

Bytes 1-2	standard Rx header
Byte 3	number of entries actually examined
Byte 4	directory cycle number
Bytes 5+	argument dependent information

Argument = 0

Bytes 5-14	object name padded to 10 characters with spaces
Bytes 15-18	load address
Bytes 19-22	execute address
Byte 23	Attributes
Bytes 24-25	Date
Bytes 26-28	System Internal Name (SIN)
Bytes 29-31	object length

Argument = 1

Bytes 5+	character string of all information data terminated by negative byte (&80)
----------	--

Argument = 2

Byte 5	10 (object name length for BBC MOS)
Bytes 6+	object name padded with spaces

Argument = 3

Bytes 5+	object name padded to 10 characters
----------	-------------------------------------

followed by formatted access string

(total length=18 characters)

### 3.4 Catalogue Header

Client (command port):

Bytes 1-5	standard Tx header, Function code = 4 (&04)
Bytes 6+	directory name, CR returns information for CSD

File server (reply port):

Bytes 1-2	standard Rx header
Bytes 3-13	last component of directory name padded with spaces
Byte 14	character indicating ownership of directory
Bytes 15-17	three space characters (&20)
Bytes 18-33	current disc name padded with spaces terminated by CR, negative byte

### 3.5 Load as Command

This is exactly the same as Load (Function code = 2), except that the file name is looked up in the CSD and then in the LIB. The error returned if the file name is not found in either directory is 'Bad command'.

Client (command port):

Byte 1	reply port
Byte 2	function code = 5 (&05)
Byte 3	data port
Byte 4	CSD handle
Byte 5	LIB handle
Bytes 6+	file name terminated by CR

File server (reply port):

Bytes 1-2	standard Rx header
Bytes 3-6	file load address
Bytes 7-10	file execute address
Bytes 11-13	file size
Byte 14	file access
Bytes 15-16	file creation date
Bytes 17+	leaf name terminated by CR this resolves wild-carded filename specifiers

If the file is of zero length then the 'data transfer' phase is omitted. If the file server detects an error (eg disc error) then the required amount of data will be sent but its data content is undefined.

File server (data port):

data blocks of undefined size repeated until 'file size' data has been sent

File server (reply port):

Bytes 1-2	standard Rx header
-----------	--------------------

### 3.6 Find (OPEN)

This function code creates a handle for the object specified with the access type requested. Such handles are used for performing random access operations and also for manipulating the user's environment. An Object will be opened only if the client has the necessary access rights to the object. When opening directories these must be specified as already existing. A file can be opened several times for reading only but only once for update. A file will be created with default size of 400 bytes if it does not already exist, and it is opened for update, and the client specifies a new file (byte 6 = 0). Limits to the number of handles a client is allowed to have open at any one time are imposed. This is dependent on the machine type. BBC machines support 8, Master series support 16 and Archimedes clients are allowed 255. These values include 3 handles which are automatically allocated when the client logs on therefore a BBC machine will be able to open a further 5 objects. See section 2.1 for more information.

Client (command port):

Bytes 1-5	standard Tx header, Function code = 6 (&06)
Byte 6	= 0 ==> create a new file, deleting existing data non 0 ==> object must already exist
Byte 7	= 0 ==> open object for update non 0 ==> open object for reading only
Bytes 8+	object name terminated by CR

File server (reply port):

Bytes 1-2	standard Rx header, Command code = 1 if leafname is returned otherwise it is 0
Byte 3	handle
Byte 4	leaf name terminated by CR (if returned)

### 3.7 Close

This function indicates to the file server that the handle passed as argument is no longer needed and that all of the updated data in the file should be written out to the disc. A handle of zero indicates to the file server that all handles to open FILES are to be closed. This call does not close handles to directories if the handle given is zero.

Client (command port):

Bytes 1-5	standard Tx header, Function code = 7 (&07)
Byte 6	handle

File server (reply port):

Bytes 1-2	standard Rx header
-----------	--------------------

### 3.8 BGET

The next four function codes deal with the facilities that the file server provides to enable the user to perform

random access operations on open files. These operations have an additional protocol to ensure the integrity of the data exchanged, provided by a SEQUENCE NUMBER. The sequence number is a single bit held in both client and file server which differentiates between successive reads of a file using the pointer held in the file server, and repeated reads of the same byte because the operation failed at the first attempt. The sequence number is sent in the least significant bit of the flag byte of the Econet control block. The file server will return its copy of the sequence number with the data to allow the client to detect data sequencing errors.

The client should invert his copy of the sequence number after every successful transaction with the file server. If the client detects a data packet with the incorrect sequence number then the client should be prepared to repeat the request.

This function code reads a byte from the file at the position specified by the file server's internal file pointer.

Chen (command port):

Byte 1	reply port
Byte 2	function code = 8 (&08)
Byte 3	file handle

File server (reply port):

Bytes 1-2	standard Rx header
Byte 3	byte read, &FE if reading first byte after file end
Byte 4	= &00 ==> normal read operation = &80 ==> last byte in the file = &C0 ==> first byte after file end

### 3.9 BPUT

This function code writes a single byte to the file at the position indicated by the file server's internal file pointer.

Client (command port):

Byte 1	reply port
Byte 2	function code = 9 (&09)
Byte 3	file handle
Byte 4	byte to be written

File server (reply port):

Bytes 1-2	standard Rx header
-----------	--------------------

### 3.10 GetBytes

This operation allows the client to read blocks of data. The client may supply an offset within the file at



which to start the operation, or may use the sequential file pointer maintained by the file server. The protocol includes a sequence number as described for BGET and BPUT.

Client (command port):

Byte 1	reply port
Byte 2	function code = 10 (&0A)
Byte 3	data port
Byte 4	CSD handle
Byte 5	LIB handle
Byte 6	file handle
Byte 7	zero ==> use supplied offset non-zero ==> use file server sequential pointer
Bytes 8-10	number of bytes to transfer
Bytes 11-13	file offset (if supplied)

File server (reply port)

Bytes 1-2	standard Rx header
-----------	--------------------

If the transfer is of zero length then the 'data transfer' phase is omitted. If the file server detects an error (eg disc error) then the required amount of data will be sent but its data content is undefined. If a read extends over the end of the file then the requested amount of data will be returned but the number of valid data bytes will be less than the amount of data requested. The remaining data is undefined.

File server (data port):

data blocks of undefined size repeated until 'transfer size' data has been sent

File server (reply port):

Bytes 1-2	standard Rx header
Byte 3	= &00 ==> all OK = &80 ==> read includes last byte in file
Bytes 4-6	number of valid data bytes transferred

### 3.11 PutBytes

This operation allows the client to write blocks of data. The client may supply an offset within the file at which to start the operation, or may use the sequential file pointer maintained by the file server. The protocol includes a sequence number as described for BGET and BPUT.

Client (command port):

Byte 1	reply port
Byte 2	function code = 11 (&0B)

Byte 3	acknowledge port
Byte 4	CSD handle
Byte 5	LIB handle
Byte 6	file handle
Byte 7	zero ==> use supplied offset non-zero ==> use file server sequential pointer
Bytes 8-10	number of bytes to transfer
Bytes 11-13	file offset (if supplied)

#### File server (reply port)

Bytes 1-2	standard Rx header
Byte 3	data port
Bytes 4-5	maximum data block size

The client and file server now enter the 'data exchange phase' of the protocol where the file server acknowledges the receipt of each data packet. If there is no data to be sent then this phase is omitted. If the file server detects an error during the data transfer phase (eg a disc error) then the 'data exchange phase' is allowed to complete but the operation is aborted. The error status is returned in the return code of the 'final acknowledge'.

#### Client (data port):

A block of data, up to the maximum data block size

#### File server (acknowledge port):

A single byte of undefined value

When the final data block has been received by the file server the 'data acknowledge' is replaced by the 'final acknowledge' which is the terminating packet of the protocol.

#### File server (reply port):

Bytes 1-2	standard Rx header
Byte 3	Undefined
Bytes 4-6	number of valid data bytes transferred

### 3.12 Read Random Access Information

This function code allows the client to discover information about files for which he currently has handles.

#### Client (command port):

Bytes 1-5	standard Tx header, Function code = 12 (&0C)
-----------	--

Byte 6	file handle
Byte 7	= 0 ==> sequential file pointer
	= 1 ==> file extent (the amount of valid data)
	= 2 ==> file size (the space allocated for the file)

File server (reply port):

Bytes 1-2	standard Rx header
Bytes 3-5	information requested

### 3.13 Set Random Access Information

Client (command port):

Bytes 1-5	standard Tx header, Function code = 13 (&0D)
Byte 6	file handle
Byte 7	= 0 ==> set sequential file pointer
	= 1 ==> get file extent
Bytes 8-10	value to set

File server (reply port):

Bytes 1-2	standard Rx header
-----------	--------------------

### 3.14 Read Disc Information

This function returns the disc configuration of the file server. Conventionally the first drive in the file server is number zero and all drives are numbered from there, so the second drive is drive one, etc. This number is NOT the same as the drive number returned, the returned drive number is the physical drive number. If the number of drives to interrogate is zero then the file server will return information on all drives in the system.

Client (command port):

Bytes 1-5	standard Tx header, Function code = 14 (&0E)
Byte 6	first drive number (ordinal)
Byte 7	number of drives to interrogate

File server (reply port):

Bytes 1-2	standard Rx header
Byte 3	number of drives found
Byte 4	drive number of first drive requested (physical)
Byte 5-20	disc name padded with spaces
Bytes 21+	further entries in same format

### 3.15 Read Current Users

This function returns the currently logged-on users of the file server, their station numbers and associated privileges. Conventionally the logged-on user entries start at zero. A request for zero entries will return information for all users, commencing at the start entry.

Client (command port):

Bytes 1-5	standard Tx header, Function code = 15 (&0F)
Byte 6	start entry
Byte 7	number of entries to get

File server (reply port):

Bytes 1-2	standard Rx header
Byte 3	number of entries returned
Byte 4	station number of first user
Byte 5	network number of first user
Bytes 6+	first user name, terminated by CR
Byte n	privilege

### 3.16 Read Date and Time

It is not necessary to be logged-on to the file server to use this function code.

Client (command port):

Bytes 1-5	standard Tx header, Function code = 16 (&10)
-----------	--

File server (reply port)

Bytes 1-2	standard Rx header
Bytes 3-4	Date
Byte 5	Hours (0 to 23)
Byte 6	Minutes (0 to 59)
Byte 7	Seconds (0 to 59)

### 3.17 Read 'End-of-file' status

This function is valid for file handles only.

Client (command port):

Bytes 1-5	standard Tx header, Function code = 17 (&11)
Byte 6	file handle

File server (reply port):

Bytes 1-2	standard Rx header
Byte 3	== 0 ==> pointer within file =/ 0 ==> pointer outside file

### 3.18 Read Object Info

Client (command port):

Bytes 1-5	standard Tx header, Function code = 18 (&12)
Byte 6	= 1 ==> read object creation date = 2 ==> read load and execute address = 3 ==> read object length = 4 ==> read object type and attributes = 5 ==> read all object information = 6 ==> read access rights and cycle number of directory = 7 ==> read unique identifier
Bytes 7+	object name terminated by CR

The results returned depend on the argument passed with the call.

#### Arguments 1-5

File server (reply port):

Bytes 1-2	standard Rx header
Byte 3	Object type
Bytes 4+	results returned in the following order: load address, execute address, length, attributes, date, access rights

#### Argument 6

File server (reply port):

Bytes 1-2	standard Rx header
Byte 3	undefined
Byte 4	0
Byte 5	10 (length of directory name)
Bytes 6-15	directory name padded with spaces
Byte 16	access rights
Byte 17	cycle number of directory

## Argument 7

### File server (reply port):

Bytes 1-2	standard Rx header
Byte 3	Object type
Byte 4-9	unique identifier for that object on that server (SIN + Disc number)
Bits 0-23 ==>	file system SIN
Bits 24-31 ==>	server disc number
Bits 32-47 ==>	filing system number

## 3.19 Set File Information

### Client (command port):

Bytes 1-5	standard Tx header, Function code = 19 (&13)
Byte 6	= 1 ==> set load address, execute address, and attributes = 2 ==> set load address = 3 ==> set execute address = 4 ==> set attributes = 5 ==> set creation date
Bytes 7+	parameters to set (depend on byte 6)
Bytes n+	file name terminated by CR

### File server (reply port):

Bytes 1-2	standard Rx header
-----------	--------------------

## 3.20 Delete Object

### Client (command port):

Bytes 1-5	standard Tx header, Function code = 20 (&14)
Bytes 6+	object name terminated by CR

### File server (reply port):

Bytes 1-2	standard Rx header
Bytes 3-6	load address
Bytes 7-10	execute address
Bytes 11-13	length
Byte 14 -	
Byte 15	attributes

Bytes 16-17      date

### 3.21 Read User Environment

Client (command code):

Bytes 1-5            standard Tx header, Function code = 21 (&15)

File server (reply port):

Bytes 1-2            standard Rx header

Byte 3                16 (length of disc name)

Bytes 4-19           name of currently selected disc padded with spaces

Bytes 20-29          name of CSD padded with spaces

Bytes 30-39          name of LIB padded with spaces

### 3.22 Set User Boot Option

Client (command code):

Bytes 1-5            standard Tx header, Function code = 22 (&16)

Byte 6                boot option, only the bottom four bits are used

File server (reply port):

Bytes 1-2            standard Rx header

### 3.23 User log-off

Client (command port):

Bytes 1-5            standard Tx header, Function code = 23 (&17)

File server (reply port):

Bytes 1-2            standard Rx header

### 3.24 Read User Information

Client (command port):

Bytes 1-5            standard Tx header, Function code = 24 (&18)

Bytes 6+             user name terminated by CR

File server (reply port):

Bytes 1-2            Standard Rx header

Byte 3                privilege

Byte 4                station number

Byte 5                      network number

### 3.25 Read File Server Version Number

Client (command port):

Bytes 1-5                  standard Tx header, Function code = 25 (&19)

File server (reply port):

Bytes 1-2                  standard Rx header

Bytes 3-11                a text string describing the file server type

Byte 12                    a space character (ASCII &20)

Bytes 13-16              a text string of the form n.xy which is the version

### 3.26 Read Disc Free Space

Client (command port):

Bytes 1-5                  standard Tx header, Function code = 26 (&1A)

Bytes 6+                  disc name terminated by CR

File server (reply port):

Bytes 1-2                  standard Rx header

Bytes 3-5                  free space on disc (in sectors of &100 bytes)

Bytes 6-8                  disc size (in sectors of &100 bytes)

### 3.27 Create Directory

Client (command port):

Bytes 1-5                  standard Tx header, Function code = 27 (&1B)

Byte 6                    maximum number of sectors to allocate

Bytes 7+                  name of directory terminated by CR

File server (reply port):

Bytes 1-2                  standard Rx header

### 3.28 Set File Server Date and Time

It is necessary to be logged-on to the file server, with privilege, to set the date and time parameters.

Client (command port):

Bytes 1-5                  standard Tx header, Function code = 28 (&1C)

Bytes 6-7                  Date



Byte 8	Hours (0 to 23)
Byte 9	Minutes (0 to 59)
Byte 10	Seconds (0 to 59)

File server (reply port):

Bytes 1-2	standard Rx header
-----------	--------------------

### 3.29 Create File

This function creates a file of the size and type specified. The contents will automatically be set to zeros for security reasons.

Client (command port):

Bytes 1-5	standard Tx header, Function code = 29 (&1D)
Bytes 6-9	file load address
Bytes 10-13	file execute address
Bytes 14-16	file length
Bytes 17+	file name terminated by CR

File server (reply port):

Bytes 1-2	standard Rx header
Byte 3	Attributes
Bytes 4-5	Date

### 3.30 Read User Free Space

Client (command port):

Bytes 1-5	standard Tx header, Function code = 30 (&1E)
Bytes 6+	UserId for free space reading terminated by CR
	CR alone means return the free space of the client

File server (reply port):

Bytes 1-2	standard Rx header
Bytes 3-6	available space for UserId, in bytes

### 3.31 Set User Free Space

This function code is only legal for privileged users. The UserId specified is that of the client whose space allocation is to be amended.

Client (command port):

Bytes 1-5	standard Tx header, Function code = 31 (&1F)
Bytes 6-9	new value for available space, in bytes
Bytes 10+	UserId terminated by CR

File server (reply port):

Bytes 1-2	standard Rx header
-----------	--------------------

### 3.32 Read Client UserId

Client (command port):

Bytes 1-5	standard Tx header, Function code = 32 (&20)
-----------	--

File server (reply port):

Bytes 1-2	standard Rx header
Bytes 3+	UserId of client terminated by CR

### 3.33 Read Current Users (extended)

This function returns the currently logged-on users of the file server, their station numbers, task numbers, and associated privileges. Conventionally the logged-on user entries start at zero. A request for zero entries will return information for all users, commencing at the start entry.

Client (command port):

Bytes 1-5	standard Tx header, Function code = 33 (&21)
Byte 6	start entry
Byte 7	number of entries to get

File server (reply port):

Bytes 1-2	standard Rx header
Byte 3	number of entries returned
Byte 4	station number of first user
Byte 5	network number of first user
Byte 6	task number
Bytes 7+	first user name, terminated by CR
Byte n	privilege

### 3.34 Read User Information (extended)

Client (command port):

Bytes 1-5	standard Tx header, Function code = 34 (&22)
-----------	--

Bytes 6+            user name terminated by CR

**File server (reply port):**

Bytes 1-2            standard Rx header  
Byte 3                privilege  
Byte 4                station number  
Byte 5                network number  
Byte 6                task number

### 3.35 Open object for multiple access

This function is similar to function 6 (Open) except it will allow multiple clients to write to a file simultaneously. This function is to be used in conjunction with a record locking feature to be provided at a later date. Extreme caution must be used when using this function.

**Client (command port):**

Bytes 1-5            standard Tx header, Function code = 35 (&23)  
Byte 6                = 0 ==> create a new file, deleting existing data  
                      non 0 ==> object must already exist  
Byte 7                = 0 ==> open object for update  
                      non 0 ==> open object for reading only  
Bytes 8+             object name terminated by CR

**File server (reply port):**

Bytes 1-2            standard Rx header  
Byte 3                handle

### 3.36 Read User Profile

This function allows the system manager to read all the details concerning the users of the system. The password file will then be updated accordingly.

**Client (command port):**

Bytes 1-5            standard Tx header, Function code = 36 (&24)  
Bytes 6-9            password file entry number

**File server (reply port):**

Bytes 1-2            standard Rx header  
Byte 3                privilege  
Byte 4                boot option  
Bytes 5-8            space allocation (in bytes)

Byte 9	station number (0 if not logged on)
Byte 10	net number (0 if not logged on)
Byte 11	allow user to logon
Bytes 12+	password terminated by CR
Bytes n+	user root directory (URD) terminated by CR
Bytes n+	currently selected directory (CSD) terminated by CR
Bytes n+	currently selected library (LIB) terminated by CR

### 3.37 Set User Profile

#### Client (command port):

Bytes 1-5	standard Tx header, Function code = 37 (&25)
Bytes 6+	user name terminated by CR

#### File server (reply port):

Bytes 1-2	standard Rx header
Byte 3	privilege
Byte 4	boot option
Bytes 5-8	space allocation (in bytes)
Byte 9	station number (0 if not logged on)
Byte 10	net number (0 if not logged on)
Byte 11	allow user to log on
Bytes 12+	password terminated by CR
Bytes n+	user root directory (URD) terminated by CR
Bytes n+	currently selected directory (CSD) terminated by CR
Bytes n+	currently selected library (LIB) terminated by CR

## **4. File Service - software features**

### **4.1 Logfiles**

These are text files which are generated by the server in order to record certain operations that are performed on the server. The aim of these files is to provide a record of usage of the system in order to derive statistics of usage of the systems or security protection ie: record of failed logons etc. The desktop interface allows file logging to be turned on or off and any individual function may be selected for logging by means of a toggle switch.

#### **4.1.1 Functions logged**

The following functions can be logged: (\*command equivalent in brackets)

1. Valid log on (\*I AM)
2. Failed log on
3. Log off (\*BYE)
4. Save file (\*save <file>)
5. Load file (\*load <file>)
6. Open file
7. Close file
8. Read disc information (\*FSLIST)
9. Create directory (\*CDIR)
10. Server start up
11. System user log on
12. Creation of new user
13. Removal/deletion of user
14. Modification of space accounts

### 4.1.2 Format of logfile

The file is ASCII text format and generally consists of a textual description of the function followed by the date, time and station number of the client requesting the service.

NOTE: This file should be deleted or moved periodically due to the disc space usage. If the server is operating for a long period the log file could be very large especially if many functions have been selected for logging. A circular buffer system can be selected if required which will overwrite the beginning of the file when the allocated file size has been used up. This is defined by using the Management application. A delimiter of zero (&0) is used to indicate the current log position.

## 4.2 Status window

The status window is a window that can be displayed when running on the desktop. It is invoked by clicking on the !server icon on the icon bar with the select button. It is provided to allow the user of the server to monitor the activity being carried out on the file server.

### 4.2.1 Functions displayed in status window

This window displays some of the commands and requests that are sent to the server. A variety of functions are displayed. These are as follows:

```
*Commands eg: *I AM FRED
current version number on startup
error messages
Load operations
Save operations
Open object
Close object
Cat (Examine) directory
```

### 4.2.2 Configuration of status window

The status display window can be configured to open automatically when the server is started. Note: Due to the amount of processing required to display text in a window, it is recommended that this window is not open during normal operation of the file server; a significant increase in performance will be achieved when the window is closed. Note: The log file options bear no relation to the operations displayed in the status window.

## 4.3 Management window

This window will allow the user to configure and view the options which may be changed to alter the way in which the server operates. The options are as follows:

Maximum number of users

Global hidden objects

Open status window on starting server

Allow logon or produce error: 'Server not available'

#### **4.4 Maximum number of objects in a directory**

Where the host filing system has a limit to the number of entries in a directory (ie: ADFS, SCSIFS) the server will perform automatic directory extension. This will appear transparently to the client on the network as simply a large directory however when viewed from the RISC OS desktop on the server machine the directory will be seen to contain a special directory where the rest of the objects may be found. This type of extension will put an overhead on the system when viewing directories, loading and saving files, as the system may have to search a number of sub-directories in order to locate the object required. When an object is to be created in a directory, any extensions will have to be searched to ensure duplication of object names does not occur. The system will prevent the directory becoming full by creating a sub-directory and placing the new object in that directory. The directory name will be made of a top bit set space (&A0) and will appear as a directory with no name under the filer on the host machine.

The server limits the number of directory entries to 511. If a filing system supports more than this number, only the first 511 will be accessible.

#### **4.5 Multi-threading**

The server code is multi-threaded. This allows any function to be operating on different sets of data at anytime. This will allow multiple clients carrying out the same operation to share the processing time available from the server. For example a load operation from 10 clients no longer requires the server to service each client in sequence thereby reducing the chance that several clients will time out due to not receiving a response from the server. Most functions in the server are atomic so the four functions that support multiple jobs are as follows:

```
LOAD <file>
SAVE <file>
GETBYTES - filehandle, x bytes
PUTBYTES - filehandle, x bytes
```

When the system decides that an operation will take longer than the time slot available for an atomic operation, it will push the process state on to a stack and set a flag to indicate the pending process. This is determined by the amount of time it takes to transmit 4k of data across the network. A WIMP POLL is then performed followed by a check of any pending messages in the receive blocks. The process queue is then checked and the next process due for a time slot allocation is called. Before the process is called its entry in the queue is removed.

#### **4.6 Multitask switch**

Due to the nature of the RISC OS system it will be beneficial to allow the Level 4 system to operate in a mode where it is the only application running in the machine. This will enable the server to gain the

maximum performance benefits without being interrupted by other tasks. It should also be noted that while running under the desktop it is possible to stop the server software by simply pressing the f12 key, this is not acceptable for a file server system and therefore a mode of operation will be provided that prevents this. This also means the Level 4 system can be installed and used in a similar way to the previous systems (ie: box with no keyboard or monitor).

Two versions of the system will be provided on the release disc:

a) RISC OS desktop applications

!Server

!Spooler

!Manager

b) RISC OS stand alone code

Level4 (comprising of server, spooler and management interface)

## 4.7 Management tools

It is possible to control various functions of the server from a client machine or on the machine running the file service. In order to use these functions it is necessary to be authorised as a system user. Some of the functions are supported via a command line interface, however certain operations are only possible with a utility program (eg: setting file server time/date). All management functions can be achieved with the Management Service application (see section 6).

### 4.7.1 Commands/functions supported:

display users in password file

display current status of user - logged on etc.

change password of user

\*pass <old password> <newpassword> [<user name>]

list/display file servers (see section 6).

display/set/change file server date

Free space on server

\*free [<disc name>] [<user name>]

create new user

\*newuser <user name>

remove user a user from the password file

\*remuser <user name>

set user privilege

\*priv <user name> [{F|L|S}]



## 4.8 Objects per user (OPEN)

Level 4 will allow RISC OS clients to have up to 255 objects open at any one time. This will include 3 for URD, CSD and LIB. The BBC machine running NFS 3.6 will be allowed 8 (including 3 user handles), while Master series machines (ANFS 4.25) will support 16 handles. When the client logs on, the server will peek the client machine to determine the machine type. This will automatically define the maximum number of open objects allowed. It is possible to define a lower number when setting up the user profile using the management tools.

## 4.9 File systems supported

File system configuration will be via a text file. This will specify the devices connected to the server which are exported as 'discs'. The order they appear in the file will determine how they appear to other users. It is possible to export any filing system which supports the filing system interface standard ie: ADFS, SCSI, RAMFS, NET, CDFS, etc.

The format is as follows:

```
<filing system> <SWI Prefix> <drive number> <mount point>
```

up to the maximum number of file systems.

An example file is

```
ADFS ADFS 0 $
ADFS ADFS 4 $
SCSI SCSIIFS 4 $
NEXUS Nexus 4 $
RAM RamFS 0 $ -
CDFS CDFS 0 $
NET NetFS net:Logon:winnie owner
NFS NFS nfs:Logon -host unix guest
NFS NFS nfs:Mount -host unix etc/etc
```

## 4.10 Record locking

Access by multiple readers and writers will be allowed in a special mode. This means several clients may update a file simultaneously. A lock server may be provided in a later release of Level 4 to provide true record locking facilities.

## 4.11 User types

Different levels of access permission are allowed in order to control the security features of the system. Each type of user has the ability to execute certain functions within the server (see section 2.3).

There are currently four types of user, these are as follows:

System

Normal

Locked

Fixed

#### **4.11.1 System**

Full access to all file systems and areas. Ability to view/change password file and maintain accounting details.

#### **4.11.2 Normal**

Read/write/create access to URD and sub-directories and others assigned to user.

#### **4.11.3 Locked**

Like a normal user except not able to change password or boot option

#### **4.11.4 Fixed**

This user may only access/see the directory and sub-directories that the URD is set to. This means the ROOT directory may not be viewed. The password may not be changed by the user only the system manager. The boot option may only be changed by the system manager. Space accounts operate as normal. The URD of this user appears as root: \$.

### **4.12 Hidden files and directories**

Objects may be 'hidden' from unauthorised users. This provides an added level of protection. A switch is provided from the file service management interface which allows this feature to be turned on or off. When the hidden files option is active only the objects which the user 'owns' or has public read or write permission can be seen. This means a directory viewer will only display the objects the user can access. Directories are locked in order to hide them. Objects within a locked directory that have public access may be accessed if the clients specify the correct name. Unlocked directories may be viewed by any user.

### **4.13 Ownership system**

#### **4.13.1 General**

A user with normal status will have full ownership of their URD. This allows them to create/ examine/ read/ write /modify/ delete objects within the URD. Ownership is not lost by moving from this directory with the \*DIR command. The user also automatically owns any sub-directories within the URD. It is also possible to define an environment which will allow users to own more than the URD. Any other objects will have to have public permissions defined in order to allow access ie: /wr to allow reading and writing. If the hidden

object option is selected, the user will only be able to perform an operation (examine, load, save etc.) if the public permissions are set. Any attempt to access an object when the user is not authorised will result in an error such as: 'File not found'

#### **4.13.2 Ownership of more than URD**

The server will provide the means to allow the system manager to allow certain users to 'own' more than their root directory. This will allow group accounts to be set up allowing a number of users to share a directory and be able to create, read and write objects within it. The space accounts will be kept on a per user basis. Space is allocated to each user and removed and deleted according to the amount of space they use on the discs. The group directory has a user identifier allocated with a space allocation but no ability to log on, any space used in this directory will be accounted for by the group user id rather than the user. Group directories are limited to being in the root directory of the device used.

#### **4.13.3 Configuration of access**

It is possible to configure the level of access the user has to a group directory. The following options are available:

- Examine directory

- Read objects

- Write objects

- Creation of objects

- Deletion of objects

This will be set up in the user profile by the system manager.

#### **4.14 System Internal Number (SIN)**

This number is used to uniquely identify an object stored on the server. Its primary function is to enable the broadcast loading system to operate. It is accessed by using functions 3 or 18 with the appropriate arguments.

The SIN is derived as follows:

Bits 0-23	Filing system SIN
Bits 24-31	Server disc number
Bits 32-47	Filing system number

#### **4.15 User Root Directory (URD)**

This is the directory that the client is allocated when authorisation is successful. The user may own this directory and may create objects within it. The name of this directory may be different from the username.

A space allocation is associated with this directory and additions or deletions are added and subtracted from free space allocation. This is the directory that the user is returned to when the command \*DIR <return> is executed.

It is defined by the system manager when the user is created and can only be changed by the manager. It is set up in the user profile and if changed while the user is logged on will not take effect until the user logs in for the next session.

The URD takes the form:

```
adfs::harddisc4.$fred
scsi::4.$jim.bill
ram::$henry
```

If no URD is specified it will default to the username. The sequence of actions for locating the URD is as follows:

When the user logs on, the specified URD is looked for. If not found then the file systems the server is exporting are searched in sequence for a directory which matches the username. If no directory is found then the user is given public access to the ROOT directory of the first file system in the server file system list.

## 4.16 Library (LIB)

The library is the directory that is searched if the object requested is not found in the CSD. Usually system functions such as \*users are provided as transient utilities stored on the server. The location of these utilities depends on the machine type using them. The client will be allocated \$.Library as their library unless the client machine specifies otherwise. ie: A RISC OS system can be configured to select an alternative to \$.Library when log on takes place, this prevents confusion between 6502 based utilities and ARM code versions. If no \$.Library is found then '\$' is allocated as LIB.

## 4.17 Currently Selected Directory (CSD)

This is the directory the user is currently in. Any load/save/create operations that do not specify a pathname will be performed on this directory. A handle is allocated and when the client moves from this directory by using the \*DIR command the old handle is deleted and a new one allocated. It is possible to use these handles to control the way the user sees the server ie: The LIB handle can be made the CSD handle therefore \*CAT would display the Library directory. If bad handles are passed to the server the error 'Channel' will be generated.

## 4.18 Passwords

Passwords are the secondary level of protection against unauthorised access to objects and to features of the server. Passwords are stored in the password file. They are in encrypted format to prevent anyone who 'dumps' the file to find out what they are. When a user logs on the file is searched for a user identifier match and if found the password is compared (if one exists). If the password does not match the user is not logged

on and receives the error “Wrong password”.

The maximum length of a password is 22 characters. If a password is used it must be at least 6 characters long.

#### **4.19 Maximum number of users**

The maximum number of users can be defined by the system manager. This can be set to a maximum of 128.

#### **4.20 Memory management**

The server employs dynamic memory management for allocation of workspace for users and other internal tables. This ensures the system attempts to minimise the amount of memory used for the current session.

#### **4.21 RISC OS Desktop interface**

When the server is running on the RISC OS desktop the interface will be as follows:

##### **Server**

Info > info dialogue box

Log file > dialogue box to allow setting of log file options (As specified in Section 4.1.1 )

Set up > configuration of global server options (Hidden objects, Allow logon, Open Status Window on start up, Space accounting enable)

Save choices (save all options specified)

Quit

#### **4.22 Standard system timeout**

When a client logs on a timer is started. After a specified time of inactivity the client will be automatically logged off. The client can be notified before log off occurs. Time is specified in minutes and is set up on the desktop server configuration.

##### **4.22.1 Connect time**

Each user may be allocated a connect time. This will allow certain users to have short connect time allocations while others may have no automatic logoff set.

#### **4.23 Server shutdown**

The server has to be shutdown cleanly to avoid files being left open and potentially unsaved or incomplete

transactions. The server has two modes of shutdown; fast and normal.

#### **4.23.1 Fast shutdown**

In fast mode the active processes are closed as quickly as possible and all users are automatically logged off.

#### **4.23.2 Normal shutdown**

Normal shutdown will first send a message to all logged on clients notifying them of the pending shutdown. The message will say how long it will be before the service is halted. A timer is started and before the actual shutdown occurs a final message is sent. User's files are then closed and they are automatically logged off. The server application will continue to run but return the error 'Server not available' when any requests for service are received. When quit is selected from the server main menu then all receive blocks and claimed memory is released. This function may be accessed by a system user as \*FSShutdown [<time>]. If no time is specified then fast shutdown is carried out.

#### **4.24 Caching of files**

No caching is performed by the server code in the current implementation. It is expected that the Broadcast Loader module may go some way in overcoming some of the bottleneck problems, but such an enhancement is not ruled out for the future. It is desirable to set ADFSBuffers and ADFSDirCache to large values to increase the potential performance of the Level 4 System when using ADFS.

#### **4.25 Error messages**

The server responds with errors under certain circumstances. The errors generated are as follows:

<b>Error string</b>	<b>Error Number</b>
Insufficient space	&5C
Too much data	&83
Sorry, not supported	&85 The command/function is not supported
Bad privilege letter	&8C
Bad rename	&B0
Already a user	&B1
Directory full	&B3
Is a directory	&B5
Too many users	&B8
Insufficient privilege	&BA
Wrong password	&BB
User not known	&BC
Access violation	&BD
Insufficient access	&BD
Is a file	&BD
Who are you?	&BF
Too many open files	&C0
Already open	&C2
Disc full	&C6
Bad file name	&CC
Bad drive	&CD
Invalid access string	&CF
Not found	&D6
File not found	&D6
Channel	&DE
Bad command	&FE
<b>Server not available</b>	<b>&amp;FF</b> The server has been shutdown
<b>Server Internal error, please report to system manager</b>	

Errors in bold are new Level 4 errors.

## **5. Print service - software features**

### **5.1 General**

This module allows up to eight printers to be shared via the network. It provides a multiple client spooling system to enable many clients to send data for printing simultaneously. If the printer is busy the data is stored on a suitable medium such as hard disc until the printer is free. A queue system is employed to manage the order in which the jobs are submitted to the printer. This will simply send the data to the printer as soon as it is free. It is based on a first come first served basis.

Internally the spooler runs three main tasks. The first responds to poll, status and queue information requests. The second deals with receiving data and filing it on disc. The third is responsible for sending data to the printer when it is free.

The spooler has the ability to allow several physical printers to be attached thus allowing a serial printer and a parallel printer to be connected to the same machine. Each printer supported may also offer a variety of styles each being defined by a particular name and associated header and footer files. A maximum of eight printers is allowed. Postscript printers are dealt with as a special case in order to achieve the maximum efficiency when using such printers.

### **5.2 Print service - setup/configuration**

Various options are available to define the environment in which the print service operates. The options that can be selected are as follows:

- Printer name - up to a maximum of 6 characters

- PostScript printers - see section 5.3

- Open spooler queue status window when spooler starts up

- Start printing before all data for print job has been received

- Timeout period for aborted jobs

### **5.3 PostScript printers**

If a PostScript printer is connected a number of advantages can be gained when using with applications that do not generate PostScript (see section 5.9).

If data is PostScript ie: identified by %! as the first two characters of the data, then no header is sent before the data. If data is not PostScript then a PostScript header is first sent to the printer to interpret the data that follows. By using this method a significant improvement in performance is achieved due to the fact that less data is being sent to the printer.

### **5.4 Printing before received all data**

If this option is selected the spooler will start sending data to the printer whenever it is free. This means the entire print job does not have to be sent before the data is transmitted to the printer. This is especially useful



if large amounts of data are involved. The option may be switched on and off via the spooler control panel.

## **5.5 Header file**

The header is a file which is sent to the printer before the main data is sent. This can be used to initialize the printer or in the case of the PostScript printer send a program to interpret non-PostScript data.

This file may be edited with a text editor.

## **5.6 Footer file**

This is similar to the header file except it is sent after the main data has been sent. A common example is to send a form feed code to line the paper up at the next page break.

## **5.7 Multiple physical streams**

The spooler can be configured to output data to one or more physical devices. Common devices include the parallel, serial and network. One or more of these streams can be active and jobs sent to particular ports. This allows the spooler to support a serial and a parallel printer at the same time. A file containing a list of the printer types is used to define which devices to use for each printer name. The format is as follows:

```
<name> <output stream> <line feed>
```

```
Spool Serial:
```

```
Matrix printer:
```

## **5.8 Multiple logical streams**

It is possible to define the characteristics of different printers that share the same physical device. This may be to provide form feed on one and not the other. The name of the printer is used to determine which port and logical printer type to use. The directory for each printer will contain a sub-directory for spooled files, a footer and a header file for that particular printer.

## **5.9 Support of applications that cannot generate PostScript**

If applications such as VIEW and 1st Word Plus are being used in conjunction with a PostScript printer, they will not be able to print anything because they do not generate PostScript or use the RISC OS printer drivers. The spooler supports a header file which is a PostScript program designed to run inside the printer. This program accepts and interprets codes sent to it from the client running the application. A special printer driver is required for VIEW or 1st Word plus in order to use bold, underline etc. on a PostScript printer. The ability to print normal ASCII text is also supported.

## **5.10 Desktop interface**

The spooler interface consists of a main menu, options page and a window display of queue information and status.

### 5.10.1 Main menu

The main menu is as follows:

#### **Spooler**

Info -> info box

Save options

Quit

The info box is standard. Selecting the Spooler displays the list of printers available, with their icons.

The menu on each printer gives:

Display => (Icon size, full info)

Printer spool => (Rename, delete, parallel, serial=>, network=>, file=>, save choices, stop printing)

Options => (PostScript printer, Open status window on startup, Start printing while spooling, job banners)

The 'save options' save the current configuration.

### 5.10.2 Queue information

This window displays the current name of the spooler and its station number. Options to allow the printer to be on or off and the queue to be suspended are provided. If the printer option is off then no data will be transmitted to the printer however spooling will continue to operate as normal. The suspend queue option will stop any jobs being submitted for printing but allow the current job to complete and spooling to operate as normal.

The rest of the display concerns details about currently active print jobs. The information displayed is as follows:

Station, Time, Date, Size, Status.

A percentage of how much data has been printed is also given for the currently printing job.

Each job can be individually deleted, suspended or continued.

## 5.11 Timeouts

If the client stops transmitting print data without completing the print job then after a specified time period the job will be aborted and the data already sent deleted. This time can be configured on the printer service set up panel.

## **5.12 PostScript printer messages**

A separate application is provided that will display the messages returned by a PostScript printer. This will also allow the serial port to be set up.

## **5.13 Banners**

Banners can be enabled via the option menu (see above) for each printer. They can be configured to enable a job banner and/or a page banner.

## 6. Management service

### 6.1 General

This is a RISC OS desktop based application which multitasks and allows the system manager to access the management functions associated with the service. This includes the creation and removal of users, allocation of space and management of security and access control. The aim is to make the interface as simple as possible and to provide a consistent intuitive interaction between the manager and the system. To this end the RISC OS style guidelines are followed. The application can run on the machine running the service or from a remote station. All the management functions available on the file server will be accessible through the management service interface.

The application provides three hierarchical levels to the system manager corresponding to selection of fileserver, selection of user, and selection of user attribute.

Note that the application can only be used to control Level 4 Fileservers. An appropriate error message will be returned if an attempt is made to control a non-Level 4 fileserver.

For details of the look and feel of this application, refer to the Acorn RISC OS Network Managers Guide.

### 6.2 Fileserver Selection

When the Manager application has been loaded, menuing on the icon gives the options:

Info => Program information box

Quit

Clicking Select on the Manager icon initiates a display list of available fileservers. This is recalled each time it is recalled.

This display of file servers, similar to the net filer display, allows the user to select which file server they want to manage. This will effectively show an overall view of the network at the 'top' level.

Clicking menu in the fileserver window presents the following box:

Display =>

Logon 'Fred' =>

Bye

Shutdown

'Display' opens a standard filer display for icon size / full information and sorting options. Note that the "full information" option displays additional information about fileserver type and software issue numbers.

The 'Logon' dialogue box presents the name of the fileserver currently pointed to, and is identical to the

standard logon box. It is necessary, of course, to log onto the selected fileserver as a privileged user if the management tools are to be used.

'Bye' logs off the selected server.

'Shutdown' logs all users off the system and makes it unavailable. It can be restarted by reloading the application.

Double clicking on the selected fileserver will, if logged on, enter the second menu level.

### **6.3 Display list of users in password file**

A 'filer' type display of the users on the file server will be given. This will display user icons representing each user in the password file. Clicking menu inside this display gives the following options:

Display =>

User "Fred" =>

Select All

Clear Selection

Options

New User =>

The display options are large icons, small icons and full information. The icons represent the different classes of user; System, Normal, Fixed and Locked. Full info gives their station number and type, and their logged on status. There is also a sort by name, number or privilege option.

The 'User' path allows the selected user profile to be copied, renamed, or deleted, and notified or logged off.

The 'New User' option allows a new user profile to be generated with the default parameters.

By double clicking on the user icon the details or profile for that user can be accessed thereby allowing the manager to view or modify the state of a user.

It is possible to drag a suitable text file into the user list filer window to create a ready made number of users.

## 6.4 User profile setup

The user profile is a set of data that identifies the user and its environment. All the attributes associated with the user can be set up, viewed and modified by the system manager using the management tool. This profile is displayed as a window with text insert areas, buttons and drag bars for quick visual inspection and fast update. The following user attributes can be defined:

username - up to 10 characters with 10 for group name.

password - up to 22 characters, minimum of 6. (Option to display as text or ----))

user root directory (URD) - pathname of logon directory

boot option on logon - none, load, run or exec

privilege - system, normal, locked or fixed

logon disable/enable flag - ability to log on

current directory (status info) - CSD

current library (status info) - LIB

mail system directory - create objects for mail system

space allocation (Drag bar)

logged on status, and time when last logged on.

If the profile is updated by the system manager, then a create button updates the user profile accordingly.

## **7. Manual design and contents**

The manual for the product is titled "Level 4 Fileserver Network Manager's Guide". It will be based on the similarly titled document for the Filestore product.

It will be in three major sections: Part 1 details suitable hardware that can be used as a platform for the Fileserver and network products to support it and the various options that are available in terms of Filing system. Part 2 outlines the basics of filing structures and the role of the network manager. Part 3 details management tasks necessary for the day to day running of the system. It concentrates on the Remote management tools provided with the server, for both fileserver and printer spooler functions.

What follows is some guidelines to help the formulation of the manual. It uses the Filestore guide as a starting point.

In general, this manual is designed for Archimedes clients, rather than BBC micros. To this end, the emphasis should be placed on the desktop environment. For those users with BBCs, they are recommended to refer to the Filestore documentation, with which this product is backwards compatible.

### **7.1 Warnings**

As the L4FS is a software product only, there is no requirement for hazard warnings and mechanical handling of the product. However, the "continuous improvement" disclaimer, correspondence addresses and trade marks remain.

### **7.2 Introduction**

As per Filestore but use up to date examples. "About this guide" shows the structure of three parts as mentioned above.

Applicability to versions is not relevant.

### **7.3 Part I**

This section deals with the platforms that can support L4FS. The main point being that as a RISC OS application it can run on any Archimedes machine with 1MByte of memory, networking and suitable backing store.

Two versions of the software are available (both on the same disc), the first being the standalone code "server" and the second the desktop version that comes as !Server, !Spooler and !Manager.

When running in standalone mode, it is possible, although not necessarily recommended, that the machine can be run without keyboard or monitor.

The section also covers the variety of filing systems that can be supported by the server. These are:

ADFS - Standard floppy and Winchester disc storage

SCSIFS - For larger capacity drives and for use with A3000 machines

NFS - For connection to a NFS server across either Ethernet or Econet

CDROMFS - For connection via SCSI to CD ROM drives

and any other existing or future compatible filing system.

## **7.4 Part II Managing L4FS**

Background information. Explanation of computer terms can remain as appendix.

It would be useful to describe briefly the server as it appears to clients; that it is seen on the icon bar as a server and can be used in the same way as any other filing medium. The additional facilities, such as additional filing systems are accessible via server menus in the normal fashion as disc names.

- i) Discs. Needs to be expanded to all media types. A summary of capacities and advantages useful.
- ii) Filing structure, moving between directories. Can remain intact.
- iii) Logging on. As it stands. Reference to Archimedes user guide for logging on procedure.
- iv) Network Manager's role. Intro, Security OK. Passwords should mention minimum password size.
- v) Restricted Access. Should mention desktop means of changing attributes. Should also mention Privilege, User Types as described in the relevant sections of the FS. Physical access to machine, data loss, backing up to be referred to.
- vi) Adding, removing users - see later section (management tools)
- vii) Adding new stations, see later section. (making changes)
- viii) Utilities Discs. This section no longer relevant. Refer to the three modules in the level 4 system: File Service, Print Service and Management Service. An outline of the software structure could be useful as in Sect 1 of FSpec. Software will be supplied on Floppy with two versions on it: Standalone and desktop.
- ix) Hard discs. Need to mention their existence in relevant platforms. Otherwise no info required.
- x) Managing discs. Need to stress that operation without hard disc is unlikely! Perhaps this section needs rewriting, stressing use of floppy as backup medium only.
- xi) Managing space. Section can stay. However, needs to refer to later section on management tools that describes space accounting.
- xii) Adding new files. Use standard desktop facilities. If in stand alone mode then can be done remotely.
- xiii) File availability. As filestore.



xiv) Keeping records. Needs modifying as printer server and fileserver the same platform(s).

xv) NETMGR. This utility is now replaced by the remote management tools.

xvi) Maintenance tasks. As per filestore. Adding hard discs or adding other filing systems no longer a normal task. No maintenance mode is provided.

xvii) Viewdata system. Delete reference to this.

## **7.5 Part III Management Tasks**

Section 6 of FSpec contains useful info on general management. In particular, a description of the RISC OS application !manager needs to be described.

This includes a) display list of file servers, b) display list of users in password file, and allows manipulation of the password file. Also, it sets the user profile set up information. (username, password, URD, boot option, max no. of open files default save attributes, privilege, logon enable/disable, delete/create objects, connect time allocation, CSD info, LIB info, mail system directory, space allocation and logged on status).

Other information similar to filestore:

i) Starting up. Depending on whether the stand alone or desktop version is required, and/or has been added to the system boot file run commands, the machine will either wake up on power-up as a full server, as a desktop server, or the applications !server !spooler and !manager will have to be found and clicked on to initiate the service.

ii) Shutting down. There are two shutdowns that can be used: Fast Shutdown and Normal Shutdown. See description in FSpec Section 4.23.

iii) Formatting discs. Part of normal desktop operation.

iv) Adding/Deleting a user. Via management tools.

v) Changing a disc. Shouldn't need this section.

vi) Changing CSD. Shouldn't need to describe \*DIR function - use desktop.

vii) Wiping / Moving / Copying files. Use desktop.

Safeguarding data.

The introduction to this section should describe the various levels of privilege that are available: System, Normal, Locked and Fixed. (See FSpec section 4.11 - 13) It should also describe Hidden files and directories, ownership of files. Describe the file attributes WR/wr etc.

*NB: Sheets 57 to 59 are missing from the original document scan.*