c-tree® Server Administrator's Guide

for

Version 7.12

Published by

FairCom Corporation
2100 Forum Blvd., Suite C
Columbia, MO 65203
USA
Telephone: (573) 445-6833

Fax: (573) 445-9698

www.faircom.com

Sixth Edition, First Printing: December 2002

V7.12(0922)(030203)

Dedication

This guide is dedicated to the employees of FairCom in recognition of their dedication and the teamwork required to bring the c-tree Server to fruition.

WLF

Table of Contents

1. c-tree Server Administrator's Guide

1.1 Server Quick Start

The **c-tree Server** is designed for ease of use with minimal administration. This is different from other database server products, which require extensive setup, administration, and support.

To make the **c-tree Server** operational, simply:

1. Install,
2. Activate, and
3. Execute the Server.

Install

Follow the installation instructions on the **c-tree Server CD**.

Activate

Execute *fcactvat* to activate the Server executable file (this occurs automatically during the Windows install) and follow the prompts to activate the Server. The **c-tree Server** activation process stamps the Server executable for the number of concurrent connections specified by the Server license purchased.

Execute

Execute the *ctsrvr* executable (*ctsrvr.exe* on Windows, *ctsrvr.ppc* or *ctsrvr.68k* on Mac). This runs the Server with the default settings.

That’s all it takes to get your Server up and running.

While the **c-tree Server** runs properly right out of the box, the rest of this guide details the installation, operation, and optional configuration settings available to the Server Administrator. FairCom recommends using the available security options, establishing regular data backup procedures, and optimizing the Server configuration for your environment to maximize performance.

See the next section, "Introduction" for an overview of the content of the Guide.

1.2 Introduction

This manual has two main purposes for the **c-tree Server** Administrator:

1. To provide a quick, easy way to see what responsibilities you have, and
2. To provide the information needed to manage **c-tree Server** operation.

The **c-tree Server** supports high-level database management, including:

- **client/server computing** - increases performance and provides the ability to maintain database integrity, especially in multi-user environments. The basic principle of client/server computing is: applications, or "clients", interact with the server, which manages file operations and communicates with clients.

- **online transaction processing (OLTP)** - the **c-tree Server** can group a specified set of operations, called a "transaction," and ensure either all of them are done or, if there is a problem, none will be done, e.g., either all of an invoice is processed, or none of it.
- **security controls** - **c-tree Server** access is controlled with user IDs, passwords, file permissions, and encryption. Users and files may be added to Administrator defined "groups", e.g., shipping department, payroll department.
- **database maintenance and utilities** - the **c-tree Server** automatically saves necessary information for use in automatic, or Administrator-specified, backups and for recovery from problems.
- **configuration flexibility** - from basics such as which communication protocol the **c-tree Server** uses and memory allocations to enforce for specific users, to a wide range of advanced controls.

Further technical details concerning the **c-tree Server** are available in Appendix B of this Guide and other FairCom documents.

The **c-tree Server** Administrator has the following six areas of responsibility, each of which could be divided among several people:

Installation

Someone, not necessarily the Administrator, must physically load the **c-tree Server** software onto the computing environment. Once completed, installation issues usually are no longer a concern unless the **c-tree Server** needs to be reinstalled, e.g., to install a new version. See Chapter 2, "c-tree Server Installation".

Operating the Server

Starting and stopping the **c-tree Server**. Any user can start the **c-tree Server** by running the executable module, *ctsrvr*, as any other program in the environment. See the "Operating the c-tree Server" chapter of this guide for details.

Controlling access to the c-tree Server

Begin by setting up valid User IDs and passwords (including your own!). Establish rules of access to given database files. Establish groups where users and files can be associated and control access according to membership in those groups.

Use *ctadmn*, the **c-tree Server** Administration Utility, to control access with user IDs, file passwords, file permissions, and Administrator defined groups with specified access rights to particular files. This utility also monitors user status and/or disconnects users from the **c-tree Server**.

ctpass is used by the Administrator or any other authorized user to change the password associated with their User ID.

ctfile is used by the Administrator or any user to change file security information on any file owned by the user. See the "Controlling c-tree Server Access" chapter of this guide for further details.

Maintaining Database Integrity

Schedule and conduct backups or dumps of system generated files for later use in recovering from problems or returning a database to its status at a prior time.

Use the utility *ctdump* to schedule Dynamic Dumps that can be used at a later time to restore database files or to roll back to a state at a previous point in time.

ctrdmp works with information saved in a Dynamic Dump to either recover from a catastrophic system failure by restoring specified files to a consistent, well-defined state or to roll back specified files to their state at a specified time.

Use the utility *ctfdmp* to recover from a catastrophic failure using a previously saved dynamic dump or complete backup, which may be made using any standard backup utility. This allows you to restore backups then 'roll forward' to a given time using preserved log files.

(For programmers) Use the utility *ctldmp* to carry out a transaction log dump, which records partial log related information, for use in application development. See the "Maintaining Database Integrity" chapter of this guide for further details.

Configuring the c-tree Server

Understand how the **c-tree Server** is currently configured and, optionally, change configuration settings (e.g., to set memory allocation limits, to select communication protocols, to activate a particular dump description script).

The **c-tree Server** is started by any user authorized to start *ctsrvr*. Routine starting of the **c-tree Server** is not necessarily a major responsibility for the Administrator.

The User ID "`ADMIN`" (default password is "`ADMIN`") and members of the `ADMIN` group are the only users who can access *ctstop*, the utility for stopping the **c-tree Server**, so stopping the **c-tree Server** is always a major Administrator responsibility.

Customize the c-tree Server

No configuration file is required, but if the **c-tree Server** is to be reconfigured to replace any default settings, a file named *ctsrvr.cfg* must be created for the Server to load at startup. See the "Configuring the c-tree Server" chapter of this guide for details.

NOTE: Utility names and methods of executing them may vary slightly in different environments, so see Chapter 2 for specifics. The utilities covered here are not the only ways to carry out Administrator duties and the utilities listed here are not necessarily the only ones available.

The basic topics covered here are for orientation only. Chapters 2 and 3, "Installation" and "Operating the c-tree Server", are considered required reading for **c-tree Server** Administrators. Chapters 4 and 5 and sections 6.1 through 6.3 are recommended reading. Sections 6.4 through 6.7 are optional and intended for advanced users.

Some issues may require the assistance of others with specialized knowledge relevant to the operating environment (e.g., configuring memory access allotments, defining dynamic dumps).

Additional information can be found in the following appendices:

- Appendix A: "User's Control of Security Options" contains user password control information.
- Appendix B: "Overview of The c-tree Server" contains a conceptual overview of the **c-tree Server**.
- Appendix C: "Glossary" defines terms.

2. c-tree Server Installation

Installing the **c-tree Server** is mostly a matter of copying software from the distribution CD onto the system. **c-tree Server** installation can be completed in three simple steps:

1. Install the **c-tree Server** and support utilities.
2. Activate the **c-tree Server** using the *fcactvat* program. **NOTE:** Some **c-tree Server** OEM vendors may provide pre-activated **c-tree Servers**.
3. Start the **c-tree Server**.

The following sections provide the information necessary for installing the **c-tree Server** on specific environments. Before proceeding, verify that the computer on which the **c-tree Server** is to be installed has sufficient capacity for the **c-tree Server** and the associated applications.

The "Minimum Hardware Requirements" sections discuss the minimum memory (RAM) requirements of the **c-tree Server** not including operating system memory requirements. Additional RAM for file caching, opening files, supporting many users, etc., is encouraged for optimal performance and functionality. Section 6.6 of this Guide provides formulas for approximating **c-tree Server** memory requirements.

The hard drive space specifications contained in the following sections indicate the minimum space necessary to install the **c-tree Server** on each particular operating system or platform.

Skip to the appropriate operating system section where your **c-tree Server** is to be installed.

2.1 c-tree Server for Windows

Operational Environment

The c-tree Server for Windows 95/98/NT/2000/XP uses the Windows multi-threading functions and is named *ctsrvr.exe*. The c-tree Servers for Windows are distributed on a CD containing *Setup.exe*, which installs the product, including the following Windows communication DLL's:

Protocol	DLL File Name
Shared Memory	`FSHAREMM`
NetBIOS	`FNETBIOS`
TCP/IP	`F_TCPIP`
TCP/IP (Encrypted)	`FETCPIP`

NOTE: Windows 95/98 supports only NetBIOS and TCP/IP and therefore does not include the *FSHAREMM.DLL*.

The c-tree Server for Windows defaults to the TCP/IP protocol. To activate any other DLL, use the `COMM_PROTOCOL` keyword in a *ctsrvr.cfg* file (discussed in the

"Advanced Configuration Options" section of this Guide). Use the DLL name as the token following the `COMM_PROTOCOL` keyword in the ctsrvr.cfg. This disables the default protocol, so if you want to load the default and another protocol, each must have a `COMM_PROTOCOL` entry in *ctsrvr.cfg*. For example, to load all supported communication protocols for the c-tree Server on Windows NT, add the following lines to *ctsrvr.cfg*:

```
COMM_PROTOCOL  F_TCPIP
COMM_PROTOCOL  FNETBIOS
COMM_PROTOCOL  FSHAREMM
COMM_PROTOCOL  FETCPIP
```

NOTE: If `COMM_PROTOCOL` is specified for one protocol, all protocols to be used must be specified. If no `COMM_PROTOCOL` is specified, the **c-tree Server** uses the default, `F_TCPIP`.

Automatic LANA Support

The c-tree Server for Windows offers automatic LANA support for the NetBIOS protocol, allowing the Server to connect with clients on any LANA (specific NetBIOS adapter). When `COMM_PROTOCOL NETBIOS` is listed in the Server configuration file, the **c-tree Server** listens to all LANA ports. To specify a specific LANA number from 0-9, use the following format:

```
COMM_PROTOCOL FNETBIOS@1
```

This disables automatic listen and sets a specific LANA number, LANA 1 in this example. LANA 0 is the default. Specifying a desired LANA number from a client side application uses the following format:

```
FAIRCOMS@1^NETBIOS
```

Minimum Hardware Requirements

The minimum CPU and memory requirements for operating the c-tree Server for Windows are: a Pentium 133 CPU, and the memory required by Windows 95/98/NT/2000/XP plus 2MB RAM. The minimum CPU and memory requirements for operating the c-tree Server for Windows XP 64-bit Edition are: an Itanium CPU, and the memory required by Windows XP 64-bit Edition plus 3MB RAM.

NOTE: Windows limits application memory space to 2GB.

The minimum hard drive space required by the c-tree Server for Windows is:

The size of the **c-tree Server** executable,
+ the size of specified communication DLLs,
+ the amount specified by the `LOG_SPACE` keyword (10 MB default),
+ 1MB for **c-tree Server** status logs,
+ 3.5MB for the pre-compiled **c-tree Server** utilities,
+ the size of the data (.dat) and index (.idx) files.

The Windows NT, 2000, and XP operating systems using NTFS support files up to 4 GB using Standard **c-tree Plus** files, and will support Extended files larger than 4 GB.

Windows 95 and above using FAT32 support Standard files up to 4 GB and require segmented files to support huge files. All other versions of Windows will support 2GB file sizes and require segmented files to support huge files.

c-tree Server for Windows Installation

1. Insert the **c-tree Server** CD into the proper drive.
2. The installation Setup utility starts automatically. Follow the instructions to install the **c-tree Server**. If Setup does not start automatically, execute it from the CD.
3. The installation Setup utility should automatically attempt to activate the **c-tree Server** using the *fcactvat* program. See the **c-tree Server** Activation Key flier for instructions. Some **c-tree Server** OEM vendors may provide pre-activated **c-tree Servers** with their applications.

During installation, communication DLL files will be placed in the **c-tree Server** directory. These c-tree communication DLL's can be accessed by leaving them in the same directory as the **c-tree Server** executable or by placing them in a directory referenced by the PATH environment variable.

Installing the c-tree Server Service

During installation you can install the Server as a service. If this option is selected, the installation process will automatically set up the service with default settings.

To manually install the **c-tree Server** as a Windows NT/2000/XP Service after installation, run *ctntinst.exe*, specifying the `-install` option. You may optionally specify the service name. For example:

```
c:\faircom\bin.svc> ctntinst -install
c-treeServer Created
```

See the "c-tree Server Service" section for more information on operating the **c-tree Server** as a Service.

Installing Multiple Instances of the c-tree Server

To run multiple instances of the **c-tree Server** on one machine, each instance must be individually licensed, have a unique name, and have a unique set of server and data files. Multiple **c-tree Servers** cannot share files. For each Server, you must:

1. Install each instance of the **c-tree Server** into a unique directory.
2. Activate each license as described in the "c-tree Server Activation Key" flier.
3. Specify a unique Server Name in the Server configuration file, *ctsrvr.cfg*, as described in Chapter 6 of the **c-tree Server Administrator's Guide**.
4. Configure each instance to have a unique Service Name as described in the "Configuring the c-tree Server Service" section.
5. Every instance of the **c-tree Server** requires a separate license.

See the "Running Multiple c-tree Servers on One Machine" section for more information.

Tool Tray interface

When the Server configuration file contains the `CONSOLE TOOL_TRAY` keyword, the **c-tree Server** starts in background, displaying only a c-tree icon in the Windows tool tray. This feature is especially nice for 'simple' user sites, with no system administrative expert. Although more sophisticated sites will prefer running the **c-tree Server** as a service, this feature gives a similar 'service-like' background effect, without the user needing to learn Windows service administration.

Add the following keyword to your server configuration file, *ctsrvr.cfg*:

```
CONSOLE   TOOL_TRAY
```

This keyword is not supported when the Server is running as a Service.

The c-tree Servers for Windows accept the "&" symbol, ("^&" for Windows NT/2000/XP), as a command line parameter to execute in `CONSOLE TOOLTRAY` mode. The following example launches the Server in "background-tool-tray" mode:

```
C:\server>  ctsrvr &
```

Windows 95/98 Service Support

It is possible to execute the c-tree Server as a Windows 95/98 service. Not to be confused with a Windows NT/2000/XP Service, this support is limited to the Windows 95/98 platform. This feature allows the Server to remain in operation even if a user logs off of Windows without shutting down. Add the following keyword to the Server configuration file to activate this support:

```
CONSOLE W9X_SERVICE
```

This keyword will be ignored on all other platforms except Windows 95/98.

2.2 c-tree Server for Novell

Operational Environment

The **c-tree Server** for Novell is designed to operate as a NetWare Loadable Module (NLM) on a Novell file server machine. The **c-tree Server** supports Novell versions 3.X, 4.X, 5.X, and 6.X and communicates with IPX/SPX or TCP/IP clients.

Minimum Hardware Requirements

The **c-tree Server** for Novell is very efficient and operates well on Pentium 133 and later processors. The minimum RAM requirement for the c-tree Server for Novell is the RAM required by the Netware system plus 4MB RAM. 8MB of RAM, in addition to the RAM required by Netware, is recommended for **c-tree Server** for Novell installations configured for 128 `CONNECTIONS` and higher.

The minimum hard drive space required by the **c-tree Server** for Novell is:

The size of the **c-tree Server** executable,
+ the amount specified by the `LOG_SPACE` keyword (10 MB default),

+ 1MB for **c-tree Server** status logs,

+ 3.5MB for the pre-compiled **c-tree Server** utilities,

+ the size of the data (.dat) and index (.idx) files.

The NetWare 5 and 6 operating systems support files up to 4 GB using standard **c-tree Plus** files. All other versions of NetWare support 2GB file sizes. All NetWare versions require segmented files to support files larger than these limits.

c-tree Server for Novell Installation

1. Insert the **c-tree Server** CD into the proper drive on a node of the Novell Network, not necessarily the Novell file server machine.
2. If on a Win32 system, use the automatic installation. Otherwise, copy the files from the directories below /servers/NLM/ to the intended destination directory and remove the "Read-Only" attribute from the files.
3. Activate the **c-tree Server** using the *fcactvat* program. See the FairCom Activation Key flier for instructions. Some **c-tree Server** OEM vendors may provide pre-activated **c-tree Servers** with their applications.
4. Copy the activated *ctsrvr.nlm* to the desired directory (see the example below).
5. To load the **c-tree Server**, type the following from the Novell Server's Console:

```
load <path>\ctsrvr.nlm
```

Note: The **c-tree Server** for Novell places logs, status files, etc., in the root directory of the Netware file server machine by default. Unless directed to another directory, user data and index files will also be placed in the root directory. c-tree parameter files and any *ctsrvr.cfg* file should be placed in the NLM machine root directory.

Example: If the Novell Netware drive is loaded as drive E: on your network node, the following command copies *ctsrvr.nlm* from your local directory, which is assumed to be the directory where the **c-tree Server** was installed:

```
copy ctsrvr.nlm E:\SYSTEM\<path>
```

When the c-tree Server for Novell is loaded, it places its logs and status files in the root directory of drive E: by default. Use the `SERVER_DIRECTORY` keyword to change the **c-tree Server** working directory.

Note: When using the SPX protocol, include the `CONSOLE NO_SHUTDOWN_PROMPT` Server keyword to avoid trouble when unloading a **c-tree Server**.

Running Multiple NLM Servers on One Machine

It is possible to run multiple c-tree Servers on one NLM machine. The necessary steps are detailed in the "Running Multiple c-tree Servers" section of this Chapter.

2.3 c-tree Server for Macintosh

Operational Environment

This version of the **c-tree Server** is designed specifically to work on the Apple Macintosh platform. Applications using this release can communicate via either the ADSP communication protocol or TCP/IP. As with most **c-tree Servers**, Macintosh client processes can execute on the same machine as the c-tree Server for Macintosh.

Macintosh Server Installation

To install the c-tree Server for Macintosh on your platform, take the following steps:

1) Insert the **c-tree Server** CD into the drive.
2) From the Macintosh folder, open the folder matching your OS (Mac OS X or Mac OS 9 and below).
3) Drag the **c-tree Server** folder to the desired location on your hard drive.
4) Read the notes below in the General Information section prior to starting the c-tree Server for Macintosh.

General Information

Before trying to start the c-tree Server for Macintosh, execute the *fcactvat* program. See the **c-tree Server** Activation Key flyer for instructions. Some **c-tree Server** OEM vendors provide pre-activated **c-tree Servers** with their applications.

Feel free to move the utilities included in the **c-tree Server** folder to another folder. It is not necessary for the utilities to reside in the **c-tree Server** directory.

The c-tree Server for Macintosh can dynamically load communication protocols. The c-tree Server for Macintosh always loads the ADSP communication protocol. To support ADSP and TCP/IP concurrently, set the `COMM_PROTOCOL` configuration keyword in *ctsrvr.cfg* as follows:

```
COMM_PROTOCOL  FMacTCP
```

See "`COMM_PROTOCOL`" in this guide for more information. The c-tree Server for Macintosh using TCP/IP can support other operating system client processes. For example, connect Unix and Windows clients to the Mac Server along with a Macintosh client.

For Macintosh networks with multiple zones, the c-tree Server for Macintosh naming convention can incorporate the zone name as follows:

```
FAIRCOMS@ZoneName
```

`FAIRCOMS` is the default **c-tree Server** name and is defined by the `SERVER_NAME` keyword. Contact your network administrator to determine whether network zones are defined and what zones are appropriate for your Server.

NOTE: The **c-tree Server** for Mac OS X expects the configuration file, *ctsrvr.cfg*, to be a standard Unix text file.

Minimum Hardware

The minimum CPU required by the c-tree Server for Macintosh is the fastest 68K processor or the PowerPC 60 Mhz processor. The RAM required to operate the c-tree Server for Macintosh is the RAM required by the operating system plus 4MB RAM for 8 users or 8MB for 9 users or more.

The minimum hard drive space required by the c-tree Server for Macintosh is:

The size of the **c-tree Server** executable,

+ the amount specified by the `LOG_SPACE` keyword (10 MB default),

+ 1MB for **c-tree Server** status logs,

+ 2MB for the pre-compiled **c-tree Server** utilities,

+ the size of the data (.dat) and index (.idx) files.

This operating system supports 2GB file sizes and requires segmented files to support larger files.

To increase the amount of memory available to the c-tree Server for Macintosh, do the following:

1. Single click on the *ctsrvr* icon.
2. Select the Get Info from the Finder File menu.
3. Increase the preferred memory size to the desired amount.

2.4 c-tree Unix-based Servers

Overview

The following c-tree Unix Servers are supported at the printing of this Guide:

AT&T SVR4	FreeBSD
Hewlett Packard HP-UX 10 and 11	IBM AIX 4.2 and 4.3
Linux (Intel, PPC, and Sparc)	LynxOS
Mac OS X	NetBSD
QNX and QNX RTP	SCO OpenServer/Unixware
Silicon Graphics (SGI-Irix)	Solaris (Intel and SPARC)
Sun Interactive Unix	SunOS
Tru64 Unix	

The above **c-tree Servers** are installed by following the same general method and for the most part share the same hardware requirements. Items specific to a particular **c-tree Server** are discussed at the end of this section.

Operational Environment

The c-tree Unix Servers are shipped on the **c-tree Server** CD, which contains the **c-tree Server** executable, *ctsrvr*, and the utility and companion programs discussed throughout this Guide.

Minimum Hardware

The minimum hard drive space required by all c-tree Unix Servers is:

The size of the **c-tree Server** executable,

+ the amount specified by the `LOG_SPACE` keyword (10 MB default),

+ 1MB for **c-tree Server** status logs,

+ 3.5MB for the pre-compiled **c-tree Server** utilities,

+ the size of the data (.dat) and index (.idx) files.

c-tree Server Unix Installation

1) Make the desired directory where the **c-tree Server** is to be installed the current directory. We suggest /usr/fairserv.
2) Place the **c-tree Server** CD in the drive and copy the files in the CD directories below /servers/<platform> to the desired directory.
3) When using shared memory or message queue protocols, a directory by the name /usr/ctsrv must exist prior to running the **c-tree Server**. The **c-tree Server** does

not have to be resident in /usr/ctsrv; however temporary files are created in this directory. Create this directory with sufficient permissions for the **c-tree Server** process to read, write, create and delete files within the directory.

4) After installation, activate the **c-tree Server** using the *fcactvat* program. See the **c-tree Server** Activation Key flier for instructions. Some **c-tree Server** OEM vendors provide pre-activated **c-tree Servers** with their applications.

Native Threads

FairCom offers two **c-tree Server** executables for the following platforms:

Hewlett-Packard HP-UX	IBM RS/6000 AIX V4.2 and above
Linux (Intel, PPC, and Sparc)	LynxOS V3.x and above
Mac OS X	SCO Unixware
Silicon Graphics (SGI-Irix)	Sun Solaris (Intel and SPARC)

The c-tree Native Thread Server supports the native threading routines. The c-tree Proprietary Thread Server uses FairCom's proprietary threading technology. FairCom recommends using the Native Thread Server when available since a possible performance enhancement may result. **NOTE:** The Native Thread Server supports only the TCP/IP communications protocol.

Unix Server Platform Hardware Requirements

The requirements for the **c-tree Server** on each listed operating system follow:

AT&T SVR4

The c-tree Server for AT&T SVR4 requires a Pentium 133 or greater CPU and a minimum of 2MB RAM. This **c-tree Server** supports AT&T System V Release 4 on Intel platforms and most System V Release 4 100% binary compatible Unix versions. This operating system supports 2GB file sizes and requires segmented files to support larger files.

FreeBSD

The c-tree Server for FreeBSD requires a Pentium 133 or greater CPU and a minimum of 2MB RAM. This operating system supports standard **c-tree Plus** files up to 4 GB in size and allows huge files.

Hewlett Packard HP-UX

The c-tree Server for HP-UX requires a minimum of 2MB RAM. The HPUX 11 operating system, and above, supports standard **c-tree Plus** files up to 4 GB in size and allows huge files. Earlier versions support 2GB file sizes and requires segmented files to support larger files.

IBM AIX

The c-tree IBM Server for AIX requires a minimum of 2MB RAM. There are specific versions for AIX v3.x/4.1, v4.2, v4.3, and v5.1 and above. The AIX v4.2 and above versions support both native and proprietary threading. See the section on native threads for additional information. The AIX v4.2 (and above) operating systems can be configured to support standard **c-tree Plus** files up to 4 GB in size and allows huge files. Earlier versions, and versions with drives configured for smaller files, support 2GB file sizes and requires segmented files to support larger files.

Linux

The c-tree Server for Linux requires a Pentium 133, Sparc, or PPC CPU and a minimum of 2MB RAM. Linux versions using kernel 2.400 and above support Standard **c-tree Plus** files up to 4 GB in size and allows huge files. Earlier versions support 2GB file sizes and requires segmented files to support larger files.

LynxOS

The c-tree Server for Lynx requires a Pentium 133 or greater CPU and a minimum of 2MB RAM. This **c-tree Server** supports only the TCP/IP communication protocol. This operating system supports only 2GB file sizes and requires segmented files to support larger files.

Mac OS X

The c-tree Server for Mac OS X requires a minimum of 3MB RAM beyond the requirements to run Mac OS X. This operating system supports only 2GB file sizes and requires segmented files to support larger files.

NetBSD

The c-tree Server for NetBSD requires a Pentium 133 or greater CPU and a minimum of 2MB RAM. The NetBSD operating system supports standard **c-tree Plus** files up to 4 GB in size and allows huge files.

QNX and QNX RTP

The c-tree Servers support QNX Software's proprietary communication protocol and TCP/IP. Specific requirements include a Pentium 133 or greater CPU and a minimum of 2MB RAM. This operating system supports only 2GB file sizes and requires segmented files to support larger files.

SCO OpenServer / Unixware

The c-tree Server for OpenServer and UnixWare requires a Pentium 133 or greater CPU and a minimum 2MB RAM. This operating system supports only 2GB file sizes and requires segmented files to support larger files.

Silicon Graphics (SGI-Irix)

The c-tree Server for SGI-Irix requires a minimum of 2MB RAM. This operating system supports only 2GB file sizes and requires segmented files to use larger files.

Solaris – SPARC and Intel

The c-tree Server for Solaris requires a minimum of 2MB RAM. The Solaris 2.7 and above operating systems supports standard c-tree Plus files up to 4 GB in size and allows huge files. Solaris 2.6 supports 4 GB file sizes and requires segmented files to support larger files.

Sun Interactive Unix

The c-tree Server for Interactive Unix requires a Pentium 133 or greater CPU and a minimum of 2MB RAM. This operating system supports only 2GB file sizes and requires segmented files to support larger files.

SunOS

The c-tree Server for SunOS requires a minimum of 2MB RAM. This operating system supports only 2GB file sizes and requires segmented files to support larger files.

Tru64 Unix

The c-tree Server for Tru64 requires a minimum of 2MB RAM. This **c-tree Server** supports only the TCP/IP communication protocol. The Tru64 operating system supports standard c-tree Plus files up to 4 GB in size and allows larger files with huge file support.

2.5 c-tree Server for OS/2

Operational Environment

The **c-tree Server** CD contains the c-tree Server for OS/2, communications DLL's and pre-compiled **c-tree Server** support utilities.

This c-tree Server for OS/2 supports the Shared Memory and TCP/IP communication protocols. By default, the TCP/IP DLL is activated. To activate the Shared Memory DLL, use the `COMM_PROTOCOL` keyword in a *ctsrvr.cfg* file discussed in the "Advanced Configuration Options" section of this Guide. Use the DLL's name as the token following the `COMM_PROTOCOL` keyword in the *ctsrvr.cfg*. For example, to load both communication protocols, add the following lines to *ctsrvr.cfg*:

```
COMM_PROTOCOL  F_TCPIP
COMM_PROTOCOL  FSHAREMM
```

Minimum Hardware Requirements

The minimum CPU and memory requirements for operating the c-tree Server for OS/2 are a Pentium 133 CPU, and the memory required by OS/2 plus 2MB RAM.

The minimum hard drive space required by the c-tree Server for OS/2 is:

The size of the **c-tree Server** executable,
+ the amount specified by the `LOG_SPACE` keyword (10 MB default),
+ 1MB for **c-tree Server** status logs,
+ 3.5MB for the pre-compiled **c-tree Server** utilities,
+ the size of the data (.dat) and index (.idx) files.

This operating system supports 2GB file sizes and requires segmented files to support larger files.

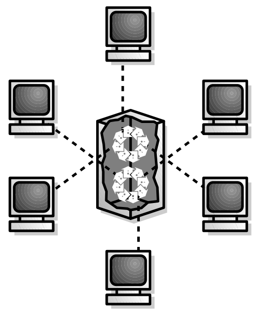
c-tree Server for OS/2 Installation

1. Insert the **c-tree Server** CD into the drive.
2. Copy the contents of the directories beneath /servers/os2/ to the desired directory.
3. After installation, activate the Server using *fcactvat*. See the c-tree Activation Key flier for instructions. Some **c-tree Server** OEM vendors may provide pre-activated **c-tree Servers** with their applications.

During installation of the **c-tree Server**, communication DLL files will be placed in the **c-tree Server** directory. These OS/2 communication DLL's can be accessed only if they are in a directory referenced by the LIBPATH environment variable. In order for the **c-tree Server** to use these DLL's use one of the following solutions:

- Copy the DLL files into the \OS2\DLL subdirectory. The \OS2\DLL subdirectory is assumed to be defined in the `LIBPATH` environment variable. This will be the case unless `LIBPATH` has been changed from the default.
- Add the subdirectory name containing the DLL's to the `LIBPATH` environment variable. The machine must be re-booted after changing the `LIBPATH` environment.

2.6 Running Multiple c-tree Servers on One Machine



It possible to execute multiple **c-tree Servers** on the same machine as long they have separate serial numbers and are given separate Server Names and directories, with two exceptions: Windows NT Services and Netware NLMs. Subtle issues must be addressed to run multiple servers on either of these platforms. The details are:

- The c-tree Server requires a unique Service Name when executed as a Service. See the "Windows NT Service" section for more information on executing two services. Each **c-tree Server** requires a unique configuration file, a unique local directory, a unique Service Name (if applicable), and a unique Server Name. Every instance of the **c-tree Server** requires a separate license.
- The NLM requires a utility to internally ‘rename’ the *ctsrvr.nlm* binary. The NLM’s internal binary name (not to be confused with the Server Name defined in the *ctsrvr.cfg* file) must be unique for each NLM. The unique NLM name is accomplished with the *RENNLM.EXE* or *RENNLMW.EXE* utilities. Execute *RENNLM.EXE* at the command line as follows, or see the *RENNLMW.EXE* description below:

```
RENNLM ctsrvr.nlm ctsrvr my_nlm "24 char Screen Name"
```

- ctsrvr.nlm is the source file name.
- ctsrvr is the original server NLM name without extension.
- my_nlm is the replacement value for ctsrvr. This is the unique NLM name, and must be different for each instance of the **c-tree Server**.
- "24 char Screen Name" is a 24-byte string shown on the console to identify the NLM.

- **CTSRVR_CFG Keyword -** Each **c-tree Server** must have a separate configuration file. Since the default would point to the same file, a command line option was added to specify a configuration file when loading the **c-tree Server**. `CTSRVR_CFG` must be followed by the full path and file name for the configuration file, for example:

  ```
  load my_nlm CTSRVR_CFG SYS:\fcs1\ctsrvr.cfg
  ```

- **Requirements -** In addition to specifying a unique configuration file, each **c-tree Server** must be assigned a unique `LOCAL_DIRECTORY` and `SERVER_NAME` in the configuration file.
- **Licensing -** Every instance of the **c-tree Server** requires a separate license.

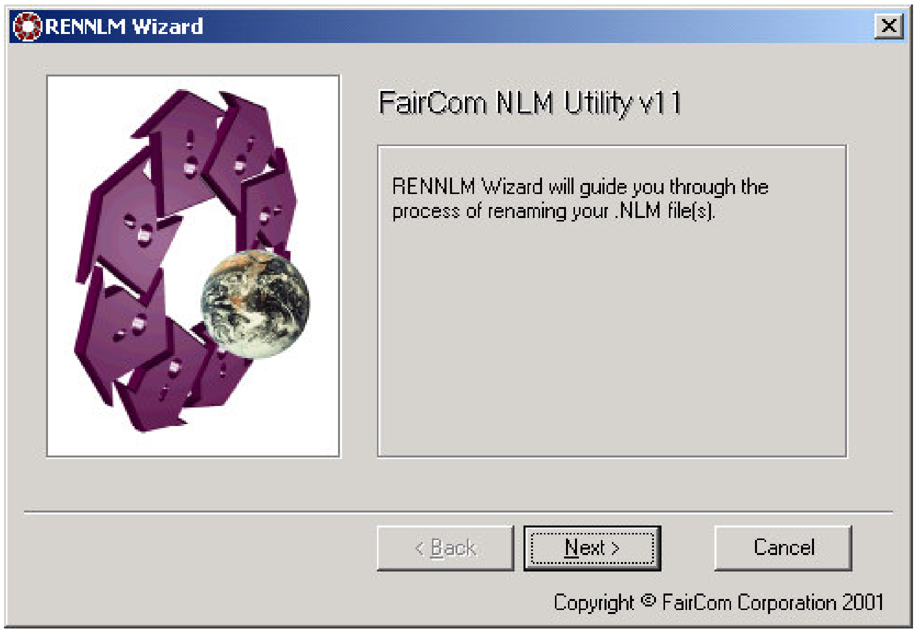
To summarize the steps in running multiple **c-tree Servers**:

- Install the **c-tree Server** as usual.
- For each instance to be run:
  - Create a working directory specific to that instance.
  - In that working directory, create a *ctsrvr.cfg* containing the keywords:
    `LOCAL_DIRECTORY` <full path to working directory>
    `SERVER_NAME` <unique Server Name for this instance>
  - On NLM:
    - In the installation directory, run *RENNLM* as described above.
    - Copy *ctsrvr.nlm* to *my_nlm.nlm* (or the unique name for this instance).
    - Activate *my_nlm.nlm* with *fcactvat* as usual.
    - Move *my_nlm.nlm* to a unique working directory.
  - When using a Windows NT/2000 Service:
    - Configure each instance to have a unique Service Name.
- Execute each instance from its working directory, using the `CTSRVR_CFG` command line keyword to specify the configuration files, as shown above.

Caution: Due to the non-preemptive nature of the Novell environment, running multiple **c-tree Servers** on the same machine is not recommended and might seriously impact performance. Even in preemptive environments, a single **c-tree Server** operates more efficiently than multiple **c-tree Servers**.

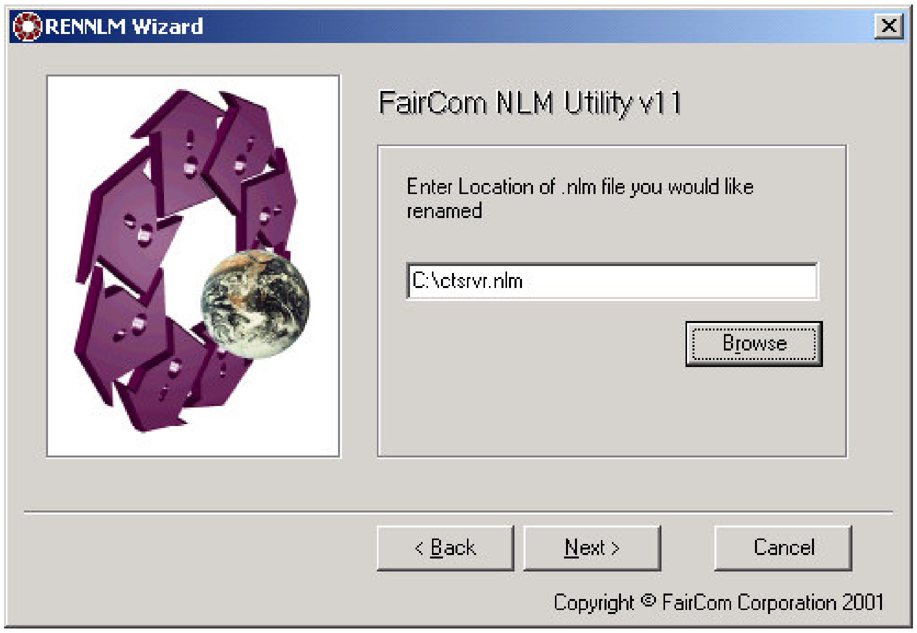
GUI NLM Rename Utility

Developers who wish to run more than one instance of the **c-tree Server** for Novell may use the GUI version of the NLM Renaming utility as shown below.



RENNLM Wizard will guide you through the process of renaming your .NLM file(s).

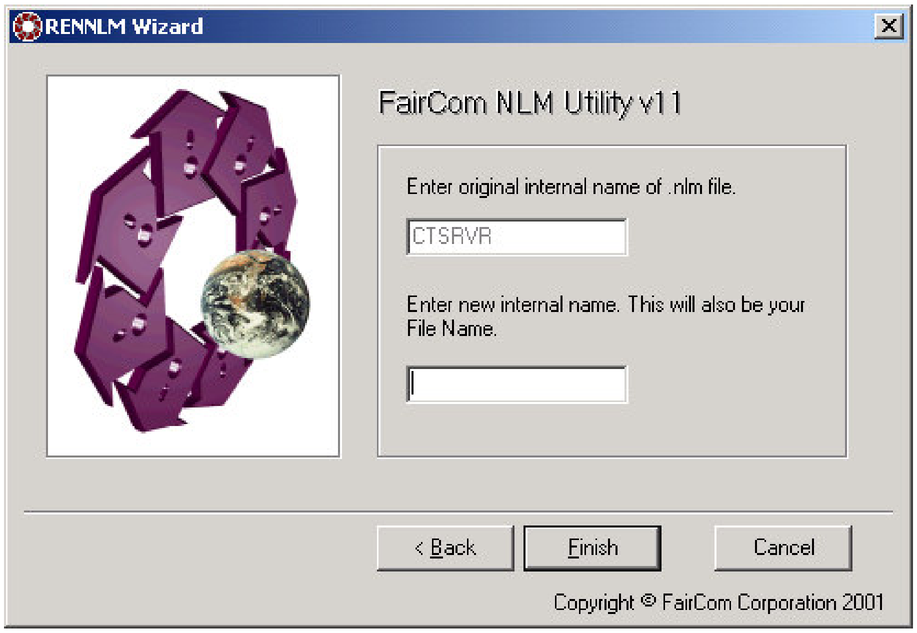
Simply provide the original file name from which the utility can generate a new NLM (by browsing the file folders, if desired):



Enter Location of .nlm file you would like renamed

C:\ctsrvr.nlm

Then provide a new name and click Finish:



Enter original internal name of .nlm file.

CTSRVR

Enter new internal name. This will also be your File Name.

Copyright © FairCom Corporation 2001

The new NLM is placed in the same directory as the original.

2.7 Legacy/Custom Platforms

FairCom formally stopped supporting the following platforms for general use:

88Open Server	Apple AUX Server
Banyan	Chorus
Data General Aviion	DOS Server
FreeBSD V2.2	IBM AIX 3.2
MIPS ABI	

If you need support for one of these platforms or a platform not currently supported by FairCom, contact your nearest FairCom office for assistance.

3. Operating the c-tree Server

Once the **c-tree Server** is installed on the operating system, it is ready to be used. Starting and stopping the **c-tree Server** are basic Administrator responsibilities, therefore this chapter is required reading.

3.1 Starting the c-tree Server the First Time

Before we cover the actual process of starting the **c-tree Server**, there are a few points to make about the Administrator's first-time duties:

1. If the vendor has supplied an (optional) settings file, *ctsrvr.set*, ensure it is in the location specified in the vendor's installation documentation. The settings file is not user-configurable. See the "Advanced Configuration" section in the "Configuring the c-tree Server" chapter for more information.
2. See if there is a **c-tree Server** configuration file, *ctsrvr.cfg*. If so, verify the file contents, change it if necessary, and prepare it to run when the **c-tree Server** starts. See the "Configuring the c-tree Server" chapter for details.
3. Change the Administrator's password to protect future access to the **c-tree Server** and access to Administrator utilities. Use the Administrator Utility described in the "c-tree Server Administrator Utility" section.

   NOTE: Initially, the **c-tree Server** recognizes only one user, who is intended to be the Administrator. This "super user" has the unchangeable User ID name of `ADMIN` and the initial password of `ADMIN`. Administrator functions can be run by anyone with knowledge of the Administrator User ID and password. The first thing to do is to change the initial password and keep the new password secure. The steps to change the password are described in the "c-tree Server Administrator Utility" section.
4. Set up initial User IDs so users can log on to the **c-tree Server**. Use the Administrator Utility, described in the "c-tree Server Administrator Utility" section.

3.2 Starting the c-tree Server

The following is a general discussion of the process used to start a **c-tree Server**. In most environments, *ctsrvr* is the name of the **c-tree Server** executable.

1. Ensure *ctsrvr* is in the base directory for database operations. See the "c-tree Server Installation" chapter in this guide.
2. If reconfiguring the **c-tree Server**, use a text editor to create a configuration file, *ctsrvr.cfg*. See the section in this manual on "Configuring the c-tree Server".
3. If adding (or changing) a configuration file, make sure it is in the same directory as the **c-tree Server** (SYS:\ for NLM), is optionally set with the `FCSRVR_CFG` environment variable, or is listed on the command line (`CTSRVR_CFG <file>`).

NOTE: If the configuration file is not found by the **c-tree Server**, the Server will not use the customized configuration file but will begin operation using default configuration settings. Check the installation instructions for your platform in the "c-tree Server Installation" chapter for any exceptions.

4. Start the **c-tree Server** by entering or selecting the name of the **c-tree Server** executable file, *ctsrvr*, just as any ordinary program in the environment.

NOTE: The **c-tree Server** name may have a file extension - see the platform specific information in the "c-tree Server Installation" chapter for details.

NOTE: No password is required to start the **c-tree Server**, therefore an automated process, such as a batch, script, or cron process, may start the **c-tree Server**.

Every time the **c-tree Server** starts, it checks log files made when it last stopped and, if necessary, uses these files to automatically recover from problems. See "Automatic Recovery" for details.

In most Unix environments, FairCom recommends Administrators run the **c-tree Server** in background to decrease the opportunity to unwittingly terminate the **c-tree Server**. For example,

```
ctsrvr &
```

The Unix "no hang up" option may also be used to keep the **c-tree Server** from being terminated if the user starting the **c-tree Server** logs off the system. For example,

```
nohup ctsrvr &
```

3.3 Launching c-tree Server companion executable

The `SIGNAL_READY` keyword provides the ability to launch an executable when the **c-tree Server** comes up. This keyword takes as its argument the name of an executable to be launched when the **c-tree Server** is ready (i.e., automatic recovery is completed). See `SIGNAL_READY` in the "Advanced Configuration" section for additional information.

This option allows applications that require the Server to launch automatically. Some examples including an auditing application, subordinate controlling interfaces (bar code readers, etc.), or a client running on the same machine as the Server.

3.4 Start Up Errors

The **c-tree Server** verifies database integrity and the operation of its own components at startup. If any problems are detected, the **c-tree Server** places error messages in the **c-tree Server** Status Log, *CTSTATUS.FCS*, and displays them on the system console. In extreme cases, the **c-tree Server** halts operation. Several kinds of errors can occur at startup.

These errors, and the appropriate reaction to each, are as follows:

Error	Explanation and Reaction to Error
12	A file required during automatic recovery cannot be located. Either it was removed after the Server failed or the physical media (e.g. disk drive) was damaged during a failure. To recover from this problem, reload the last complete system backup. See the "Maintaining Database Integrity" chapter for more information.
46	File number overflow. Too many files opened. Verify the configuration to ensure the proper file allocations and limits. Verify the applications are opening the appropriate files in the appropriate manner.
96	A log file required for recovery is not available. To recover from this problem, reload the last complete system backup. See the "Maintaining Database Integrity" chapter for more information.
143	Communication handler not installed. Be sure the network drivers are loaded or the shared memory directory (*/usr/ctsrv*) for Unix platforms has been created.
173 174	Can also occur if the Server is unable to create a socket for the client connection. Socket creation will fail if the available file descriptors for the Server process have been used (if the server has many files open, for example). Check the per-process file descriptor limit on your system. This limit must be large enough to accommodate the number of files you wish to open (the `FILES` setting in the Server configuration), plus the number of clients to connect to the Server at a time (the `CONNECTIONS` or `USERS` setting in the Server configuration).
509	Another copy of the particular **c-tree Server** you attempted to start is already running. Each installed copy of the **c-tree Server** must have its own license and a unique serial number. If you do not have enough **c-tree Server** licenses, please contact your **c-tree Server** provider.

NOTE: Utility error messages covered in this manual refer to messages a **c-tree Server** sends to a program connected to it. Although we list error numbers with brief explanations of each error, it is important to understand how errors are treated, including messages sent to users. It is the responsibility of the client application programs receiving the error messages to properly capture and display the errors.

3.5 Server Operational Errors

The **c-tree Server** performs rigorous error checking and logging during the course of everyday operation. Because of the depth of error checking that is performed, warnings and error messages are logged in even the most benign situations.

VDP Errors (127/128)

When a communication error such as `VDP_ERROR` (127 or 128) occurs, the **c-tree Server** logs an entry in the Server Status Log *CTSTATUS.FCS*. This is not a serious situation unless the client application is also getting errors such as 127, 128, or similar errors.

The context of the VDP error is that a server thread gets a notification that a message is available, but when the Server performs a read, nothing is returned.

This can be caused by:

- Physical network problems
- An overworked network transport layer that is timing out and doing retries
- Clients exit without calling **CloseISAM** or **StopUser**, or end users that turn their machines off without properly logging out of the application.

To ensure the errors are not serious, try to reconcile the VDP errors in the log with the client events that triggered them. Since these errors do not usually happen frequently and user names are provided, it should be easy to determine which event caused this situation.

To avoid there errors, ensure the **c-tree Server's** host machine is not burdened beyond its capacity. Using a more powerful machine or limiting the number and types of applications on a machine can improve performance and limit errors at the communication level. Also, ensure no specific application is over-using resources on the host machine. If appropriate in the Server's operating environment, increasing the priority of the c-tree Server can eliminate or reduce VDP errors. This should be done cautiously as it will affect other applications running on the same machine.

The error messages in the Server Status Log can be turned off, but unless they are an inconvenience, **this is not recommended**. The messages serve as a good health check on the state of your network and may be an early warning of more serious network and system problems. To disable the messages, add `CTSTATUS_MASK  VDP_ERROR` to the *ctsrvr.cfg* file and restart the **c-tree Server**.

749X

Most 749X errors are the result of memory corruption on the host machine. These errors are extremely rare, but can be very serious. In most cases, restarting the Server will clear a transient memory error. If these errors repeat, contact your application vendor.

Value	Cause
7490	A request for memory beyond reasonable limits.
7491	Attempting to return memory carrying an unreasonable large length value in its control header.
7492	Encountered an invalid “availability” counter during a sub-allocator get operation.
7493	An invalid “availability” counter while returning a block of memory.
7494	An invalid header pointer for an allocation block.
7495	A list connected with c-tree's memory sub-allocator has become invalid or corrupted.
7496	Failure in memory swapping operation.
7497	Attempting to return memory carrying a zero length in its control header.
7498	An invalid semaphore owner while getting memory.
7499	An invalid semaphore owner while returning memory.

8770

The 8770 error occurs when the Server attempts to remove an internal unique file ID from a list as a file is closed, but that file ID is not on the list. This might be caused by an application opening different files with the same internal file ID. This would typically be the case when a file is copied and both files are then opened; therefore they both have the same file ID. To avoid these errors do not copy server-controlled files. If the 8770 error occurred after another more serious error, the 8770 error can be safely ignored. If it recurs, contact your application developer for assistance.

3.6 Stopping the c-tree Server

Only a user in the `ADMIN` group can stop the **c-tree Server**. The Server can be stopped using the Windows Close icon or menu item, using the *ctstop* or *ctadmn* utilities, or by an application using the **StopServer** function.

To stop the c-tree Server with the Close button in the upper right corner of the application window, or with the Close item in the File menu, just click either object as any other Windows application. An `ADMIN` group User ID and password is required to complete the close operation.

To stop the **c-tree Server** with the module *ctstop*, a special client application:

1. Start this program like any other.
2. The stop module asks for four things:
   a) The `ADMIN` user ID, which must be `ADMIN` or a member of the `ADMIN` group.
   b) The `ADMIN` password, which is necessary for continuing with the procedure.
   c) The current name of the **c-tree Server**, if an alternative to the default name was given in the configuration file (see the keyword `SERVER_NAME` in Basic c-tree Server Configuration Options) to specify which **c-tree Server** to stop.
   d) The delay time (if any) before shutting down the **c-tree Server**. If a greater-than-zero delay is specified, the **c-tree Server** will not accept any new users or transactions. Logon attempts during the delay time specified will fail with Error #150, which means, "The Server is shutting down". New transactions cannot be started while waiting to shut down. They will return Error #150 or Error #162, "Server gone", depending on how far the shutdown process has gone.

The **c-tree Server** may also be stopped by an application program, as long as it supplies an `ADMIN` group User ID and password, using **StopServer** discussed in the **c-tree Plus Programmer's Reference Guide**, (distributed only to **c-tree Plus** developers).

During **c-tree Server** shutdown, messages reflect when communications terminate and when the final system checkpoint begins. In addition, two aspects of the shutdown that involve loops with two-second delays generate output indicating their status. The first loop permits the delete node queue to be worked down. The second loop permits clients to shutdown cleanly during **c-tree Server** shutdown. If these loops are entered, the **c-tree Server** could take a measurable amount of time to shut down, depending on the amount of work to be done, and output indicates how many queue entries or clients remain. A notice indicates whether everything was cleaned-up. A clean-up notice is NOT generated if a loop was not entered.

This output permits a **c-tree Server** Administrator to monitor the shutdown, and avoid an incorrect assumption about whether the **c-tree Server** is making progress or has hung during shutdown. After the **c-tree Server** shuts down, it sends a message saying **c-tree Server** operations have been terminated. The output is routed to the console and *CTSTATUS.FCS*, although the latter does not receive the numeric information concerning the number of queue entries or active clients.

3.7 Launching Server companion upon shutdown

The `SIGNAL_DOWN` keyword provides the ability to launch a customer executable when the **c-tree Server** comes down. This keyword takes as its argument the name of an executable that will be launched when the **c-tree Server** has been successfully terminated. See `SIGNAL_DOWN` in the "Advanced Configuration" section for additional information.

This option could be used to launch a backup utility, to relaunch the Server, or to execute a batch/shell script to perform actions that can only be performed while the Server is inoperative.

3.8 c-tree Server Service for Windows NT/2000/XP

Windows NT/2000/XP supports a special type of process known as a service. A service is a background Win32 process that receives special treatment from the operating system. Services may be configured to start automatically at system startup or to start manually by a user. Services typically have no user interface and can continue to run even when no users are logged on to the system. The operating system automatically terminates services at system shutdown or a user can manually terminate them.

The Service support consists of three components: the Windows Service Control Manager (SCM), the service executable, and the service control program (SCP). Windows NT/2000/XP provides access to the SCM from the Services Control Panel applet. Application vendors provide the service executable and SCP.

FairCom has developed a version of the c-tree Server that runs as a Windows Service. The **c-tree Server Service** consists of the following components:

ctsrvr.exe	**c-tree Server Service** executable
ctntinst.exe	**c-tree Server Service Control Program** (SCP)

The c-tree Server Service features all of the capabilities and advantages of Windows services described above. As with any service, the c-tree Server Service can be configured to start automatically when the machine comes up, can run invisibly no matter which users are logged on or if no user is logged on, and will shut down automatically when the host machine shuts down.

Use the **c-tree Server SCP** command-line utility to install, configure, and control the **c-tree Server Service**. The following command-line options are available:

```
ctntinst [executableName] option
         [-u <username> [-p <password>]] [serviceName]
```

- *executableName* is the file name of the desired c-tree Server executable.
- *option* is a single entry selected from the table below:

`-install`	Install the **c-tree Server Service**.
`-remove`	Uninstall the **c-tree Server Service**.
`-auto`	Set service to start up automatically.
`-demand`	Set service to start up on demand.
`-showconfig`	Show service settings.
`-status`	Show current status of service.
`-start`	Start the **c-tree Server Service**.
`-stop`	Stop the **c-tree Server Service**.

- *username* is the name of the account under which the service should run. Use an account name in the form: DomainName\UserName. The service process logs on as this user. If the account belongs to the built-in domain, specify .\UserName. If the –u option is not specified, the service runs under the LocalSystem account.
- *password* is the password for the account name specified by the *username* option. If the account has no password or if the service runs under the LocalService, NetworkService, or LocalSystem accounts, omit the –p option.
- *serviceName* is an optional service name used to provide a unique service name. This is required if more than one **c-tree Server Service** is to be executed on the same physical machine. See below for additional details on starting multiple **c-tree Server Services**. *serviceName* defaults to "c-treeServer" if not specified.

Installing the c-tree Server Service

To install the **c-tree Server** as a Windows NT/2000 Service when this option was not selected during the original installation, run *ctntinst.exe*, specifying the `-install` option. You may optionally specify the service name. For example:

```
c:\faircom\bin.svc> ctntinst -install
c-treeServer Created
```

Installing Multiple Instances of the c-tree Server

To run multiple instances of the **c-tree Server** on one machine, each instance must be individually licensed, have a unique name, and have a unique set of server and data files. **c-tree Servers** cannot share files.

For each **c-tree Server**, you must:

1. Install each instance of the **c-tree Server** into a unique directory.
2. Activate each license as described in the "c-tree Server Activation Key" flier. Every instance of the **c-tree Server** requires a separate license.
3. Specify a unique Server Name in the configuration file, *ctsrvr.cfg*, as described in Chapter 6 of the **c-tree Server Administrator's Guide**.

4. Configure each instance to have a unique Service Name as described in the "Configuring the c-tree Server Service" section.

See the "Running Multiple c-tree Servers on One Machine" section for more information.

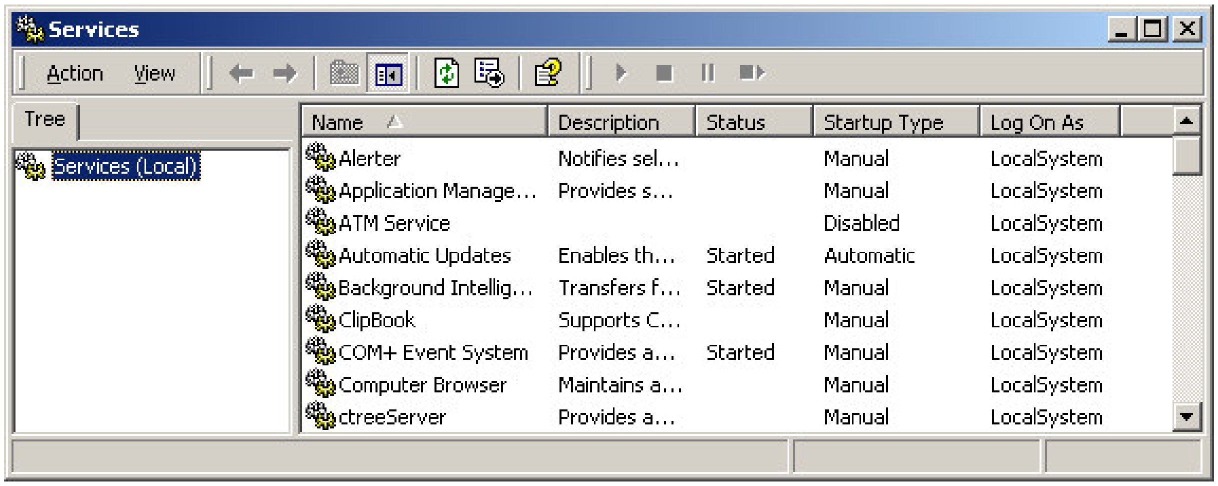
Configuring the c-tree Server Service

The **c-tree Server Service** has two configurable properties: the Startup Type and Logon User. Set the Startup Type using *ctntinst.exe*, specifying the `-auto` for automatic startup at system boot time, or `-demand` for manual startup. For example:

```
c:\faircom\bin.svc> ctntinst -auto
Changed Service Configuration Successfully

c:\faircom\bin.svc> ctntinst -demand
Changed Service Configuration Successfully
```

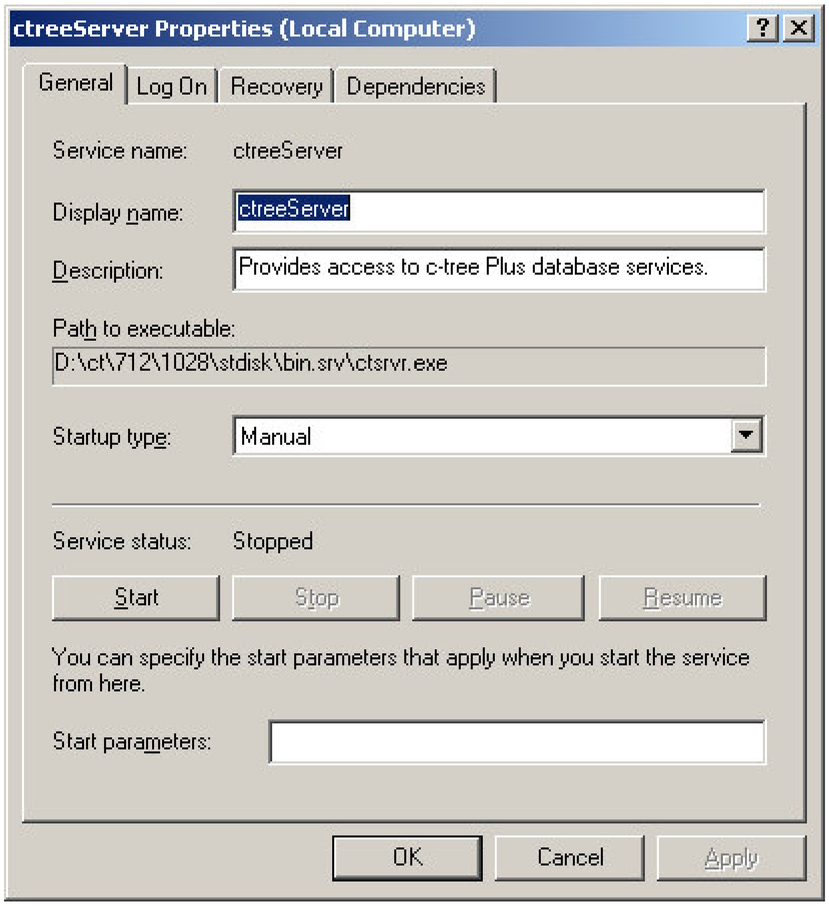
The Logon User and the Startup Type (see Figure 2) can also be set using the Windows NT Services Control Panel applet. To do so, open the Windows Control Panel (by clicking Start, Settings, Control Panel) and select the Services applet. You will be presented with a list of the installed services (Figure 1).



Name	Description	Status	Startup Type	Log On As
Alerter	Notifies sel...		Manual	LocalSystem
Application Manage...	Provides s...		Manual	LocalSystem
ATM Service			Disabled	LocalSystem
Automatic Updates	Enables th...	Started	Automatic	LocalSystem
Background Intellig...	Transfers f...	Started	Manual	LocalSystem
ClipBook	Supports C...		Manual	LocalSystem
COM+ Event System	Provides a...	Started	Manual	LocalSystem
Computer Browser	Maintains a...		Manual	LocalSystem
ctreeServer	Provides a...		Manual	LocalSystem

Figure 1. Windows 2000 Services Control Panel applet.

Select the **c-tree Server Service**, and click the button labeled "Startup…". The service configuration options window will appear (Figure 2). Set the Startup Type and Logon User, as desired.



Service name: ctreeServer

Display name: ctreeServer

Description: Provides access to c-tree Plus database services.

Path to executable:
D:\ct\712\1028\stdisk\bin.srv\ctsrvr.exe

Startup type: Manual

Service status: Stopped

You can specify the start parameters that apply when you start the service from here.

Start parameters:

Figure 2. Windows 2000 Service configuration options.

To display the current configuration for the **c-tree Server Service** using the **c-tree Server SCP**, run *ctntinst.exe*, specifying the `-showconfig` option.

For example:

```
c:\faircom\bin.svc> ctntinst -showconfig
Service Configuration for c-treeServer
Display Name:      c-treeServer
Type:              SERVICE_WIN32_OWN_PROCESS
Start Type:        Manual
Error Level:       SERVICE_ERROR_NORMAL
Binary path:       c:\faircom\bin.svc\ctsrvr.exe
Load Order Group:  None
Tag ID:            0
Dependencies:      None
Login Under:       LocalSystem
```

Starting the c-tree Server Service

Start the **c-tree Server Service** using either the **c-tree Server SCP** or the Windows Services Control Panel applet.

- Start the **c-tree Server Service** using the FairCom SCP utility using *ctntinst.exe* with the `-start` option. For example:

```
c:\faircom\bin.svc> ctntinst -start
Starting the c-tree Server service...
c-treeServer started successfully.
```

- Start the c-tree Server Service using the Windows NT Services Control Panel applet by opening the Control Panel and selecting the Services applet. From the list of the installed services (Figure 1), select the name for the **c-tree Server Service** ("c-treeServer" by default). Click "Start" to start the **c-tree Server Service**.

Displaying the current status of the c-tree Server Service

The current status of the **c-tree Server Service** can be determined by using either **c-tree Server SCP** or the Windows Services Control Panel applet.

- To check the current status of the **c-tree Server Service** using **c-tree Server SCP**, run *ctntinst.exe*, specifying the `-status` option. For example:

```
c:\faircom\bin.svc> ctntinst -status
Service Status for c-treeServer
Current State:     RUNNING
Controls Accepted: STOP  SHUTDOWN
Win32 Exit:        0
Service Exit:      0
Checkpoint:        0x0
WaitHint:          0x0
```

- To check the current status of the **c-tree Server Service** using the Windows Services Control Panel applet, open the Control Panel. Select the Services applet. If the **c-tree Server Service** is running, the Status field shows "Started". Otherwise the Status field is blank.

Stopping the c-tree Server Service

The **c-tree Server Service** can be stopped by using either the **c-tree Server SCP** or the Windows Services Control Panel applet.

- Stop the **c-tree Server Service** using the **c-tree Server SCP** by running *ctntinst.exe* with the `-stop` option. *ctntinst.exe* signals the **c-tree Server Service** to stop, but does not wait for the service to terminate before exiting. Use the `-status` option to verify the service stopped. For example:

```
c:\faircom\bin.svc> ctntinst -stop
Stopping the c-tree Server service...
Service Status for c-treeServer
Current State:     RUNNING
Controls Accepted: STOP  SHUTDOWN
```

```
Win32 Exit:        0
Service Exit:      0
Checkpoint:        0x0
WaitHint:          0x0
```

- Stop the **c-tree Server Service** using the Windows Services Control Panel applet by opening the Control Panel. Select the Services applet. Select the **c-tree Server Service**. Click "Stop". To verify you want to stop the service, click "Yes".
- If the **c-tree Server Service** is configured to allow user interaction, clicking the close gadget stops the Service.
- The Service stops automatically when Windows signals the operating system itself is shutting down. A clean shutdown of Windows should result in a clean shutdown of the **c-tree Server Service**. However, since Windows only allows a 20-second delay for service shutdown, FairCom recommends all files be maintained under transaction processing to allow automatic recovery if cache cannot be safely flushed to prevent data corruption. The default 20-second delay can be adjusted using the WaitToKillServiceTimeout registry key, found in HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\, when present.

Removing the c-tree Server Service

If you wish to remove the c-tree Server Service from the list of installed Windows NT Services, run *ctntinst.exe*, specifying the `-remove` option. You may optionally specify the service name. For example:

```
c:\faircom\bin.svc> ctntinst -remove MyService
MyService Removed
```

Troubleshooting Tips

This section identifies possible problems that may be encountered when using the c-tree Server Service, and ways to diagnose and solve them.

Problems starting the c-tree Server Service

If the c-tree Server Service fails to start, it returns a service-specific error, and logs a message to the Windows NT application event log. This information can be used to determine the reason why the c-tree Server Service failed to start up. Below is the output of a failed startup when starting the c-tree Server Service using FairCom's SCP. The service-specific error is displayed as the "Service Exit" code.

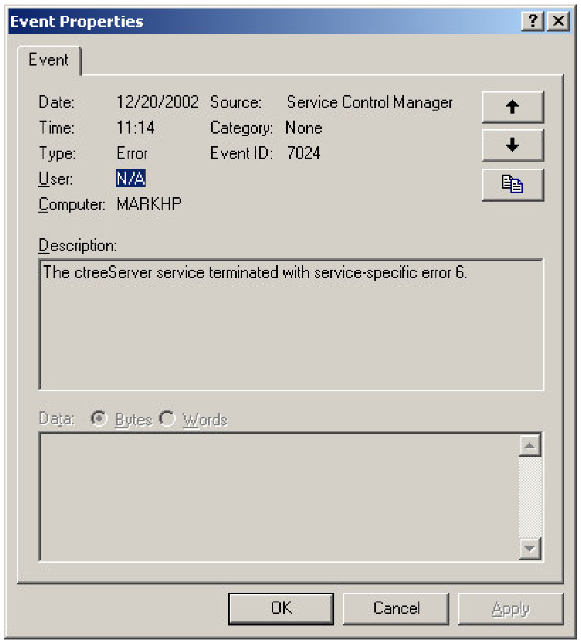
```
c:\faircom\bin.svc> ctntinst -start
Starting the c-tree Server service...
c-treeServer start unsuccessful:
Current State: STOPPED
Win32 Exit:    1066
Service Exit:  6
Checkpoint:    0x0
WaitHint:      0x0
```

Table 1 shows possible service-specific errors returned by the **c-tree Server Service**, the corresponding message, and possible causes for each of these errors.

Error Code	Error Message	Possible Causes
2	(Varies)	An operating system function call failed. See the event log for a detailed error message.
4	The Server's settings file is missing. It is required to operate this server.	The **c-tree Server** requires a settings file, but it was not found.
5	The current Server's settings file is invalid.	The **c-tree Server** requires a settings file, and the settings file that was found was not valid. Contact your application developer for assistance.
6	This server must be activated with a FairCom activation key in order to operate. See 'Server Activation Key Card' within your package for more information.	The **c-tree Server** has not yet been activated.

Table 1. c-tree Server Service-specific errors.

Use the Windows Event Viewer to list events reported by the **c-tree Server Service**. Start the Event Viewer and select the Application log option from the Log menu. Events logged by the **c-tree Server Service** have the "Source" field set to the service name ("c-treeServer" by default). Double-clicking an event displays the event detail (Figure 3).



Date: 12/20/2002 Source: Service Control Manager
Time: 11:14 Category: None
Type: Error Event ID: 7024
User: N/A
Computer: MARKHP

Description:
The ctreeServer service terminated with service-specific error 6.

Figure 3. Using the Event Viewer to display events logged by the c-tree Server Service.

Problems connecting to the c-tree Server Service

If client applications are unable to connect to the **c-tree Server Service**, verify that the **c-tree Server Service** is running (See "Displaying the current status of the c-tree Server Service" above for details).

If the **c-tree Server Service** is running, check the **c-tree Server** status log file (*CTSTATUS.FCS*, typically located in the directory in which the **c-tree Server** executable resides) for the following information:

1. Are there any error messages logged to *CTSTATUS.FCS*?
2. Is the Server Name displayed in *CTSTATUS.FCS* the same Server Name your client applications are using?
3. Are the protocols displayed in *CTSTATUS.FCS* the same as those your client applications are using?

FairCom’s *ctadmn.exe* utility (provided with the **c-tree Server**) is also a useful tool for verifying whether clients can connect to the **c-tree Server**.

Problems stopping the c-tree Server Service

If you are unable to stop the **c-tree Server Service**, check the event log for an error message. Also check for error messages in the **c-tree Server** status log file (*CTSTATUS.FCS*, typically located in the directory in which the **c-tree Server** executable resides).

FairCom’s *ctadmn.exe* utility (provided with the **c-tree Server**) can also be used to stop the **c-tree Server Service**.

4. Controlling c-tree Server Access

One of the main responsibilities of a **c-tree Server** Administrator is to establish and maintain access to the **c-tree Server**. Although reviewing this chapter is not required for operating the **c-tree Server**, FairCom recommends Administrators consider the following features.

Access to the **c-tree Server** can be controlled in four basic ways:

User access restrictions	Requiring User ID and/or User password to access the **c-tree Server**.
File access restrictions	Requiring file password to access a file.
File operation permissions	Controlling which specific operations (e.g., read, write, delete) a given class of users can perform on a particular file they have accessed.
Group-based restrictions	Defining groupings, then assigning users to given groups and giving appropriate file permissions only to members of a specific group.

The details of access and security control through user, file and group information are covered in this section. Basic concepts needed to understand security operations are covered first. Descriptions of the Administrator Utility used to enter security information for the **c-tree Server** and monitor users while they are connected to the **c-tree Server** follow.

NOTE: The controls discussed here are those available to the Administrator. Applications can also be programmed to allow certain security controls (e.g., change file passwords) to users who have appropriate access to the **c-tree Server**, using available security functions in FairCom's **c-tree Plus** API. Consult application documentation or application vendor for further login instructions.

It is important to be aware that the file security provided by the **c-tree Server** is a function of access to files through the **c-tree Server**. When files are not controlled by the **c-tree Server**, they may not be secure.

4.1 Users, Files, Groups, and File Permission Masks

This section covers the security concepts needed to understand and make use of the full range of Administrator security controls offered by the **c-tree Server**. These security features are designed to work together. For example, security instructions can be arranged allowing only certain sets of users particular rights with respect to a given file.

Users

Whenever an application connects to a **c-tree Server**, it must identify itself to the **c-tree Server**. The identifying code is called the User ID. To gain access to the **c-tree Server**, the User ID seeking access to the **c-tree Server** must be one already authorized as a valid User ID. A password for the User ID may also be required to access the **c-tree Server**.

If one attempts to log on to a **c-tree Server** with an invalid User ID (i.e., one not issued by the Administrator or created by changing an existing User ID), the **c-tree Server** will deny the request and send a message to that effect (i.e., error message 450). An attempt to log on with a valid User ID but an invalid user password will also be denied, with a message stating the reason (i.e., error message 451).

When an application, i.e., a user running a given application, logs on to the **c-tree Server**, a task user is created to identify the session with the User ID. This is relevant when monitoring or disconnecting clients from the **c-tree Server**.

The **c-tree Server** recognizes four kinds of users:

Administrator	The Administrator, or "super user", is the only user with a pre-set, and unchangeable, User ID (`ADMIN`). By having the only initial valid User ID, `ADMIN` is the first user to gain access to the **c-tree Server**. After changing the password for User ID `ADMIN` from the initial password, `ADMIN`, to a secure private password, the Administrator uses the `ADMIN` User ID and the private password to obtain exclusive access to the Administrator utilities needed to carry out the responsibilities discussed in this manual.
Unique User ID	The Administrator can create new User IDs (and passwords) for other users, who then log onto the **c-tree Server** with these names. This includes new members to the ADMIN group with limited Administrator capabilities.
Application-based User ID	Application programs can be designed to supply a given User ID code when attempting to log on the **c-tree Server**, regardless of who the user is. This User ID is treated like a unique User ID, although several users may share a common ID. In other words, the application/user is allowed onto the **c-tree Server** only if the User ID (and the password, if any) supplied to the **c-tree Server** has been authorized.
Guest Users	Users without a unique User ID. An application program can be designed to log onto a **c-tree Server** without requiring the user to supply a User ID and without supplying an application-based User ID. When no User ID is supplied to the **c-tree Server** as an application logs onto the **c-tree Server**, the **c-tree Server** automatically assigns the special User ID "GUEST" to that session.

User IDs can be up to 31 characters long. Characters can be letters, numbers, or punctuation marks. User IDs are not case sensitive (i.e., upper and lower case characters are treated as the same).

User passwords can be up to nine characters long. Characters can be letters, numbers, or punctuation marks. Passwords are case sensitive (i.e., upper case and lower case characters are treated as different).

NOTE: Users, including `ADMIN`, can use the *ctpass* utility (see Appendix A) to change their own password, and `ADMIN` can always use the c-tree Server Administrator Utility, described below, to review current passwords for all User IDs.

User ID and Membership in Groups

The Administrator can establish groups of any sort (e.g., a payroll group, a shipping room group, a data entry group) and associate each User ID to as many as 16 of these groups. For example, User ID "B.Smith" is a member of Group ID "Payroll". These connections are ordered, from the "1st" to "Nth" group membership, where N is a maximum of 16.

If the Administrator does not assign a given User ID to a group, the **c-tree Server** automatically assigns that User ID to a special group with the `GUEST` Group ID. In addition, the special `GUEST` User ID is automatically assigned to the `GUEST` group.

A primary (i.e., default) group is always defined for each User ID. This is either the Group ID for the first association or, if no Administrator established associations, the `GUEST` group. For instance, number 1 on the list of 16 possible connections between the user and groups set up by the Administrator.

These group mechanisms are important in connection with the file permission masks. See the Groups section for more information.

User ID and Ownership of Files

Each file created by the **c-tree Server** has an owner. The User ID in effect when a file is created is automatically made the owner of the file, but the Administrator can later change a current file owner to any other valid User ID. The concept of file owner is important because it can be used with the file permission mask. See the Files section for more information.

User ID and Logon Limits

The Server Administrator can set several system-wide limits and User ID overrides for those limits. The number of consecutive logon failures, the delay after failure limit is reached, and a minimum time between logons can all be set system-wide with configuration keywords. These settings can be overridden for each User ID using the Server Administration utility, *ctadmn*, which can also set beginning and ending dates for each User ID. These features are detailed below, in the Server Administration Utility section in this chapter, and in the Keywords section in the "Configuration" chapter.

The Server Administrator can set an optional limit on the number of consecutive failed logons that will cause subsequent logon attempts to fail for a specified time interval. The default logon limit is zero (0), which implies the feature is not active. Logons are blocked for 5 minutes by default after exceeding the limit. A logon during this period returns `LRSM_ERR`(584). Set the logon limit with `LOGON_FAIL_LIMIT <logon limit>` in the configuration file. The length of time the logons are blocked is set by `LOGON_FAIL_TIME <minutes>` in the configuration file.

The **c-tree Server** can be configured to require user logons within a given period. This ensures all users log on "at-least-once" within the defined time (e.g., at least once a week). If the time expires for a specific user, the Server deactivates the user’s profile, preventing access to the Server. The Server Administrator, or other `ADMIN` group user, must reset the user’s account once the time limit has elapsed. To activate this feature, add the following keyword in the Server configuration file, *ctsrvr.cfg*, where <value> is the period in minutes during which the user must logon:

```
LOGON_MUST_TIME <value>
```

Files

Database files have several security features in addition to the file permission mask, discussed in a separate section:

File Password

Files created by the **c-tree Server**, and others, can be assigned a file password when created. File passwords can be changed later by the Administrator or the file's owner, and then be required for users to access files. For example, a user could be required to enter a file password before initiating the file operations specified in the file permission mask (see below).

File passwords can be up to 9 characters long. Characters can be letters, numbers, or punctuation marks. Passwords are case sensitive (i.e., upper case and lower case characters are treated as different).

File Owner

As explained in the "Users" section, when a file is created by the **c-tree Server**, the User ID requesting the creation is established as the owner of the file. The Administrator may change the file owner any time to any other currently valid User ID. The owner is used to define one of the ways file permissions can be granted, e.g., the owner typically has permission to write to the file.

File Group

When created, a file is typically associated with the current primary group of the User ID who created the file. The file group is designed for use with the file permission mask. This can be changed later to any other currently valid Group ID for that User ID by the Administrator or owner. For example, the file permission mask may allow "group permission" to read the file, while no others can (see below).

If instructed by the user's application when it creates a file, a file's Group can be any one of the owner's other Group IDs, instead of the owner's primary Group.

The current Owner of a file may use the *ctfile* utility, after entering both the current User ID password and the current file password, to change: the file password; the file permission mask (see below); the file Group; and even the file Owner itself, which would block the user from accessing the file through the original Owner User ID. Appendix A contains a further description of this treatment.

Groups

A Group is an arbitrary category of associated User IDs and files. For example, a business wanting to separate the payroll department and the shipping department could establish a "shipping" Group and a "payroll" Group and associate appropriate User IDs with one or more of these Administrator-defined Groups. By establishing and using groups, the Administrator can offer file-level operation control to selected groups of users. For example, by using Groups along with file permission masks it is possible to enable users in the payroll department to read, but not write, to any file created by anyone else in the payroll department.

Two Kinds of Groups

The **c-tree Server** maintains a guest group, to which User IDs are associated if they are not assigned to any Administrator-defined Group ID. This means every User ID is associated with at least one group (i.e., the `GUEST` Group or a Group ID).

The Administrator can create any number of Groups each of which has a Group ID, a text description (for display), a memory allocation specification, and a list of User IDs associated with the Group ID. As noted, the Administrator can associate a given User ID with as many as 16 Group IDs. A `GUEST` User cannot be associated with any Group IDs; instead, the **c-tree Server** automatically assigns a `GUEST` User to the `GUEST` Group.

Group IDs can be up to 31 characters long. Characters can be letters, numbers, or punctuation marks. User IDs are not case sensitive (i.e., upper and lower case characters are treated as the same).

File Permission Masks

Once a user has access to a given file, which might need both user and file passwords to reach, there is one additional level of access control available. This is the "file permission mask," a set of controls over who can do what with a given file. The "what" and the "who" of file permission masks follow.

Operations controlled

User permissions with respect to the following file operations can be controlled with the file permission mask for a given file (i.e., "YES, TYPE X USERS have permission to do this operation" or "NO, TYPE X USERS do not have permission to do this operation"):

- READ the file,
- WRITE to the file (i.e., add, update, or delete individual items in the file),
- CHANGE THE DEFINITION(s) of the file, including such characteristics as alternative collating sequences or record schemas (see the **c-tree Plus Programmer's Reference Guide** for details),
- DELETE the entire file, or
- ANY combination OF THE ABOVE

If a file has no permission mask, any user who can access the file can perform all the above operations.

User Controls

Each of these permissions for a given file can be specified for any or all of the following classes of users:

WORLD access: Allow the specified file operations to any user who can access the file (so users who lack a required User ID and/or file password do not have these file-operation permissions).

OWNER access: Allow the specified file operations to the current owner of the file. The owner is either the User ID in effect when the file was created or a different User ID who was later assigned as the owner.

GROUP access: Allow the specified file operations to any User ID currently a member of the same Group as the current File Group.

In summary, a file permission mask permits different degrees of access to a file for the file's owner, users belonging to the file's group, and all other users, including guests.

Using the concepts discussed above, the Administrator can establish a sophisticated and flexible security system with the **c-tree Server**. The mechanism for actually entering information for use by the **c-tree Server** is a separate program utility, called the Administrator's Utility, *ctadmn*.

4.2 c-tree Server Administrator Utility

The **c-tree Server** Administrator Utility, *ctadmn*, is used by the Administrator to manage users, groups and files. It can also be used to monitor the users logged on to the **c-tree Server** and to disconnect any user from the **c-tree Server**.

To use this utility, do the following (which may vary between environments, depending on user-interface specifics):

1. Start running the program, *ctadmn*, as any other program in the environment.
2. Enter an `ADMIN` group User ID. Initially, only `ADMIN` will exist until you create others.
3. Supply the current password for the User ID given.

   NOTE: If this is the first time *ctadmn* has been run and the password for the User ID `ADMIN` is still `ADMIN`, change it before leaving *ctadmn* to ensure secure access to this program in the future. You can also change your password using *ctpass* (see Appendix A).
4. You will be prompted for the (optional) file password for the file *FAIRCOM.FCS*. We recommend you do NOT give this file a password since you should already be the only one who can run this utility. To confirm the absence of a password, press the return key.
5. In response to the next prompt, supply the current name for the **c-tree Server** and press enter. If the **c-tree Server** name has not been changed (see `SERVER_NAME` in the "Configuring the c-tree Server" chapter), simply press Enter to use the default name.

After finishing these steps, the main menus for the **c-tree Server** Administrator Utility will be displayed, allowing access to the following groups of operations:

1. User Operations.
2. Group Definitions.
3. File Security.
4. Monitor Clients.
5. Server Information (**IOPERFORMANCE**)
6. Server Configuration (**SystemConfiguration**)
7. Stop Server

The following sections detail the areas controlled by the Administrator Utility. The steps necessary for each operation may vary slightly in different environments.

NOTE: While using the Administration Utility, press the question mark key ("?") at any prompt to access Help.

User Operations

User options available in *ctadmn* are as follows:

- Add New User by:
  - Entering a new User ID.
  - (optional) Enter a long name (i.e., a text string) for use as a User Description (e.g., for screen displays, where the User ID may be too terse).
  - (optional) Enter a User Memory Limit, which is a maximum memory allocation for this particular user that will override maximum memory allocations set at the Server-level (for any user) or at the group level (for any particular group).
  - (optional) Enter the User Memory Rule: Absolute, Guideline, or Default. See `USR_MEM_RUL` in Chapter 6 for details.
  - (optional) Enter a user password.
  - (optional) Assign user membership in from 1 to 16 Groups (i.e., Group IDs).
  - (optional) Enter the first valid date for this User ID.
  - (optional) Enter the last valid date for this User ID.
  - (optional) Enter limit on consecutive logon failures if different from system default. See `LOGON_FAIL_LIMIT` for details.
- Remove an existing User ID.
- List authorized User IDs.
- Change the Password for a given User ID.
- Change which Group(s) a User ID is associated with by adding (up to 16) or removing groups from a list of a User's association with Group IDs.
- Change the User Description, i.e., change the long name identifying the User ID.
- Change User Memory. Change the maximum amount of **c-tree Server** memory a given user can consume.
- Change Extended Logon Validation, including start date, end date, and consecutive logon failures.

Group Definitions

Group definition operations available in *ctadmn* are as follows:

- Add New Group by:
  - Entering a new Group ID.
  - (optional) Entering a long name (i.e., a text string) for use as a Group Description (e.g., to display, where the Group ID may be too terse).

  - (optional) Entering a Group Memory Allocation, which is a maximum memory allocation for Users who are a member of this particular Group, and will override maximum memory allocations set at the Server-level (for any user).

- Remove an Existing Group ID.
- List Groups: list all current Group IDs and descriptions.
- Change Group Membership: Add or remove User IDs from a given Group.
- Change Group Description: change the long name for the Group ID.
- Change Group Memory: change the maximum amount of Server memory user in a given group can consume.

File Security

File security operations that can be performed on a given file using *ctadmn* are as follows:

- Change the File's Password.
- Change the File's Permission Mask, controlling file operation permissions for three classes of users: World, Group, and File Owner.
- Change the File's Group.
- Change the File's Owner.

NOTE: Applications can be designed so separate data files and/or index files can be joined into a "superfile," which is a single physical file from the point of view of the operating system. Separate "logical" files within a superfile are called superfile member files. From the point of view of the Administrator, superfiles, member files, and separate data or index files are all treated the same way.

Monitor Clients

The Administrator may want to know which users are currently attached to the **c-tree Server** or he/she may want to force a user to disconnect from the **c-tree Server**. These functions are available:

- List Attached Users.
- Disconnect Users.

NOTE: Users IDs are associated with Task users (i.e., sessions). It is a task, or session, that is actually terminated. If a User is disconnected using *ctadmn*, the *CTSTATUS.FCS* entry is augmented by the terminated user ID and node name.

Server Information and Server Configuration

These prompts provide performance and configuration information since the last startup of the **c-tree Server**.

Stop Server

This prompt allows the Administrator to stop the **c-tree Server**. *ctadmn* will ask for verification that the **c-tree Server** is to be stopped and ask for a shutdown delay in seconds.

Informing Users of their Security Options

Users can change the password for their own User ID and they can change security controls for a file if they are the owner of the file. To optimize the use of the **c-tree Server**, you may wish to be sure users are aware of these powers, and how to use them. See Appendix A: Users' Control of Security Options for details.

4.3 sa_admin - The command line system administrator utility

The command line version of the system administrator program, *sa_admin*, can be used to perform operations from shell scripts.

```
sa_admin [ -aadminuserid ][ -padminpassword ][ -ffilepassword ]
        [ -sservername ][ -o..... see other options below]
```

ADMINISTRATOR OPTIONS

Note: There is no space between the switch and its parameter.

-a ... System administrator User ID.
-p ... System administrator password.
-f ... Optional server system file password.
-s ... Optional server name.

USER OPTIONS

The following options, all beginning with -ou, allow changes to user information. Additional group and file options are described below.

NOTE: To use any optional entry, you must use all the previous entries even if they would otherwise be optional. For example, to add a user with the –oua option and specify a group, you must also enter the *userid*, *desc*, and *password*.

Option User Add

```
-oua userid [desc [password [group [memory [rule [begdat
            [enddat [loglimit]]]]]]]]
```

userid	User id is mandatory.
desc	Optional user description.
password	Optional user password.
group	Optional user group.
memory	Optional user memory limit. This can be a number or "D" for default or left NULL for no limit.
rule	Optional user memory rule. Used only with memory. This may be "A" for Absolute. "G" for Guideline. "D" for Default. NULL for Default.
begdat	Optional starting validity date. Specify as mm/dd/yyyy. NULL for Default.
enddat	Optional ending validity date. Specify as mm/dd/yyyy. NULL for Default.
loglimit	Optional maximum invalid logon attempts. NULL for Default.

Option User Remove

```
-our userid
```

userid	User id is mandatory.

Option User List

```
-oul
```

Option User Change Password

```
-oup userid password
```

userid	User id is mandatory.
password	New password is mandatory.

Option User Add user to Group

```
-oug userid group
```

userid	User id is mandatory.
group	Group name is mandatory.

Option User (group) Extract - Remove a user from a group.

```
-oux userid group
```

userid	User id is mandatory.
group	Group name is mandatory.

Option User Change Description

```
-oud userid desc
```

userid	User id is mandatory.
desc	New user description.

Option User Memory

```
-oum userid memory rule
```

userid	User id is mandatory.
memory	New memory limit. This can be a number of bytes or "D" for default or left NULL for no limit.
rule	Optional user memory rule. Used only with memory. This may be "A" for Absolute, "G" for Guideline, "D" for Default, or NULL for Default.

Option User Change Extended Settings

```
-oue userid [begdat [enddat [loglimit [mstlogon
            [rsmlogon]]]]]
```

userid	User id is mandatory.
begdat	Optional starting validity date. Specify as mm/dd/yyyy. NULL for Default.
enddat	Optional ending validity date. Specify as mm/dd/yyyy. NULL for Default.
loglimit	Optional maximum invalid logon attempts. 0 for Default. -1 to disable invalid logon check.
mstlogon	Optional must logon period, e.g., how often the user must log on to remain active. Specify as number of minutes. NULL for Default. -1 to disable must logon period.
rsmlogon	Optional logon timeout remaining. If a user has been denied access to the **c-tree Server** due to excessive invalid logon attempts, you can adjust the remaining user lockout time here. Specify as number of minutes. NULL to leave unchanged.

Option User Show

```
-ous userid
```

userid	User id is mandatory.

GROUP OPTIONS

The following options, all beginning with -og, allow changes to group information. Additional user and file options are described elsewhere.

NOTE: To use any optional entry, you must use all the previous entries. For example, to specify a rule when adding a group with the –oga option, you must also enter the *desc* and *memory* options for the group.

Option Group Add

```
-oga groupid [desc [memory [rule]]]
```

groupid	Group id is mandatory.
desc	Optional group description.
memory	Optional user memory limit. This can be a number of bytes, "D" for default, or left NULL for no limit.
rule	Optional user memory rule. Used only with memory. This may be "A" for Absolute, "G" for Guideline, "D" for Default, or NULL for Default.

Option Group Remove

```
-ogr groupid
```

groupid	Group id is mandatory.

Option Groups List

```
-ogl
```

Option Group Change Description

```
-ogd groupid desc
```

groupid	Group id is mandatory.
desc	New group description.

Option Group Memory

```
-ogm groupid [memory [rule]]
```

groupid	Group id is mandatory.
memory	New memory limit. This can be a number of bytes or "D" for default or left NULL for no limit.
rule	Optional user memory rule. Used only with memory. This may be "A" for Absolute, "G" for Guideline, "D" for Default, or NULL for Default.

Option Group Show

```
-ogs groupid
```

groupid	Group id is mandatory.

FILE OPTIONS

The following options, all beginning with –of, allow changes to file information. Additional user and group options are described elsewhere.

Option File Password

```
-ofp filename password
```

filename	File name is mandatory.
password	File password is mandatory,

Option File Security (permissions)

```
-ofs filename permmask
```

filename	File name is mandatory.
filename	File permission mask. This field is interpreted as a 15-byte permission mask containing owner, group, and world permissions:

```
(offset)
0  1  2  3  4  5  6  7  8  9  10 11 12 13 14
----OWNER----  ----GROUP----  -----WORLD----
r  w  f  d  p  r  w  f  d  p   r  w  f  d  p
```

```
r = Read  w = Write  f = define  d = Delete  p = noPass
```

To set a permission, set the byte at the corresponding offset to a value of '+'.

To reset a specified permission, set the corresponding byte to '–'.

For example, the string "+++++-----+++++" sets all OWNER and WORLD permissions, and clears all GROUP permissions.

Option File Group

```
-ofg filename groupid
```

filename	File name is mandatory.
groupid	File group id is mandatory.

Option File Owner

```
-ofo filename owner
```

filename	File name is mandatory.
owner	File owner is mandatory.

5. Maintaining Database Integrity

Although this chapter is not required reading, FairCom recommends Adminstrators consider the benefits of the covered features. The **c-tree Server** creates a number of system files and logs to enable automatic or Administrator-guided recovery from problems. The Administrator is responsible for saving the relevant information, for recovering from problems and for returning database information to its state at an earlier point in time.

NOTE: The operations described in this section are performed only on files subject to transaction processing, with the only exception being those described in the "Dynamic Dump, Using A Script File" section.

5.1 c-tree Server Files

The **c-tree Server** creates special system files to maintain various kinds of information required to recover from problems. The following list details exactly what files are created, along with all required information needed by the System Administrator responsible for working with them. As the Administrator, be sure these files are backed up when appropriate and used for recovery when necessary.

NOTE: To be compatible with all operating systems, the names for all these files are upper case characters.

c-tree Server Status Log

When it starts up, and while running, the **c-tree Server** keeps track of critical information concerning the status of the **c-tree Server,** e.g., when it started; whether any error conditions have been detected; and whether it shuts down properly. This information is saved in chronological order in a text file, the **c-tree Server** Status log, *CTSTATUS.FCS*. To control the size of *CTSTATUS.FCS*, or to maintain inactive logs as T*.FCS files, use the `CTSTATUS_SIZE` keyword. See the keyword description for more detail.

Administrative Information Tables

The **c-tree Server** creates and uses the file *FAIRCOM.FCS* to record administrative information concerning users and user groups. This file can be encrypted with the `ADMIN_ENCRYPT` keyword. See Chapter 6 for details.

Transaction Management Files

The **c-tree Server** creates the following files for managing transaction processing:

D0000000.FCS	*I0000001.FCS*	*S0000000.FCS*
D0000001.FCS	*I0000002.FCS*	*S0000001.FCS*

NOTE: It is important to safeguard these files, however only the *S*.FCS* and *D0000001.FCS* files should remain after a normal Server shutdown.

Active Transaction Logs

Information concerning ongoing transactions is saved on a continual basis in a transaction log file. A chronological series of transaction log files is maintained during the operation of the **c-tree Server**. Transaction log files containing the actual transaction information are saved as standard files. They are given names in sequential order, starting with *L0000001.FCS* (which can be thought of as "active **c-tree Server** log, number 0000001) and counting up sequentially (i.e., the next log file is *L0000002.FCS*, etc.).

The **c-tree Server** saves up to four active logs at a given time. When there are already four active log files and another is created, the lowest numbered active log is either deleted or saved as an inactive transaction log file, depending on how the **c-tree Server** is configured (see inactive transaction logs).

Every new session begins with the **c-tree Server** checking the most recent transaction logs (i.e., the most recent 4 logs, which are always saved as "active" transaction logs) to see if any transactions need to be undone or redone. If so, these logs are used to perform an automatic recovery. When configuring the **c-tree Server**, the odd and even numbered logs can be written to different physical devices. See "Configuring the c-tree Server".

Inactive Transaction Logs

Transaction log files no longer active (i.e., they are not among the 4 most recent log files) are deleted by default. To save inactive transaction log files when new active log files are created, add the `KEEP_LOGS` configuration option to the Server configuration with a positive number indicating the number of logs to keep. In this case, an inactive log file is created from an active log file by renaming the old file, keeping the log number (e.g., L0000001) and changing the file's extension from "FCS" to "FCA." The Administrator may then safely move, delete, or copy the inactive, archived transaction log file.

Temporary Stream Files

The Server creates five stream files at startup. These files prevent errors when the operating system has used a large number of file handles and the Server needs a stream file. The file names begin with FREOPEN followed by a distinguishing character and ending with .FCS. These temporary files are used for internal Server operations and should automatically be deleted during a normal Server shutdown.

Optional Server System Event Log

The **c-tree Server** maintains two optional system files: *SYSLOGDT.FCS* and *SYSLOGIX.FCS*. *SYSLOGDT.FCS* is a **c-tree Plus** data file with a record for each recordable system event. Unlike the *CTSTATUS.FCS* file, the system log files can be encrypted so entries cannot be added, deleted, or modified with a simple text editor, and vendors can add application specific entries to the log. See Chapter 6 or your vendor's documentation for information on the `SYSLOG` keywords appropriate to your application.

In case of a system failure, be sure to save all the system files (i.e., the files ending with "FCS"). *CTSTATUS.FCS* may contain important information about the failure. When there is a system catastrophe, such as a power outage, there are two basic possibilities for recovery:

1. When the power goes back on, the system will use the existing information to recover automatically, or
2. The Administrator will need to use information saved in previous backups to recover (to the point of the backup) and restart operations.

5.2 Automatic Recovery

As described in the "Starting the c-tree Server" section, every time the **c-tree Server** starts it checks the current active transaction logs and determines if any transactions must be undone or redone. If any recovery operation is required, the **c-tree Server** does it automatically. The Administrator does not need to do anything. The **c-tree Server** displays messages indicating the beginning and the end of the recovery. When the recovery completes, the **c-tree Server** is ready to use.

5.3 Dynamic Dump

The purpose of any data backup is to protect data integrity. Periodically backing up application data allows the Server Administrator to recover from problems or to roll a database to its status at a specific point in time.

There are two ways to back up data controlled by a **c-tree Server**:

- Using a standard backup utility while the Server is shut down.
- Using the dynamic dump capability while the Server is operational.

FairCom recommends shutting down the Server periodically to allow a full backup. This has the advantage of simplicity, since all files can be backed up and restored without using the transaction logs to ensure the data and index files are syncronized. This is especially helpful for applications that do not use transaction control to maintain database integrity. The Administrator can simply restore the files and continue operation.

However, the dynamic dump provides the Administrator a safe, secure method of backing up data while the Server is operational. The Administrator can schedule a dump of specific files, which may be all files necessary for recovery or a subset of them. The dump runs while the Server is carrying on normal activity and processing transactions and is transparent to users.

The Server performs the dump at the first opportunity on or after the scheduled time. When beginning a scheduled dump, the **c-tree Server** temporarily stops new transactions from beginning and starts the actual dump as soon as all active transactions are complete or after aborting transactions that do not complete during a pre-set delay period. After the dump begins, there are no restrictions on transactions.

NOTE: The dynamic dump and recovery processes are intended primarily for files under transaction processing control. Non-transaction controlled files can be dumped under certain restrictions. See the "Dump Files Without Transaction Control" section for more information.

The following sections describe the following dump and recovery utilities:

Process	**Utility**	**Description**
Dynamic Dump	*ctdump*	Dumps data during Server operation.
Dump Recovery	*ctrdmp*	Restore files to state as of last dump.
Rollback	*ctrdmp*	Roll the database state to an earlier time following a dump recovery.
Roll Forward	*ctfdmp*	Roll the database state to a later time following a dump recovery.

ctdump - Dynamic Dump Utility

The user may pass a User ID, Password, Dump Script name, and Server name to this utility.

ctdump - Schedule a Dynamic Dump.

Syntax: `ctdump [adminuser adminpass] dumpscript [servername]`

`adminuser`	ADMIN group User ID
`adminpass`	Administrator password
`dumpscript`	Name of the Dynamic Dump script file on the Server system. A path relative to the Server system may be included.
`servername`	Optional Server name.

Example:

```
ctdump ADMIN ADMIN thescript FAIRCOMS
```

Script File for Defining Dynamic Dump

The dump script file is a plain text file that specifies when to perform the dump, at what interval to repeat the dump, and what files to include in the dump.

Format

The script file consists of a series of instructions, each of which is given by a keyword followed by a space and an argument (e.g., the keyword "`!DAY`" followed by the argument "`WED`"). All keywords begin with an "!" and are not case sensitive (i.e., `!DAY = !Day`). Arguments are strings of letters, numbers, and punctuation, in the format shown below for each keyword (e.g., `WED`). Spaces and new lines divide keywords and arguments. Spacing and new lines are free form, but it is easier to create and change a script with each keyword/argument pair on a separate line, as in the sample script shown after the list of keywords.

With the following two exceptions, the order of keywords does not matter:

1. The next to last keyword must be "`!FILES`", followed by an argument which is a list of the files to be dumped (with each file name separated by at least one space or line feed from the next), rather than a single value, and,
2. The last keyword in the script file must be "`!END`", which takes no argument.

Dynamic Dump Options

The keywords and arguments available for defining a dynamic dump are as follows:

!BLOCK_RETURN

Causes *ctdump* to wait until the dynamic dump is completed before returning. Without `BLOCK_RETURN`, *ctdump* returns as soon as the **c-tree Server** receives the dump request. By waiting for completion, `BLOCK_RETURN` permits the System Administrator to determine when the dump is completed. Developers may find it useful to alert a System Administrator when a backup is done.

!DATE <mm/dd/yyyy>	Date to perform dump. If the date has already passed, the !FREQ interval is applied to find the next scheduled time. If no !DATE or !DAY is specified, today is assumed.
!DAY <day of week>	Instead of a date, a day-of-week may be used to schedule the dump. They are specified as SUN, MON, TUE, WED, THR, FRI, or SAT. If no date, time or day-of-week is specified, then the dump is scheduled for immediate execution.
!DELAY <seconds>	Number of seconds to wait until aborting active transactions. If zero, the **c-tree Server** will not abort active transactions. The dump waits indefinitely until all active transactions have completed and no new transactions will be permitted to begin. If this delay value is greater than zero, the **c-tree Server** waits until either the delay has expired or all active transactions have completed. At this point, it begins the dynamic dump and permits new transactions to start up. If all transactions have not completed, the **c-tree Server** aborts those transactions still in progress, with one of two error messages: 78 (TABN_ERR), indicates the transaction has been abandoned. 162 (SGON_ERR), a generic error indicating a break in communication between the **c-tree Server** and the application.
!DUMP <dump file>	The name of the file or device into which all the data for all the dump files will be stored. **Caution:** If a file by this name already exists, it will be deleted at the beginning of the dump and the new dump file replaces the old file. **NOTE:** There must be sufficient space for the dump file, which is limited to the maximum file size for the operating system (2 GB on most systems). If enough space is not available, the dump fails. A failure due to insufficient disk space will not corrupt anything, but additional space must be allocated before a backup is completed.
!END	Terminates the instructions in the script file. Place !END immediately after the !FILES keyword and list of files. !END takes no argument.
!ENDSEGMENT	Terminates the list of segments when specifying individual segment size and location.
!EXT_SIZE <megabytes | NO>	Change the default extent (segment) size from 1GB or disable with NO. See the "Dump To Multiple Files – No Size Limit" section in this chapter for more details.
!FILES <list of files>	The !FILES keyword is followed by names of files to include in the dynamic dump. This must be the next to last keyword in the script file. We strongly suggest that *FAIRCOM.FCS* be included in your list. Members of a superfile cannot be individually "dumped." The entire superfile must be dumped; that is, the name of the host superfile, not a superfile member, is placed in the list of files. The * and ? wildcards are supported. See !RECURSE for other options.
!FREQ <hours>	Hours between successive dumps. For example, 24 to repeat the dump once a day, or 168 to repeat the dump once a week.

!PROTECT
!PROTECT_LOW

The keyword `!PROTECT`, without an argument, added to a dynamic dump script file suspends updates to each non-transaction file while it is being dumped. This ensures the file's data integrity. The associated index files for a data file are not guaranteed to be consistent with the data file because the files are not dumped at the same time. With transaction files, the files are consistent because transaction log entries are used to bring all files back to the same point in time, i.e., the effective dump time. In most situations it is more efficient to dump only the data files and rebuild to recreate the indices.

The update suspension is enforced only at the ISAM level unless the keyword `!PROTECT_LOW` is used instead. `!PROTECT` and `!PROTECT_LOW` are mutually exclusive options. The last one in the script is used.

Whether or not `!PROTECT` or `!PROTECT_LOW` are used, resource updates are suspended at the **AddResource**, **DeleteResource**, and **UpdateResource** entry points.

!RECURSE <YES | NO | MATCH_SUBDIR>

Default is NO. Controls directory recursion when using wildcards. The `!RECURSE` keyword only applies when processing a `!FILES` entry containing a wildcard. In the case of `MATCH_SUBDIR`, not only does the file name require a match on the wildcard pattern, but only directory names which match the pattern will be considered for recursion.

!SEGMENT

See details in the "Segmented Dynamic Dump" section of this chapter.

!TIME <hh:mm:ss>

Time of day, on a 24 hour clock, to perform back up. If the time has already passed, then the !FREQ interval is used to find the next scheduled time. If a !DATE or !DAY is specified without a time, then the time defaults to 23:59:59.

If no time, day, or date is specified the dump begins immediately.

Sample Script

```
!TIME   23:00:00!day    Sun
!Dump   SYSTEM.BAK
!Delay  600
!FREQ   168
!FILES  FAIRCOM.FCS
!END
```

This script schedules a weekly dump, at 11:00 PM on Sunday nights. The only file included in the dump is "*FAIRCOM.FCS*". The system will wait until 11:10 PM (i.e., 600 seconds delay, after the starting time of 11:00 PM) for all active transactions to complete and then it will abort any transactions still active and begin the actual dump. The dump data will be saved in the file named *SYSTEM.BAK*.

NOTE: The **c-tree Server** configuration file can also control the way lingering transactions are aborted.

Scheduling a Dynamic Dump Using the Script

There are two ways to schedule dynamic dumps:

1. The **c-tree Server** configuration file may be used to schedule dynamic dumps. In the configuration file, the keyword `DUMP` is followed by the name of the script file defining the dump.
2. Use the dynamic dump utility, *ctdump*, which is a separate utility for the Administrator to use while the Server is active.

To use *ctdump* for scheduling a Dynamic Dump use the following procedure:

1. While the **c-tree Server** is running, start the utility program *ctdump* as any other program in the environment.
2. Enter the password for ADMIN.
3. Enter the current **c-tree Server** name if the **c-tree Server** has been given a different name than the default name. See `SERVER_NAME` in "Configuring the c-tree Server" for more information.
4. Enter the name of the Dynamic Dump script file.

The **c-tree Server** confirms that it has scheduled the requested Dynamic Dump.

Mirrored files are supported during dynamic dump and dump recovery as follows:

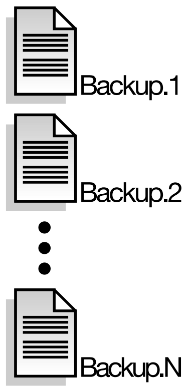
1. If a mirrored file should be opened for use by an application during a dynamic dump, the dump script should contain the "mirrored" name, i.e., the name with the vertical bar ('|'). For example, *sales.dat|msales.dat*.
2. If this is not done, and the dynamic dump opens the primary file, because it is not in use, a client opening the primary|mirror combination gets an `MNOT_ERR` (file already opened without mirror). To avoid blocking users from gaining file access, open primary files with their mirrors when specified for dynamic dumps.
3. The dump recovery program recreates both the primary and mirror files. It reproduces the primary file, and copies it over the mirror file.

Once a dynamic dump has been completed, the files may be used for Dump Recovery and/or Rollbacks.

Killing a Dynamic Dump

To kill a dynamic dump, simply execute *ctadmn* and list active clients. The dynamic dump will appear with the COMM PROTOCOL set to DYNAMIC DUMP. Now use the kill clients option to terminate the process. This allows a backup procedure to be canceled (killed) after it has been submitted to the **c-tree Server**.

Dump To Multiple Files - No Size Limit



The Dynamic Dump backup feature defaults to breaking-up the backup file (stream file) into multiple physical files (segments). This gets around individual file size limits imposed by the host OS (e.g., 2GB for a typical Unix system). Each backup file segment defaults to 1GB. There is no limit on the number of backup files (segments) supported.

Use the `!EXT_SIZE` keyword to change the segment size at runtime (up to 2000MB) by setting the argument of `!EXT_SIZE` to the desired number of bytes. Set the argument to NO to disable this feature and limit the dump to one file up to the OS maximum file size.

When a backup stream file is broken into segments, they are named as follows: original.001, original.002, etc, unless the original dump file has a name of the form name.nnn where nnn represent digits. For example, if the original dump file is named dump.str, then the first additional segment after dump.str gets to the extent size will be dump.str.001. However, if the original dump file is named dump.111, then the first extent will be dump.112.

On some systems, the dynamic dump extent names formed from the original dump stream file name by adding .001, .002, etc. are not legal. Therefore the extent name is first checked internally. If it does not work, the original dump stream file name is modified to produce a safe name in one of the following ways:

- Replace name extension, if any, in original with numeric name extensions (.001, .002, etc.);
- If the original name had no name extension, truncate the name to 8 bytes, and add numeric name extensions; or
- If the original name had no name extensions and is not more than 8 bytes, use the name FCSDDEXT.001 for the first dump stream segment, incrementing the numeric name extension as needed.

Segmented Dynamic Dump

The **c-tree Server** and the *ctdump* and *ctrdmp* utilities support dynamic dumping of segmented files and the creation of segmented (stream) dump files. This is different from the `!EXT_SIZE` feature that automatically breaks the dump file into 1GB 'extents'. Dumping to segmented files allows you to take advantage of huge file support and to specify the files size and location for each dump file segment.

- To dump segmented data or index files, simply list the host (main) file in the `!FILES` list and the segments will be managed automatically.
- To cause the output dump file produced by the dynamic dump itself to be segmented, use these script entries:

```
!SEGMENT      <size of host dump file in MB>
<dump seg name>     <size of segment in MB>
...
<dump seg name>     <size of segment in MB>
!ENDSEGMENT
```

The host dump file is the file specified in the usual `!DUMP` entry. Only the last segment in the `!SEGMENT` / `!ENDSEGMENT` list can have a zero size specified, which means unlimited size.

For example, assume *bigdata.dat* is a segmented file with segment names *bigdata.sg1*, *bigdata.sg2*, and *bigdata.sg3*, and the index file *bigdata.idx* is not segmented. To dump these files into a segmented dump file, use the script:

```
!DUMP  d:\bigdump.hst
!SEGMENT  50
    e:\bigdump.sg1  75
    f:\bigdump.sg2   0
!ENDSEGMENT
!FILES
    bigdata.dat
    bigdata.idx
!END
```

The host dump file is up to 50 MB on volume D:, the first dump file segment is up to 75 MB on volume E:, and the last dump file segment is as large as necessary on volume F:.

Wildcard Support for File Names

Dynamic dump backup and restore scripts specify the names of **c-tree Plus** data and index files that are to be backed up or restored, delimited by the `!FILES` and `!END` script keywords. It is possible to specify filenames using wildcards. Support for the typical asterisk (*) and question mark (?) wildcard symbols has been added.

In addition, the dynamic dump script keyword `!RECURSE` controls directory recursion when using wildcards. The `!RECURSE` keyword only applies when processing a `!FILES` entry containing a wildcard. Keep in mind that it is possible to specify standard file names and wildcard file names, one after the other, in the script. For example:

```
!RECURSE YES
!FILES
myfile.dat
cust*.dat
thedata.dat
*.idx
!END
```

There are three parameters for the `!RECURSE` keyword:

`!RECURSE  NO`	Do not recurse subdirectories.
`!RECURSE  YES`	Recurse underlying directories (max depth 32).
`!RECURSE  MATCH_SUBDIR`	File names and directory names must match.

In the case of `MATCH_SUBDIR`, not only does the file name require a match on the wildcard pattern, but only directory names which match the pattern will be considered for recursion.

The dynamic dump is specifically designed to address **c-tree Plus** data and index files, including superfiles. It does not support non-c-tree/server files. Please keep in mind: it is possible for your wildcard representation to represent non-c-tree files. The following definitions cause all files within the Server's `LOCAL_DIRECTORY` to be considered. If any non-c-tree files are encountered, the Dynamic Dump rejects them and a message is written to the *CTSTATUS.FCS* file if the `DIAGNOSTICS DYNDUMP_LOG` keyword is active. A rejection does NOT cause the dump to terminate. It will proceed to the next file.

```
!FILES *.*                          !FILES *
!END                                !END
```

Please remember that the dynamic dump does not support individual superfile member names. Specify the host superfile name in the script to back up the members. Here are examples of wildcard names:

- the pattern m*t matches mark.dat but does not match mark.dtx;
- the pattern *dat matches markdat and mark.dat;
- the pattern *.dat only matches mark.dat (not markdat).

Dump Files Without Transaction Control

It is possible to back up data files that are not under transaction control while the **c-tree Server** remains running. Of course, the safest way to perform a complete backup of data and index files while the **c-tree Server** remains running is to ensure that all your files are under transaction control. This way you are sure that all data and index files are completely synchronized, and updates to the files can continue during a dynamic dump.

Some developers choose not to implement transaction control for one reason or another. In some cases, developers migrating from the **c-tree Plus** Standalone Multi-user model, `FPUTFGET`, to the **c-tree Server**, choose to implement the **c-tree Server** in an `FPUTFGET`-like manner. An `FPUTFGET`-like Server is defined with the following **c-tree Server** keywords:

```
COMPATIBILITY    FORCE_WRITETHRU
COMPATIBILITY    WTHRU_UPDFLG
```

Although it is possible to define a non-transaction controlled file within a dynamic dump backup script, there is no protection against updates to this file. In other words, it is possible for the file to be updated during the dynamic dump. Updating a file controlled by transaction processing is okay, because the dump restore process can use the transaction logs to restore to a consistent state. However, updating files NOT under transaction control while they are being backed up presents a situation where there is no known state for the file, resulting in an undefined backup.

The keyword `!PROTECT`, without an argument, when added to a dynamic dump script file causes the non-transaction files to be dumped cleanly by suspending any updates while each file is dumped. At this point, the associated index files for a data file are not guaranteed to be consistent with the data file because the files are not dumped at the same time. Updates are only suspended while the data file is being backed up.

This technique ensures the data file is backed up in a known state. The restore process for a non-transaction control file MUST be complemented with an index rebuild. Because protection is for data files only, under most situations, the indices are not worth dumping since they must be rebuilt.

Note: `!PROTECT` suspends updates at the ISAM level only. The keyword `!PROTECT_LOW` also suspends low-level updates in addition to the ISAM level. FairCom suggests using the `!PROTECT_LOW` when using low-level function calls.

Dump Progress Messages Displayed in Function Monitor

Many times, especially during testing, watching the progress of a running dynamic dump can be beneficial. Adding the keyword `DIAGNOSTICS DYNDUMP_LOG` writes low-level progress messages to the *CTSTATUS.FCS* file. If the `FUNCTION_MONITOR YES` keyword is also active, dynamic dump progress information will also be written to the function monitor.

5.4 Dynamic Dump Recovery

In the event of a catastrophic system failure that renders the transaction logs or the actual data files unreadable, it will be necessary to use a dynamic dump or complete backup, to restore data to a consistent, well defined state. This is known as a dynamic dump recovery.

NOTE: If you make your own system backups when the **c-tree Server** is not in operation, and include the file *FAIRCOM.FCS*, you can restore from that backup in the event of a catastrophic failure.

Additional Keywords for Recovery Script

A Dynamic Recovery Script is used with the recovery utility. This can be the same script used to make the Dynamic Dump. The following keywords, with arguments in the same format as above, control the recovery process without having any effect on the dump itself:

`!COMMENT` Default: Off	Informs the **c-tree Server** that the remainder of the script file is for documentation purposes only and is not to be evaluated. Do not place keywords after this keyword.
`!DELETE` Default: `!SKIP`.	The opposite of `!SKIP`. It causes an existing file to be deleted and replaced by the recovered file.
`!#FCB <number of files>` Default: 100 files.	The desired number of file control blocks.
`!FORWARD_ROLL` Default: No forward roll is performed.	If planning to do a forward roll after a dump recovery, this keyword must be in the recovery script. The keyword is ignored during dynamic dump (backup). When present during dump recovery, this keyword causes a transaction start file to be restored with the archive file extension (i.e., S*.FCA). Be sure to rename the file from S*.FCA to S*.FCS before starting the forward roll. See the "System Recovery" section later in this chapter for more information on rolling forward.

`!PAGE_SIZE <bytes per buffer>` Default: 2048 bytes.	The number of bytes per buffer page, rounded down to a multiple of 128. **NOTE:** Required only if the **c-tree Server** configuration file changes the default page size.
`!REDIRECT <old path> <new path>`	Redirect output dumped from the old path into the new path. See "Define Alternative Restore Destinations" for more information.
`!SKIP` Default: `!SKIP`	Skip recovery for any file listed under the `!FILES` keyword if the file already exists. **NOTE:** Be aware of the differences between using `!SKIP` with a recovery and with a rollback (see System Rollback discussion) where it must be used with caution.

NOTE: Only the files specified by the `!FILES` keyword will be restored. It is not necessary to restore all files contained in the dump.

Running the Recovery Utility

The *ctrdmp* utility provides dynamic dump recovery. *ctrdmp* is a **c-tree Server** so there are some important points to keep in mind before running it:

1. Be sure the particular **c-tree Server** undergoing a recovery is not running when *ctrdmp* starts, since two **c-tree Servers** operating simultaneously interfere with each other.
2. Because it is a **c-tree Server**, *ctrdmp* generates temporary versions of all system files associated with a **c-tree Server** (i.e., files with the extension "FCS," as described above). Therefore, the dynamic dump file and the *ctrdmp* utility should be moved to a directory other than the working directory where the **c-tree Server** undergoing recovery resides. This is so the system files created by the recovery program will not overwrite working **c-tree Server** files. The temporary files are automatically deleted when Recovery completes successfully unless `!FORWARD_ROLL` is in the recovery script. In that case, the S*.FCS files are renamed to S*.FCA and kept in the directory.

After taking these preliminary steps, do the following to recover a dynamic dump:

1. Start *ctrdmp* the same way as any normal program in the environment.
2. When prompted, enter the name of the dynamic dump script file to be used for the recovery.

NOTE: The same script file used to perform the dump can be used to restore the dump. If a forward dump is planned, include the `!FORWARD_ROLL` keyword.

The dump recovery begins automatically and produces a series of messages reporting the progress of the recovery.

1. Each recovered, i.e., recreated, file will be listed as it is completed.
2. After all specified files have been recovered, a message indicates the recovery log, i.e., the transaction log, is being checked and recovered files were restored back to their state as of a given time, i.e., the time the dynamic dump started.
3. A message indicates the dump recovery process finished successfully.

Define Alternative Restore Destinations

By default, dynamic dump restore returns files to their original directory. There is also a convenient way to "redirect" the destination of the data during a dynamic dump restore. Developers may get a live "snap-shot" of a customer's database and restore it to an alternative destination for testing or for other purposes.

The dynamic dump script used during restore may contain one or more of the following redirection directives:

```
!REDIRECT  <old path> <new path>
```

Note: To specify an empty string for one of the `!REDIRECT` arguments use a pair of double quotes ("").

Examples: The following directives cause files that were backed up using absolute names to be restored into the directory `temp` (relative to the current directory during restore) and files that were backed up from the directory `local` (relative to the Server working directory) to be restored into the absolute directory `\temp\local`:

```
!REDIRECT \      temp\
!REDIRECT local\ \temp\local\
```

The following will add `temp\` to all restored files:

```
!REDIRECT ""     temp\
```

The following will strip `d:` from any restored files starting with d: (or D:):

```
!REDIRECT d:     ""
```

Note: The `!REDIRECT` keyword only affects the restore operation and is ignored when the script is used for the backup process.

5.5 System Rollback

System rollback restores the system to its status as of a specified point in time. For example, if company payroll processing was started at 1:00 PM and something went awry later in the afternoon, a system rollback can reset the system to the way it was at 1:00 PM, so processing could start again. If other applications using transaction processing files were running while the payroll processing was underway, these other files would also be rolled back to their 1:00 PM state. The Administrator should be aware of all files and related data that will be affected before starting a rollback to avoid interfering with multiple, unrelated systems sharing a **c-tree Server**.

Doing a rollback, like doing a recovery, involves running a program using a Dynamic Dump script using different keywords to control how the rollback is to be done.

Script File for Rollback

The format of the Rollback script is the same as the Dynamic Dump script discussed above. The rollback options are as follows:

`!COMMENT`	Informs the **c-tree Server** that the remainder of the script file is for documentation purposes only and is not to be evaluated. Do not place keywords after this keyword.
`!DATE <mm/dd/yyyy>`	The `!DATE` optionally specifies the target date, otherwise today's date is assumed.
`!#FCB <number of files>`	`!#FCB` overrides the default number of files.
`!PAGE_SIZE <bytes per buffer>`	Overrides the default buffer page size.
`!ROLLBACK`	`!ROLLBACK` must be the first entry in the script. It takes no argument. If `!ROLLBACK` is not the first entry, the script is interpreted as a Dynamic Dump script.
`!SKIP`	`!SKIP` takes no argument. If present, it does not cause an error termination if a file required during the rollback is not accessible to the program. Extreme care must be exercised if `!SKIP` is used, since the rollback has no way of ensuring the integrity of the resulting data.
`!TIME <hh:mm:ss>`	The `!TIME` keyword must be used to specify the targetted rollback time. The time is measured on a 24-hour clock. If the `!TIME` keyword is not in the script, *ctrdmp* fails.

Running the Rollback Utility

The *ctrdmp* utility performs a system rollback as well as dynamic dump recovery. *ctrdmp* checks the first keyword in the script file. If the first line is "`!ROLLBACK`" the script is used for a rollback. If it isn't, the script is considered a Dynamic Dump Script and used for a dump or a recovery.

Warning: As in dump recovery, be sure the particular **c-tree Server** undergoing the rollback is not running when *ctrdmp* starts, since *ctrdmp* is a **c-tree Server** and the two **c-tree Servers** operating simultaneously interfere with each other.

Do the following to complete a rollback:

1. Collect ctrdmp, the transaction log files covering the period from the target time to present, and the current log files into a working directory.
2. Start *ctrdmp* the same way as any program in the environment.
3. When prompted, enter the name of the rollback script file to be used.

The rollback will begin automatically and produce a series of messages reporting the progress of the recovery. You will be informed when the utility completes a successful rollback.

5.6 System Roll Forward

The forward dump utility, *ctfdmp*, can be used to recover from a catastrophic failure following the successful execution of a dynamic dump or from a full backup made after a safe, clean, controlled shutdown of the system.

Rolling Forward from Dynamic Dump

The *ctfdmp* utility is used to roll forward after a dynamic dump, and can be used only when the **c-tree Server** is stopped.

NOTE: When restoring from a dynamic dump, be sure the `!FORWARD_ROLL` keyword is in the recovery script. See Section 5.4.

Preparing for Using the Forward Dump Utility

To prepare for using the forward dump utility, *ctfdmp*, follow these guidelines:

1. Set the `KEEP_LOGS` configuration option to retain all log files. This setting causes log files no longer required for automatic recovery to be renamed instead of deleted. The extensions of log files are changed from "FCS" to "FCA", which changes the transaction log files from "active" to "inactive". These "old" log files may be needed to roll forward.
2. Be sure to make periodic, complete backups, using any standard backup utility. The following files must be included in a complete backup:
   - All data and index files.
   - The file named "*FAIRCOM.FCS*".
   - The S*.FCS files.

   NOTE: Once all necessary files have been backed up, the transaction log files (L*.FCS) may be deleted with one exception: DO NOT DELETE the most recent active transaction log file, which is the file of the form "L<seven digit number>.FCS" with the highest valued seven digit number.
3. After restarting the **c-tree Server** following a safe, complete backup, save all transaction log files created until the next complete backup. Active transaction log files have names of the form "L<seven digit number>.FCS," with the number being incremented by 1 for each new active transaction log. As specified in the `KEEP_LOGS` configuration value, when the **c-tree Server** creates a new active log it will rename the active log being replaced from "L<log number>.FCS" to L<log number>.FCA" and save it as an inactive transaction log file.

Running the Forward Dump Utility for System Recovery

If the system has a catastrophic failure and preparations have been made as recommended, the data can be recovered as follows:

1. Restore the contents of the most recent backup (dynamic dump or standard backup) provided it includes:
   - All the data and index files.
   - The file named "*FAIRCOM.FCS*".

   NOTE: If the restore is from a dynamic dump, be sure the `!FORWARD_ROLL` keyword is placed in the dump recovery script. This keyword causes creation of a transaction start file for the recovered logs. The transaction start file will be named S*.FCA. After the restore is complete, rename S*.FCA to S*.FCS.
2. Load all transaction log files saved between the time of that backup and the time of the catastrophic failure and rename all inactive transaction files in this group (i.e., log files with the extension "FCA") to give them the extension of an active transaction log file (i.e., extension "FCS").
3. Start the forward dump utility, *ctfdmp*, as any other program in the environment. The forward dump will proceed without any further instructions.

NOTE: Only transaction-processed files will be updated beyond the state they held when the backup was made.

ctfdmp accepts the command line arguments shown below. The first two need to be used only if the application uses more than the default number of `#FCB` or the `PAGE_SIZE` is larger than default, (see previous definitions for `#FCB` and `PAGE_SIZE`). If either of the first two command line arguments is used, they both must be specified as illustrated below. `!SKIP` is optional and described in the Rollback section above.

```
CTFDMP [!#FCB <number of files>]
       [!PAGE_SIZE <bytes per buffer page>]
       [!SKIP]
```

5.7 Transaction Log Dump

A transaction log dump is not something a Server Administrator typically needs to use, but we explain it here to be complete. Developers most often use this functionality as an aid to design, code, and debug an application being developed for use with a **c-tree Server**.

ctldmp is a utility providing a partial dump of transaction log files. This utility will attempt to create an ASCII log from the records in the transaction log and display it on the screen. It converts only the first 39 bytes of each record in the transaction log.

Options for Transaction Log Dump

The format of keywords for defining a transaction log dump is the same as the dynamic dump script file, but they are not put in a separate file or script. Instead, they are entered along with the name of the program when starting the program.

The keywords and arguments for *ctldmp*, the transaction log dump utility, are:

`DATE <mm/dd/yyyy>`	Begin dumping transactions as of the date specified. If no date is specified, begin dumping transactions from the beginning of the log file.
`LOG <number>`	Dump transactions beginning with the specified log. If no log number is specified, dump all log files meeting all other specifications.
`OFF`	Value of the position entry, offset, in the log dump that must be matched for the transaction to be listed. It is the 'P' field which follows the transaction number in the dump listing.
`POS`	Byte position in log to start the dump.
`TIME <hh:mm:ss>`	Begin dumping transaction as of the time specified. If a date is specified, then the date and time are used in conjunction with each other. If a date is not specified the current date is the default.
`TRAN`	Transaction number which must be matched in order for the transaction to be listed.
`TYPE`	Transaction type which must be matched in order for the transaction to be listed. The following code numbers correspond to the specified transaction type:

TYPE Value	Explanation
02	Add key value.
04	Delete key value.
07	Begin transaction.
08	Commit transaction.
09	Abort transaction.
12	New record image.
14	Old record image.
15	Difference of old/new record images.
26	Server checkpoint.
27	Open file.
28	Create file.
29	Delete file.
30	Close file.
31	Client login.
32	Client logoff.
34	End of log segment.
40	Abandon transaction.

`CHKPNT yes`	Outputs detailed information for each checkpoint encountered during a dump. For example, by using the following command line, the log dump begins with log file *L0000100.FCS*; only dumps checkpoints, and lists detailed information about checkpoints. tran types are found in *ctopt2.h* and the table above.

```
ctldmp log 100 type 26 chkpnt yes
```

Running a Transaction Log Dump

Warning: Like *ctrdmp*, *ctldmp* is itself a **c-tree Server**, therefore, the particular **c-tree Server** that generated the transaction logs being dumped by this utility should not be running while *ctldmp* is running.

Running a transaction log dump is a one-step process completed by starting *ctldmp* as any program in the environment, followed by up to three keyword/argument pairs specifying date, time and log number. *ctldmp* runs automatically, without prompting for any information, and informs you when it completes the transaction log dump.

5.8 Copying c-tree Server Controlled Files

WARNING: c-tree Server controlled files should only be copied, moved, or deleted when the **c-tree Server** is shut down. Copying, moving, or deleting files while the **c-tree Server** is in operation can lead to unpredictable errors.

When a file open is attempted, the **c-tree Server** checks if either a file with the same name is open, or if a file with the same unique ID is open. In either case, the match means a physical file open is not required. Instead, the open count for the file is incremented. The unique file ID permits different applications and/or client nodes to refer to the same file with different names, i.e., different drive or path mappings. However, if two different files have the same ID, problems arise because the second file will not actually be opened. The ID is constructed so that no two files could have the same ID unless someone copies one file on top of another.

When a file without a matching name matches the unique file ID, the **c-tree Server** will attempt to determine if they are actually different files. If so, it will automatically generate a new unique ID for the file. In either case, a message to the system console indicates the names of the two files involved. If this information is not critical to your operation, suppress this message by adding the following entry to the **c-tree Server** configuration file:

```
MONITOR_MASK    MATCH_FILE_ID
```

Server Unique File Detection - NetWork/Remote/UNC file names

As the **c-tree Server** manages file open/close operations for multiple users, it is critical for it to recognize uniqueness of a file. Different users can refer to the same physical file using different file names through aliases, relative paths, device mappings or SUBST commands. Internally, the **c-tree Server** has tests to determine if two files with different names are really different, or are actually the same physical file being accessed with different paths or alias names. If this internal unique file test is not accurate, the **c-tree Server** may attempt to open the files as two separate physically different files. This is a major problem since the **c-tree Server** is managing two separate caches for the same file.

In some cases, when confronted with file names mapped across a network or referred to in the UNC syntax, like "\\mymachine\c\mydata\myfile.dat", this internal unique file test incorrectly determines that two separate files were being addressed, when in actuality, the same file was being accessed. One user was using the physical name, while another was using the UNC name. This problem is somewhat uncommon, yet because of its serious consequences, the **c-tree Server** contains a number of protections.

`COMPATIBILITY NO_UNIQFILE`	This option disables attempts to determine if two files accessed with different file names (or paths) and which have identical **c-tree Plus** file IDs are actually the same or different files. This support is added in case our tests for uniqueness are somehow incomplete and lead to unintended file ID reassignments. These modifications give **c- tree Plus** the capability to disable the uniqueness test when files are suspected of having the same internal, "unique" 12-byte ID.
`COMPATIBILITY EXACT_FILE_NAMES`	This Server keyword mandates that all files have the exact same file name in order to be opened. If the internal name test determines that the files are in fact the same physical file, it will not allow the file to be opened with a different name. This compatibility keyword does not permit the same file to be opened with different names. If the same file is attempted to be opened with a different name, then error `EXCT_ERR`(642) will be returned.

There is a subtle interaction with the `NO_UNIQFILE` keyword defined above. The possible outcomes for all the combinations of keywords and files are in the table below. A file is represented by a lowercase path, an uppercase name, and a numeric file ID. For example, pA1 has path "p", name "A", and file ID 1. pA1 and qA1 are the same file accessed with different paths; pA1 and qB1 are different files mistakenly having the same file ID; and pA1 and qB2 are two different files with different file IDs (as expected).

The four possible keyword combinations are:

Standard	Neither `NO_UNIQFILE` nor `EXACT_FILE_NAMES`
NoUnique	Only `NO_UNIQFILE`
Exact	Only `EXACT_FILE_NAMES`
Both	Both `NO_UNIQFILE` and `EXACT_FILE_NAMES`

In the outcomes table below, the first (successful) open is for file pA1. The second open is as indicated. In actuality, only the second open for qB1 requires some adjustment or an error return.

Action/Return From 2nd Open Attempt

Second Open	Standard	NoUnique	Exact	Both
qA1	`NO_ERROR`	`NO_ERROR`	`EXCT_ERR`(642)	`EXCT_ERR`(642)
qB1	Modify B's file ID and return `NO_ERROR`	Incorrectly treat B as a shared open of A and return `NO_ERROR`	Modify B's file ID and return `NO_ERROR`	`EXCT_ERR`(642)
qB2	`NO_ERROR`	`NO_ERROR`	`NO_ERROR`	`NO_ERROR`

The uniqueness test (which is based on system dependent calls) may incorrectly indicate that two files are unique when they are the same. This occurs with certain mappings and/or aliases masking the sameness of the files. If this occurs, the first row of the above table becomes:

Second Open	Standard	NoUnique	Exact	Both
qA1	Incorrectly reassign file ID and have same file opened as two different files.	`NO_ERROR`	Incorrectly reassign file ID and have same file opened as two different files.	`EXCT_ERR`(642)

The most conservative approach is to turn on both keywords, but of course this requires the same name (and path) to be used for a file on all opens. If the uniqueness test is without weakness, then the standard setting (i.e., neither keyword) works best.

LOCAL FILE TEST: Because of the potential problems with network file names, and because FairCom does not recommend (actually discourages) placing **c-tree Server** files or logs on network drives (e.g. drives **not** on the local machine running the c-tree Server executable), a warning message is now written to the *CTSTATUS.FCS* file if a network file is detected. Besides the potential problem for the unique file test, placing data/index or server log files on a mounted network drive will introduce an additional network overhead and jeopardize the server's performance. The WARNING is only issued on the first such occurrence in order to avoid unnecessary overhead. If either of the `COMPATIBILITY` keywords is active, `NO_UNIQFILE` or `EXACT_FILE_NAMES`, the issue described above is not in play and the Server automatically disables this test.

Because of the possibility of a performance hit, the `COMPATIBILITY NO_TEST_LOCAL` keyword has been added to turn off the check of whether a file is local or remote.

6. Configuring the c-tree Server

The **c-tree Server** can be configured in many different ways, ranging from memory allocations and termination delays, to scripts controlling how dumps are made and how to recover previously dumped information.

Unless otherwise instructed, a **c-tree Server** starts using default settings for all configurable parameters. The **c-tree Server** takes configuration instructions from a configuration file, *ctsrvr.cfg*, placed in the **c-tree Server** directory (SYS:\ for NLM). When the **c-tree Server** finds a *ctsrvr.cfg* file, it uses all the specified configuration values.

NOTE: Your vendor may also provide a settings file which is not user-configurable, *ctsrvr.set*, described in the "Advanced – Configuration" section.

On Unix systems, keywords can be entered from the command line when starting the **c-tree Server** as described in the "Advanced – Configuration" section.

Examples of reasons the **c-tree Server** may need to be reconfigured are:

- Communications protocols, for transmitting information to and from the **c-tree Server**: The default communications support for the **c-tree Server** is TCP/IP. Implementing other communication techniques requires a **c-tree Server** configuration file, and the appropriate `COMM_PROTOCOL` keyword, as defined in the "Advanced Configuration Options" section.
- Memory allocations: To change the maximum amount of memory all users, or any given user, will be allocated -- and to specify whether this maximum is an absolute rule or only a guideline.
- Backup files: To specify the **c-tree Server** should look for a dynamic dump script and follow instructions in that script to back up specified files.

6.1 Running a Configuration Script File

To configure a **c-tree Server** with a configuration file:

1. Create an ASCII file that is a list of configuration parameters using the file format, configuration keywords and values described below.
2. Name this file *ctsrvr.cfg* and put it in the appropriate directory. By default, the **c-tree Server** looks in its own directory for a file of this name when it starts. **NOTE:** The c-tree Server for Novell looks in the root SYS:\ directory. The default file name and path can be changed with an environment variable and a command-line keyword, as described in the "Advanced – Configuration" section.
3. Start the **c-tree Server**.

NOTE: Before starting the **c-tree Server**, the Administrator should check the contents of the current configuration file, if one exists, to check its validity for the current situation. The Administrator is responsible for knowing the contents of any current configuration file, and for implementing any changes in the configuration of the **c-tree Server**.

The rest of this section is in three parts:

(1) The format of the **c-tree Server** configuration file is defined.

(2) Basic **c-tree Server** configuration options are listed and documented.

(3) Advanced **c-tree Server** configuration options are listed and documented along with other advanced topics.

6.2 Configuration File Format

The **c-tree Server** does not require a configuration file. If a configuration file is not present, the **c-tree Server** uses the keyword default values.

The format of the **c-tree Server** configuration file is as follows:

- File name: *ctsrvr.cfg*
- Location of file: Same directory as **c-tree Server**, or SYS:/ for NLM.
- File contents: An ASCII text file, consisting of a series of pairs of keyword names and keyword values, separated by one or more spaces (or a line feed). Keyword names are not case sensitive, but some values may be file names that may be case sensitive in certain environments, e.g., the value for `COMM_PROTOCOL`. Keyword values are strings of characters, without quotes around them, and without commas, decimal points, or indications of units (e.g., 1,000 bytes is entered as 1000). If a keyword is omitted from the configuration file, its default value is used.

For ease of reading and changing, we suggest using a format similar to the following configuration script file:

```
FILES        200
CONNECTIONS  15
IDX_MEMORY   500000
DAT_MEMORY   500000
```

This configuration file specifies the following changes from default settings:

- `FILES`: Increase the maximum number of files from 100 to 200.
- `CONNECTIONS`: Set the maximum number of concurrent connections to the **c-tree Server** to 15.
- `IDX_MEMORY` and `DAT_MEMORY`: Increase the memory allocated to index cache and data cache, from 225,000 bytes to 500,000 bytes.

Individual lines in *ctsrvr.cfg* can be commented out with a semi-colon ';' in front of a line. Example:

```
;COMM_PROTOCOL F_TCPIP
```

NOTE: Multiple configuration files may be used. For example, create different configuration files for different dynamic dump schedules, or for different communication protocols. Be sure the appropriate configuration file is found by the **c-tree Server** when starting.

6.3 Basic Configuration Options

The basic keywords are given below along with legal and default values of each configuration option. The keywords are sorted in alphabetical order.

`COMMENTS`
Default: Off

Informs the **c-tree Server** that the remainder of the *ctsrvr.cfg* file is for documentation purposes only and is not to be used. The Server ignores any remaining keywords. Comment individual lines by placing a semi-colon, ';', at the beginning of the line.

```
COMMENTS
```

`CONNECTIONS`
or
`USERS`
Default: Lower of 128 or activation limit.

The maximum number of connections to the **c-tree Server**. Typically, **c-tree Servers** are activated to support up to one of the following values for concurrent user connections: 8, 16, 32, 64, 128, 256, 512, or 1024. However, your particular **c-tree Server** may be customized with a different value for connections. Specifying a number of users greater than the actual number of users needed results in inefficiencies (e.g., unused memory), so the goal is to keep this number as low as feasible on the system. The Activation Key flier displays the allowable number of users for your **c-tree Server**, as does the **c-tree Server** startup screen.

```
CONNECTIONS <Number of Connections>
```

`DAT_MEMORY`
Default: 225000

The memory allocated to the data cache, specified in bytes. Within the memory constraints of the hardware, there is no effective limit.

```
DAT_MEMORY <bytes>
```

`DUMP`
Default: None

The name of a dynamic dump script file specifying when to begin and what to include in a dynamic dump. The contents of the script are described in the Dynamic Dump section of this manual. There is no default script, so the keyword DUMP does not appear in the default configuration file. An example configuration file entry for DUMP is:

```
DUMP system.dmp
```

NOTE: If the DUMP keyword is used, the file named as containing the dynamic dump script must be in the same directory as the **c-tree Server**. The **c-tree Server** will look only for this file in its own directory and if it does not find it, the **c-tree Server** will terminate immediately, with error #12 (i.e., "file not found").

`FILES`
Default: 100

The maximum number of data files and indices where each index, whether or not in a separate file, counts toward this total. For example, an index file which supports (i.e., contains as separate index members) three different keys counts as three files toward the `FILES` total. There is no effective limit to the number of files supported by the **c-tree Server**, except for any limits imposed by the available system memory.

```
FILES <Number of Files>
```

`IDX_MEMORY`
Default: 225000

The memory allocated to the index cache, specified in bytes. Within the memory constraints of the hardware, there is no effective limit. High-speed buffer search routines ensure quick access to the entire cache.

```
IDX_MEMORY <bytes>
```

MAX_DAT_KEY Default: 32	Maximum number of indices per data file. `MAX_DAT_KEY <Max Indices per Data File>`
MAX_KEY_SEG Default: 12	Maximum number of key segments allowed per index. `MAX_KEY_SEG <Max Segments per Index>`
SERVER_NAME Default: `FAIRCOMS`	A name assigned to **c-tree Server**, instead of the default name. `SERVER_NAME <NAME>`

6.4 Advanced Configuration Options

The **c-tree Server** Configuration File gives the Administrator much more control over the operation of the **c-tree Server** than covered so far. The following list of additional keywords, with explanations and default values for each, concludes the description of ways to customize the **c-tree Server**.

NOTE: Most of the concepts in this section (e.g., operating system characteristics, hardware configuration, network support) are intended for advanced users, the programmer, or the systems analyst in charge of the computing environment. We assume the Administrator will communicate with other experts as needed. For completeness, all configuration options supported by the **c-tree Server** are included in this manual. The keywords in this section are in alphabetical order.

The keywords in this section can be categorized into groups as follows:

Mirroring Control

ADMIN_MIRROR	SKIP_MISSING_LOG_MIRRORS
LOG_EVEN_MIRROR	SKIP_MISSING_MIRRORS
LOG_ODD_MIRROR	START_EVEN_MIRROR
MIRROR_DIRECTORY	START_ODD_MIRROR
MIRRORS	

Memory Control

BUFR_MEMORY	PRIME_INDEX
BUFFER_RUNLENGTH	RECOVER_FILES
GUEST_MEMORY	SORT_MEMORY
LIST_MEMORY	SPECIAL_CACHE_FILE
MPAGE_CACHE	SPECIAL_CACHE_PERCENT
NO_CACHE	TOT_MEMORY
NO_SHUTDOWN_FLUSH	USR_MEM_RULE
PRIME_CACHE	USR_MEMORY

Server Monitors/Diagnostics

CHECKPOINT_MONITOR	DIAGNOSTICS QUEUE_LOGON
DEAD_CLIENT_INTERVAL	DIAGNOSTICS TRACK_LOGON
DEADLOCK_MONITOR	DISK_FULL_LIMIT
DIAGNOSTICS DYNDUMP_LOG	FUNCTION_MONITOR
DIAGNOSTICS FILE_LOGON	LOCK_MONITOR
DIAGNOSTICS LOCK_DUMP	MEMORY_MONITOR
DIAGNOSTICS LOCK_LOGON	MEMORY_TRACK
DIAGNOSTICS NO_EXCEPTION_HANDLER	MONITOR_MASK

Transaction Control

CHECKPOINT_FLUSH	PREIMAGE_FILE
CHECKPOINT_IDLE	PREIMAGE_HASH
CHECKPOINT_INTERVAL	RECOVER_DETAILS
COMMIT_DELAY	RECOVER_MEMLOG
FORCE_LOGIDX	SKIP_MISSING_FILES
KEEP_LOGS	START_EVEN
LOG_EVEN	START_ODD
LOG_ODD	SUPPRESS_LOG_FLUSH
LOG_SPACE	TRANSACTION_FLUSH
PREIMAGE_DUMP	

Console Keywords

CTLR_C_ENABLE	NO_SHUTDOWN_PROMPT
NO_MESSAGEBOX	TOOL_TRAY
NO_PWRDWNPASSWORD	W9X_SERVICE

Miscellaneous Control

ADMIN_ENCRYPT	LOGON_FAIL_LIMIT
BROADCAST_DATA	LOGON_FAIL_TIME
BROADCAST_INTERVAL	LOGON_MUST_TIME
BROADCAST_PORT	MAX_VIRTUAL_FILES
CACHE_LINE	NODE_DELAY
COMM_PROTOCOL	NULL_STRING
CTSRVR_CFG	PAGE_SIZE
CTSTATUS_MASK	SEMAPHORE_BLK
CTSTATUS_SIZE	SERVER_DIRECTORY

DISK_FULL_LIMIT	SESSION_TIMEOUT
FIXED_LOG_SIZE	SIGNAL_DOWN
GUEST_LOGON	SIGNAL_READY
LOCAL_DIRECTORY	STARTUP_BLOCK_LOGONS
LOCK_HASH	TASKER_SLEEP
LOG_ENCRYPT	TMPNAME_PATH

Examples use the following format:

< input value >	An input value for the keyword. Replace with the value desired. For example, `COMMIT_DELAY <milliseconds>` would be put in the configuration file as `COMMIT_DELAY 10` to delay 10 milliseconds.
< value1 | value2 >	The | character indicates that either value1 **or** value2 can be used as described above, but not both together. For example, `ADMIN_ENCRYPT <YES	NO>` can be entered as either `ADMIN_ENCRYPT YES` or `ADMIN_ENCRYPT NO`.

ADMIN_ENCRYPT
Default: NO

To encrypt the *FAIRCOM.FCS* file at creation, set `ADMIN_ENCRYPT YES`. This prevents casual viewing of the data in the file, adding another level of security for passwords and user definitions stored in the file. This keyword will **not** encrypt an existing file.

```
ADMIN_ENCRYPT <YES | NO>
```

ADMIN_MIRROR
Default: No log mirror.

Permits *FAIRCOM.FCS* to be mirrored. For example, where *mirror_path* is the path to the secondary storage location for *FAIRCOM.FCS*:

```
ADMIN_MIRROR <mirror_path\FAIRCOM.FCS>
```

BROADCAST_DATA
Default: No data sent.

`BROADCAST_DATA` specifies a token to be broadcast following the Server Name. The token must not contain spaces. There is no default token. For example, add a department name or further identifying information for the **c-tree Server**.

```
BROADCAST_DATA <Token>
```

BROADCAST_INTERVAL
Default:10

The number of seconds between broadcasts. The default is 10 seconds, otherwise the token should be a number. If the number is negative, each broadcast is also sent to the **c-tree Server** standard output.

```
BROADCAST_INTERVAL <Seconds>
```

BROADCAST_PORT
Default: 5596

Specifies the TCP/IP port used for the broadcast. The default, 5596, is used when DEFAULT is specified, but any valid four-byte integer greater than 5000 that is not in use by another process may be specified. This should NOT be the port for the **c-tree Server**, which is displayed at startup and is based on the Server Name. See the examples in the "Broadcast" section in this chapter.

```
BROADCAST_PORT <DEFAULT | Port>
```

BUFFER_RUNLENGTH
Default: 10

This setting should be changed only at the request of your application developer. `BUFFER_RUNLENGTH` specifies the number of consecutive write operations performed while walking a list of buffer/cache pages before allowing other threads to acquire control of the list. A negative value is ignored.

```
BUFFER_RUNLENGTH <number of write operations>
```

BUFR_MEMORY
Default: 64000

Specifies the size of memory blocks the **c-tree Server** uses in conjunction with data and index cache buffers. To minimize interaction with the underlying system memory manager, the **c-tree Server** manages its own blocks of memory out of which the buffer pages are allocated. The **c-tree Server** acquires one large block of memory and allocates smaller pieces as needed. If you are attempting to limit memory use by reducing `IDX_MEMORY` and/or `DAT_MEMORY`, set `BUFR_MEMORY` to about one eighth of the smaller of `IDX_MEMORY` and `DAT_MEMORY`.

```
BUFR_MEMORY <bytes>
```

CACHE_LINE
Default: 16

To maximize the performance of the **c- tree Server** under **multi-CPU systems** ensure the cache line setting matches the setting for your equipment.

A cache-line is the smallest amount of memory a processor will retrieve and store in its highest speed cache. Typical cache-line sizes are 16, 32 or 64 bytes.

```
CACHE_LINE    <size>
```

CHECKPOINT_FLUSH
Default: 2

This keyword sets the maximum number of checkpoints to be written before a buffer (data or index) holding an image for a transaction controlled file will be flushed. The default value is 2. A value of zero causes the buffer to be flushed at least by the occurrence of the first checkpoint written after the buffer update. Reducing the value of this system parameter reduces the amount of buffering, slowing system performance, but decreases the amount of work to be performed during recovery.

```
CHECKPOINT_FLUSH <# of checkpoints>
```

CHECKPOINT_IDLE
Default: 300

Specifies the time in seconds between checkpoint checks. A checkpoint is an entry in the transaction log which lists open files, active transactions and transactions that are vulnerable due to pending buffer flushes. By default, every 300 seconds the **c-tree Server** checks if there has been any transaction activity, and if so, if there are any current active transactions. If there has been activity since the last checkpoint, but there is currently no active transaction, a checkpoint occurs. This strategy will not create extra checkpoints when the **c-tree Server** is idle, with respect to transactions, or when the **c-tree Server** is busy with transactions.

It is important to note that if an application routinely calls **Begin** whether or not updates are imminent, this "idle" checkpoint will be inhibited because there appears to be an active transaction. The purpose of this feature is to increase the likelihood of a clean checkpoint occurring in the transaction log, thus speeding automatic recovery. Ordinarily, checkpoints occur at predetermined intervals in the transaction log. A value of negative one (-1) will disable the idle checkpoint feature.

```
CHECKPOINT_IDLE <# of seconds | -1>
```

`CHECKPOINT_ INTERVAL` Default: 2MB	This keyword can speed up recovery at the expense of performance during updates. The interval between checkpoints is measured in bytes of log entries. It is ordinarily about one-third (1/3) the size of one of the active log files (L000....FCS). Reducing the interval speeds automatic recovery at the expense of performance during updates. The entry is interpreted as bytes if greater than 100 or as megabytes if less than 100. For example, `CHECKPOINT_INTERVAL 2` sets an approximate 2MB interval, while `CHECKPOINT_INTERVAL 150000` causes checkpoints about every 150,000 bytes of log file. `CHECKPOINT_INTERVAL  <interval in bytes or MB>`
`CHECKPOINT_MONITOR` Default: `NO`	This keyword takes three arguments: `YES`, `NO`, and `DETAIL`. If `YES`, each occurrence of an internal **c-tree Server** checkpoint will cause a time stamp message to be sent to the **c-tree Server** console screen and to the *CTSTATUS.FCS* file. The checkpoint is a snapshot of the **c-tree Server** at an instance in time and is used during automatic recovery. The checkpoint provides for a measure of the system activity. The `DETAIL` argument causes six intermediate milestones to be output for each checkpoint in addition to the beginning and ending checkpoint messages. These intermediate outputs aid in analyzing how the checkpoint procedure interacts with applications. If there is no system activity, no checkpoints will occur. This keyword should be used for debugging purposes only since performance may be compromised. `CHECKPOINT_MONITOR <YES | NO | DETAIL>`

`COMM_PROTOCOL`
Default:
Platform dependent. Most Servers load TCP/IP as the default.
	Specifies a communications module loaded by the Server. Some **c-tree Servers** support several protocols simultaneously, (i.e., Windows and Macintosh). For example, the **c-tree Server** could be communicating with users through a telephone line, others on a Novell network, and still others on an Ethernet connection. All that is needed is a separate "`COMM_PROTOCOL`" line in the configuration script for each communication module to be loaded by the **c-tree Server**. The following example loads TCP/IP, NetBIOS and IPX/SPX for a c-tree Server for Windows NT/2000/XP. See Chapter 2 for the communications options available for your platform:
	
	`COMM_PROTOCOL  F_TCPIP`
	`COMM_PROTOCOL  FNETBIOS`
	`COMM_PROTOCOL  FSPX`
	
	TCP/IP Encryption
	Encryption of c-tree's TCP/IP communication protocol allows an added level of security for client/server applications. FairCom's proprietary encryption algorithm disguises communication packets between the client and the **c-tree Server** making it difficult for a casual user to inspect the information being exchanged. Since FairCom's proprietary encryption algorithm is designed primarily for performance, the FETCPIP protocol is only marginally slower (10-20%).
	
	Use the Server keyword `COMM_PROTOCOL FETCPIP` in the Server configuration to specify encrypted TCP/IP communication. The **c-tree Server** simultaneously supports both encrypted and non-encrypted TCP/IP communication when both `COMM_PROTOCOL F_TCPIP` and `COMM_PROTOCOL FETCPIP` are specified in the Server configuration. Clients must be compiled with either encrypted or unencrypted TCP/IP. The Application Developer will specify the appropriate protocol.

Automatic LANA Support for c-tree Server for Windows

The c-tree Server for Windows offers automatic LANA support. When `COMM_PROTOCOL NETBIOS`, or `COMM_PROTOCOL NB`, is listed in the **c-tree Server** configuration file, the **c-tree Server** listens to all LANA ports. To specify a specific LANA number, use the following format:

```
COMM_PROTOCOL FNETBIOS@1
```

This disables automatic listen and sets a specific LANA number, LANA 1 in this example. Lana 0 is the default. Specifying a desired LANA number from a client side application uses the following format:

```
FAIRCOM2@1^NETBIOS
```

Identifying the c-tree Server host machine

Every protocol makes assumptions about the location of the machine hosting the **c-tree Server**. Use the following table to determine the proper method for the client to find the **c-tree Server** based on the protocol selected. `SERVER_NAME` is the name specified by the `SERVER_NAME` keyword, `FAIRCOMS` by default. ZoneName is the Mac zone hosting the **c-tree Server**. HostName is the network ID for the host machine. It can also be an IP address with TCP/IP.

Protocol	Default host	Specifying a host
ADSP	In local zone	`SERVER_NAME`@ZoneName
NetBIOS	On local network	Only default
Shared Memory	Local Machine	Only default
SPX	Default NLM Server	`SERVER_NAME` @HostName
TCP/IP	Localhost (127.0.0.1)	`SERVER_NAME` @HostName
UNXMSGS	Local Machine	Only default

COMMIT_DELAY
Default:
10 milliseconds, except –1 (disabled) on Windows NT. Keep this fact in mind when setting a time limit for aborting transactions.

Controls the length of time in milliseconds after a given transaction completes that the transaction manager waits before flushing the transaction to disk. By waiting, more than one transaction (i.e., the first one and all others that complete before the delay period expires) may be committed at the same time reducing disk-access overhead. On average, the longer the delay, the larger the number of transactions committed.

Warning: Increasing the delay time increases the chances of losing data in a catastrophe (i.e., power loss).

```
COMMIT_DELAY <milliseconds | -1>
```

CONSOLE CTRL_C_ENABLE
Default: Disabled.

The **c-tree Server** ignores <CTRL><C>. Adding the following keyword to the *ctsrvr.cfg* permits <CTRL><C> to stop the **c-tree Server**.

```
CONSOLE CTRL_C_ENABLE
```

CONSOLE NO_MESSAGEBOX
Default: Disabled.

When activated for Windows machines, this keyword deactivates error messages coming to the console in the form of a message box. The **c-tree Server** continues to log messages to *CTSTATUS.FCS*.

```
CONSOLE NO_MESSAGEBOX
```

CONSOLE NO_PWRDWNPASSWORD
Default: Request prompts.

Bypass the `ADMIN` group User ID and password validation typically required at shutdown. When this option is active and a machine shutdown or restart occurs, the prompt is bypassed and the **c-tree Server** shuts down cleanly.

```
CONSOLE NO_PWRDWNPASSWORD
```

CONSOLE NO_SHUTDOWN_PROMPT
Default: Show prompt.

To prevent an accidental unload, the c-tree Server for Novell prompts the console to acknowledge the shutdown. The number of active connections is displayed and the user has the option to proceed with the shutdown (unload) or allow the Server to continue running. Use the `CONSOLE NO_SHUTDOWN_PROMPT` keyword to disable this shutdown prompt.

```
CONSOLE NO_SHUTDOWN_PROMPT
```

CONSOLE TOOL_TRAY
Default: Disabled

Under Microsoft Windows, starts the **c-tree Server** in background, displaying only a c-tree icon in the Windows tool tray. This feature can also be activated with the ampersand, '&', character on the command line, for example: `C:>ctsrvr &`

```
CONSOLE TOOL_TRAY
```

CONSOLE W9X_SERVICE
Default: Disabled

Allows the **c-tree Server** for Windows 9x to remain in operation even if a user logs off Windows without shutting down.

```
CONSOLE W9X_SERVICE
```

CTSRVR_CFG
Default: Server Working directory

Specify the configuration file used when executing the **c-tree Server** from the command line.

```
CTSRVR_CFG <path and filename>
```

CTSTATUS_MASK
Default: Log all entries.

Allows certain types of entries in *CTSTATUS.FCS* to be suppressed. Currently, `VDP_ERROR` is the only valid argument, causing communication errors NOT to be logged to *CTSTATUS.FCS*.

```
CTSTATUS_MASK  VDP_ERROR
```

CTSTATUS_SIZE
Default: None, permitting unlimited size.

`CTSTATUS_SIZE` controls the size of the **c-tree Server** status file. The argument to `CTSTATUS_SIZE` is the approximate maximum size in bytes for *CTSTATUS.FCS*. When this limit is reached, *CTSTATUS.FCS* is renamed to *T0000001.FCS* and a new status file is created. The *T#.FCS* file numbers increase each time the limit is reached, similar to the transaction log files, i.e., the next time the maximum size is reached, *CTSTATUS.FCS* is renamed to *T0000002.FCS*.

To limit the number of archived status logs, set a negative value for `CTSTATUS_SIZE`. Only *T0000001.FCS* will be kept, being replaced each time *CTSTATUS.FCS* is archived.

A value of 0, the default, allows a the file to expand to a size limited by the operating system and storage availablity.

```
CTSTATUS_SIZE <file_size | negative_file_size | 0>
```

DEAD_CLIENT_INTERVAL
Default: 1800 (30 minutes)

`DEAD_CLIENT_INTERVAL` controls the timeout interval, the number of seconds of client idle time after which the Server checks the connection for that client. The default *nsec* value is 1800 (30 minutes), with a minimum of 120 (2 minutes).

NOTE: The timeout interval only controls how often the **c-tree Server** sends a message to test the connection. Different operating systems use different timeout values on TCP/IP messages, so the actual delay before a dead client is dropped will depend on when the operating system notifies the **c-tree Server** that the message failed.

```
DEAD_CLIENT_INTERVAL idle_time_seconds
```

DEADLOCK_MONITOR
Default: `NO`

If `YES`, each time the **c-tree Server** detects and resolves a dead lock situation, a time stamp message goes to the **c-tree Server** console screen. This keyword is used primarily for debugging since this feature consumes additional overhead.

```
DEADLOCK_MONITOR <YES | NO>
```

DISK_FULL_LIMIT
Default: No disk full check.

The **c-tree Servers** for Unix, Windows, and OS/2 support the `DISK_FULL_LIMIT` keyword, which activates a disk space threshold mechanism to detect when a disk volume is getting full. The `DISK_FULL_LIMIT` configuration keyword takes as its argument the number of bytes that must remain available on a disk volume after a file has been extended. If the update operation fails, a message is written in *CTSTATUS.FCS* naming the file involved.

c-tree Servers that do not support this feature ignore the `DISK_FULL_LIMIT` keyword.

```
DISK_FULL_LIMIT <bytes available>
```

DISK_FULL_VOLUME
Default: Off

Allows volume-specific disk full checks. `DISK_FULL_VOLUME` takes as its argument a concatenated volume name and limit. A path separator must occur between the volume name and the threshold limit, which may be zero.

In Unix this is in the form `/name/<limit>`. The following example places a disk full threshold of one million bytes on the volume `/home`:

```
DISK_FULL_VOLUME /home/1000000
```

In DOS, Windows, and OS/2 the form of the argument is `<DRIVE>:\<limit>`. The following example places a 1MB threshold on drive E:

```
DISK_FULL_VOLUME e:\1048576
```

FIXED_LOG_SIZE
Default: `NO`

Long variable-length data records can cause a problem for transaction logs because they must be rolled over so fast checkpoints are not properly issued. By default, the **c-tree Server** automatically adjusts the size of the log files to accommodate long records. As a rule of thumb, if the record length exceeds about one sixth of the individual log size (2.5MB by default), the size is proportionately increased. When this occurs, the *CTSTATUS.FCS* file receives a message with the approximate new aggregate disk space devoted to the log files.

`FIXED_LOG_SIZE YES` in *ctsrvr.cfg* disables this feature. When disabled, ensure transactions do not last longer than is necessary. If a transaction is begun and is still open when the log size is exceeded, the Server will terminate.

FORCE_LOGIDX
Default: NO

FORCE_LOGIDX allows LOGIDX support to be forced on, off, or disabled. ON forces all indices to use LOGIDX entries. OFF forces all indices not to use LOGIDX entries. NO uses existing file modes to control LOGIDX entries.

LOGIDX is an index file mode permitting faster index automatic recovery during **c-tree Server** startup. Transaction controlled indices with this file mode are recovered more quickly than with the standard transaction processing file mode TRNLOG. This feature can significantly reduce recovery times for large indices and has not noticeably degraded the speed of index operations. LOGIDX is only applicable if the file mode also includes TRNLOG.

NOTE: LOGIDX is intended for index files only, and is ignored in data files. When adding the LOGIDX file mode to an existing index, be sure to rebuild the index!

```
FORCE_LOGIDX <ON | OFF | NO>
```

FUNCTION_MONITOR
Default: NO

If YES, the client number, function number, function name, and file name are displayed in a scrolling fashion on the **c-tree Server** console screen. Alternatively, the same information, along with the return value and error codes for each function, can be routed to a file by specifying a file name. This keyword should be used primarily for debugging since this feature consumes additional overhead.

NOTE: Activate the function monitor dynamically under the **c-tree Server** for Windows by selecting View | Function Monitor Window.

```
FUNCTION_MONITOR <YES | NO | file_name>
```

GUEST_LOGON
Default: YES

When no user ID is sent to the **c-tree Server**, the client is automatically assigned a user ID of GUEST. The keyword GUEST_LOGON controls whether or not to permit GUEST logons. The keyword takes YES or NO for its arguments, and defaults to YES.

```
GUEST_LOGON <YES | NO>
```

GUEST_MEMORY
Default: 0

If greater than zero, this is the memory usage limit in bytes for a user without a User ID (i.e., a GUEST user).

```
GUEST_MEMORY <bytes>
```

IDLE_NONTRANFLUSH
IDLE_TRANFLUSH
Default: 15 seconds

The **c-tree Server** flushes data and index caches during **c-tree Server** idle time, launching two idle thread processes at start-up: One thread flushes transaction-file buffers and the other flushes non-transaction-file buffers. The threads wake-up periodically and check if the **c-tree Server** is idle to begin flushing. Subsequent **c-tree Server** activity terminates the flushes. Low priority background threads, such as the delete node thread, do not affect the **c-tree Server** idle state, but **c-tree Plus** clients and transaction checkpoints modify the idle status.

The default interval is 15 seconds. Setting the interval to zero or a negative value disables the thread.

```
IDLE_TRANFLUSH      <idle check interval (seconds)>
IDLE_NONTRANFLUSH   <idle check interval (seconds)>
```

`KEEP_LOGS`
Default: 0

If greater than zero, `KEEP_LOGS` specifies the number of non-active transaction log files kept on disk in addition to the active log files. When a greater-than-zero `KEEP_LOGS` value is exceeded, the **c-tree Server** automatically deletes the oldest inactive log file as new log files are needed. If `KEEP_LOGS` is zero, inactive log files are immediately deleted by the **c-tree Server**. If `KEEP_LOGS` is -1, no inactive log files are deleted by the **c-tree Server**.

`KEEP_LOGS` permits the archiving of transaction logs. Inactive log files may be safely moved, deleted, copied or renamed. An inactive log file which is not immediately deleted by the **c-tree Server** is renamed from the form L*.FCS to the form L*.FCA. The last character in the extension is changed from "S" to "A", with the rest staying the same.

```
KEEP_LOGS <number of inactive logs>
```

`LIST_MEMORY`
Default: 16384

Specifies the size of memory "chunks" the **c-tree Server** uses for various internal data structures. To conserve memory, set this value to 4096. In situations with large amounts of available memory, the value can be increased beyond the default.

```
LIST_MEMORY <bytes>
```

`LOCAL_DIRECTORY`
Default: Server's working directory

One of two mutually exclusive ways to supply the **c-tree Server** with the name of a directory path for processing files without absolute names. Absolute names include a specific volume or drive reference as part of the name, for example, d:\fairserv\data\. See `SERVER_DIRECTORY` for the other method. The trailing slash is required. If a `LOCAL_DIRECTORY` name is defined in the configuration script, the name will be attached to the beginning of any file name that is not absolute. If neither `LOCAL_DIRECTORY` nor `SERVER_DIRECTORY` is supplied, database and system files are stored relative to the **c-tree Server** working directory. `LOCAL_DIRECTORY` and `SERVER_DIRECTORY` cannot be used together.

NOTE: The `LOCAL_DIRECTORY` does not become a permanent part of the file name. The name entered into the transaction log does not include the `LOCAL_DIRECTORY`. `LOCAL_DIRECTORY` does not affect the location of the **c-tree Server** Status log, *CTSTATUS.FCS*.

```
LOCAL_DIRECTORY <Path>
```

`LOCK_HASH`
Default: 16

The number of hash bins available to the lock hash algorithm. A lock table exists holding all lock entries for each user. To search this table a hash algorithm is employed. The `LOCK_HASH` value specifies the number of 8 byte hash bins available for use by this algorithm. This value should only be increased with careful consideration. There is a marginally decreasing return for increasing values.

```
LOCK_HASH <Number of Hash Bins>
```

`LOCK_MONITOR`
Default:
No lock monitor.

Monitors the number of active record locks. This keyword, or the **SetOperationState** function call, enable a lock monitor that indicates when the number of lock calls over unlocks equals a multiple of the threshold value, or if it goes below a threshold.

For example, `LOCK_MONITOR 100` sends a message to the console each time the number of lock calls over unlocks equals a multiple of 100. Likewise, if the number of unbalanced lock calls falls below these thresholds, a message goes to the console.

Ordinarily, when the number of locks are in balance (i.e., excess locks over unlocks equals zero) no message is routed to the console unless a message indicating an excess of locks has already been sent to the console. If you wish the message to be sent whenever the number of excess locks equals zero, enter the threshold value as a negative. For example, `LOCK_MONITOR  -100`

```
LOCK_MONITOR <threshold value>
```

`LOG_ENCRYPT`
Default: `NO`

`LOG_ENCRYPT  YES` camouflages the contents of the transaction logs to prevent unauthorized access.

```
LOG_ENCRYPT YES
```

`LOG_EVEN`
Default: L

The alternative name for even numbered transaction log files. This name must be in the form of an optional directory path and the single character "L" (e.g., "D:\LOG0\L"). The **c-tree Server** appends a seven-digit even number and the extension "FCS" to the name given here. **NOTE:** The ability to give separate device and directory names for odd and even log files allows them to be directed to different physical storage devices.

```
LOG_EVEN <full_path>L
```

`LOG_EVEN_MIRROR`
Default: No log mirrors

The alternative name for even numbered secondary transaction log files. This keyword allows the even numbered secondary transaction log files to be mirrored to a location other than the primary transaction log files. This name must be in the form of an optional directory path and the single character "L" (e.g., "E:\LOG2EVEN\L"). The transaction management logic automatically appends a seven-digit even number and the extension "FCS" to the path provided.

```
LOG_EVEN_MIRROR <full_path>L
```

`LOG_ODD`
Default: L

The alternative name for odd numbered transaction log files. This name must be in the form of an optional directory path and the single character "L" (e.g., "D:\LOG1\L"). The **c-tree Server** appends a seven digit odd number and the extension "FCS" to the name provided.

```
LOG_ODD <full_path>L
```

`LOG_ODD_MIRROR`
Default: No log mirrors

The alternative name for odd numbered secondary transaction log files. This keyword allows the odd numbered secondary transaction log files to be mirrored to a different location than the primary transaction log files. This name must be in the form of an optional directory path and the single character "L" (e.g., "F:\LOG2ODD\L"). The transaction management logic automatically appends a seven-digit even number and the extension "FCS" to the path provided.

```
LOG_ODD_MIRROR <full_path>L
```

`LOG_SPACE`
Default: 10

This is the number of megabytes of disk space allocated to storing active transaction logs, starting with a minimum of 2. The **c-tree Server** maintains up to 4 active log files, which consume, in the aggregate, up to `LOG_SPACE` megabytes of disk space. Log files are numbered consecutively starting with 1. The log file names are in the form *L0000001.FCS*.

```
LOG_SPACE <Megabytes>
```

LOGON_FAIL_LIMIT Default: 0 (no limit)	The optional limit on the number of consecutive failed logons that causes subsequent logon attempts to fail for LOGON_FAIL_TIME minutes. A logon which fails during this period returns LRSM_ERR(584). `LOGON_FAIL_LIMIT <attempts>`
LOGON_FAIL_TIME Default: 5	The length of time logons are blocked after the logon limit is exceeded. A value of –1 indicates that there should be a permanent "log-on block" placed on a user until the Server ADMIN intervenes. `LOGON_FAIL_TIME <minutes>`
LOGON_MUST_TIME Default: 0 (no limit)	A non-zero value requires users to log on "at-least-once" within the defined time (e.g: at least once a week). If the time expires for a specific user, their profile will be deactivated, preventing access to the **c-tree Server**. The Server Administrator, or other ADMIN group user, must re-set the user's account once the time limit has elapsed. `LOGON_MUST_TIME <minutes>`
MAX_VIRTUAL_FILES Default: 500	An integer argument specifying the maximum number of virtual files that may be opened at one time. `MAX_VIRTUAL_FILES  <Maximum files>`
MEMORY_MONITOR Default: NO	Sends a message to the console whenever allocated memory exceeds the next memory threshold. The parameter specifies a size in bytes. For example, "MEMORY_MONITOR 500000" sends a message every time memory consumption exceeds the next 500,000 byte range of memory. The message is also sent when memory usage decreases for each absolute memory block. This keyword should be used primarily for debugging, as there is some additional overhead for this feature. `MEMORY_MONITOR <Bytes | NO>`
MEMORY_TRACK Default: 0 (indicates do not track)	Sends debug output to the console every time the net memory allocation count changes by a multiple of the threshold value. The count is the number of memory allocation requests. See also DIAGNOSTICS TRACK_LOGON. `MEMORY_TRACK <allocation threshold value>`
MIRROR_DIRECTORY Default: Server directory or LOCAL_DIRECTORY	Permits mirrored files WITHOUT AN ABSOLUTE PATH NAME to be placed in a specified mirror directory. This is analogous to LOCAL_DIRECTORY except that it only applies to the mirror in a primary|mirror pair. `MIRROR_DIRECTORY <directory name>`
MIRRORS Default: YES	Turns off all mirroring when set to NO. YES implies ordinary operation in which the filename determines whether or not file mirroring is in effect on a file-by-file basis. NO implies all mirror requests are ignored (including log files, administrative files, and all user mirrors). Set MIRRORS to NO only if there are strictly no plans to ever use file mirroring or during catastrophe recovery situations where the mirrored files may not be available due to a hardware problem. The absence of this keyword implies file mirrors are supported. `MIRRORS <YES | NO>`

MONITOR_MASK MATCH_FILE_ID
Default: Enabled

When a file open is attempted, the **c-tree Server** checks to see if either a file with the same name has already been opened, or if a file with the same unique ID has already been opened. By default, if a file without a matching name does match the unique file ID the **c-tree Server** sends a message to the system console indicating the names of the two files involved. Suppress this message by adding the following entry to the **c-tree Server** configuration file:

```
MONITOR_MASK MATCH_FILE_ID
```

MPAGE_CACHE
Default: 0

The c-tree data cache uses the following approach to cache data record images:

- If the data record fits entirely within one or two cache pages (`PAGE_SIZE` bytes per cache page), then the entire record is stored in the cache.
- If the data record image covers more than two cache pages, then the first and last segments of the record are store in the cache, but the middle portion is read from or written to the disk. These direct I/O's are efficient operations since they are aligned and are for a multiple of the cache page size.

The nature of this approach can be modified. Set `MPAGE_CACHE` to a value greater than zero, N, to store records falling within `N+2` cache pages entirely within the cache. The default value is zero, behaves as described above.

Note: Setting `MPAGE_CACHE` greater than zero does NOT ensure faster system operation. **It is more likely to be slower than faster.** It does cause more of a record to be in cache, but there is increased overhead managing each individual cache page. The cache pages for consecutive segments of a record (where a segment fills a cache page) are completely independent of each other. They are not stored in consecutive memory and I/O is performed separately for each cache page. This configuration option should only be used for special circumstances with careful, realistic testing.

Note: Even a record smaller than a single cache page may require two cache pages because the record position is generally not aligned on a cache page boundary.

NO_CACHE
Default: Cache files

In some cases, it might be beneficial to define that a certain file NOT be cached. For example, if a file contains very large variable length records (BLOBS), it might be more efficient to bypass the cache and rely solely on the operating systems cache support. The **c-tree Server** does not store the full variable length record in cache, but retains the first and last page of the variable length record. This prevents large blocks of data from consuming the cache and also alleviates the management of a large number of cache pages for any one particular record.

To disable cache for a given file, use the following server configuration keyword:

```
NO_CACHE   <data file name>
```

Note: `<data file name>` may include a wildcard pattern using '?' for a single character and '*' for zero or more characters. The Server Administrator can specify multiple `NO_CACHE` configuration entries.

Caching can only be turned off for entire superfiles (i.e., the superfile host), not for individual superfile members. Index files require the use of index buffer pages and must be cached.

NO_SHUTDOWN_FLUSH
Default: Flush at shutdown.

With very large (2GB+) caches, it may be possible for a file to never be written to disk during its entire life cycle. When the **c-tree Server** is shut down, it begins to flush files to disk. In the case of a “scratch” or “temp” file, the application vendor may not care if c-tree flushes this file to disk.

`NO_SHUTDOWN_FLUSH` skips file flushing during server shutdown. Note that `<file name>` may specify a wildcard pattern, with ‘?’ replacing a single character and ‘*’ replacing a group of characters. Non-transaction controlled files can be specified as shown below for this treatment, but **such a file will be corrupted after shutdown if file flushing was actually skipped**. Transaction-controlled files that are not flushed will simply lengthen automatic recovery times.

```
NO_SHUTDOWN_FLUSH   <file name>
```

NODE_DELAY
Default: 300

Reuses empty index nodes after `NODE_DELAY` seconds. This allows the **c-tree Server** to finish active searches in the index tree before removing the empty node.

```
NODE_DELAY <seconds>
```

PAGE_SIZE
Default: 2048

The number of bytes per buffer page (maximum 65536 bytes). This value is rounded down to a multiple of 128. To encourage compatibility across different **c-tree Plus** environments, we suggest not modifying the `PAGE_SIZE`. However, if performance is of concern, this value may be modified to suit the characteristics of the operating system. Generally, this is a matter to discuss with the application programmer.

```
PAGE_SIZE <bytes>
```

PREIMAGE_DUMP
Default: NO

When enabled, all `PREIMG` files, even those not in a dynamic dump are temporarily changed to `TRNLOG` files to be compatible with the upgraded transactions, and all transactions are automatically changed from `PREIMG` mode to `TRNLOG` mode during the dump. `PREIMG` files opened or created in the middle of the dump are also temporarily promoted from `PREIMG` files to `TRNLOG` files. All promoted files are restored to their `PREIMG` status at the conclusion of the dynamic dump.

The dynamic dump script file accepts a `!DELAY` parameter whose argument is the number of seconds to wait for a transaction to complete before aborting it in order to permit the start of a dynamic dump. When the `PREIMAGE_DUMP` facility is used, the `!DELAY` parameter is effectively ignored if a long `PREIMG` transaction begins prior to the dynamic dump. This means the dump will not begin until all current transactions complete. See Dynamic Dump and Section 6.7 for additional information.

```
PREIMAGE_DUMP <YES | NO>
```

PREIMAGE_FILE
Default: D

The alternative name for the file containing preimages swapped to disk. The format for this name is an optional directory path, which may include a Drive ID, followed by the single character 'D' (e.g., "E:\SWAP\D"). The **c-tree Server** appends a seven-digit number and the extension "FCS" to the name provided here.

```
PREIMAGE_FILE <Full_path>D
```

PREIMAGE_HASH
Default: 128

The number of hash bins available to the preimage hash algorithm. During a transaction, a preimage space is used to hold intermediate results pending an abort or a commit. To search this area a hashing algorithm is employed. The `PREIMAGE_HASH` value specifies the number of four byte bins available per user for use by this algorithm. This value should be increased only if a large volume of updates and/or additions per transaction (e.g., several thousand) is anticipated.

```
PREIMAGE_HASH <number of hash bins>
```

PRIME_CACHE
PRIME_INDEX
Default: No priming.

The **c-tree Server** optionally maintains a list of data files and the number of bytes of data cache to be primed at file open. When priming cache, the Server reads the specified number of bytes for the given data file into data cache when physically opening the data file.

Data files are added to the priming list with configuration entries of the form:

```
PRIME_CACHE   <data file name>#<bytes primed>
```

For example, the following keyword instructs the Server to read the first 100,000 bytes of data records from *customer.dat* into the data cache at file open:

```
PRIME_CACHE   customer.dat#100000
```

A dedicated thread performs cache priming, permitting the file open call to return without waiting for the priming to be accomplished.

Use `PRIME_CACHE` with the `SPECIAL_CACHE_FILE` keyword to load dedicated cache pages at file open.

To prime index files, use configuration entries of the form:

```
PRIME_INDEX  <idx file name>#<bytes primed>[#<member no>]
```

If the optional `<member no>` is omitted, all index members are primed. If an index member number is specified, only that member is primed. For example, the following keyword instructs the Server to read the first 100,000 bytes of index nodes in *customer.idx* into the index buffer space at file open:

```
PRIME_INDEX   customer.idx#100000
```

The nodes are read first for the host index, and then for each member until the entire index is primed or the specified number of bytes has been primed.

The following example restricts the priming to the first member (the index after the host index):

```
PRIME_INDEX   customer.idx#100000#1
```

The `<data file name>` or `<index file name>` can be a wild card specification using a '?' for a single character and a '*' for zero or more characters.

RECOVER_DETAILS
Default: `NO`

This keyword sends detailed information about the Server automatic recovery process to the server console. The time spent for each phase of automatic recovery in addition to the number of transactions processed for each phase is provided. This keyword adds minimal overhead to **c-tree Server** operations.

```
RECOVER_DETAILS <YES | NO>
```

RECOVER_FILES
Default: NO

RECOVER_FILES makes it possible to set separate limits on the number of files used during automatic recovery and regular operations. The reason automatic recovery may require more files than regular operations is that during recovery files opened stay open until the end of recovery. RECOVER_FILES takes as its argument the number of files to be used during recovery. If this is less than the number used during regular operation specified by the FILES keyword, the number of recovery files is set equal to the regular files and the keyword has no affect. If the number of recovery files is greater than the number of operational files, the number of files is adjusted downward at the end of automatic recovery freeing memory used by the additional control blocks, about 900 bytes per logical file.

```
RECOVER_FILES <number of files | NO>
```

RECOVER_MEMLOG
Default: NO

Loads one or more transaction logs into memory during automatic recovery to speed the recovery process. The argument for this keyword specifies the maximum number of memory logs loaded into memory during automatic recovery.

```
RECOVER_MEMLOG  <# of logs to load | NO>
```

SEMAPHORE_BLK
Default: 10

For Unix based systems only. This is the number of semaphores obtained at one time. These semaphores are used in the shared memory communication subsystem.

```
SEMAPHORE_BLK <number>
```

SERVER_DIRECTORY
Default: Server working directory

One of two mutually exclusive ways to supply the name of a directory path the **c-tree Server** uses when processing all files not having absolute names, (i.e., absolute names include a specific volume or drive reference as part of the name). For example, d:\fairserv\data\. See LOCAL_DIRECTORY for the other option. The trailing slash is required. If a SERVER_DIRECTORY name is defined in the configuration script, the name is attached to the beginning of any file name that is not absolute. If no SERVER_DIRECTORY or LOCAL_DIRECTORY name is supplied, all database and system files are stored relative to the **c-tree Server** working directory. SERVER_DIRECTORY and LOCAL_DIRECTORY cannot be used together. SERVER_DIRECTORY does not affect the location of the **c-tree Server** Status log, the transaction log files, or the start files.

NOTE: The SERVER_DIRECTORY, unlike LOCAL_DIRECTORY, becomes a part of the file name. The name entered into the transaction log includes the SERVER_DIRECTORY.

```
SERVER_DIRECTORY <path>
```

SESSION_TIMEOUT
Default: No timeout

The SESSION_TIMEOUT keyword instructs the Server to remove TCP/IP connections after the specified number of seconds has elapsed without activity. This option has been verified on Windows, Novell, Linux, Mac OS 9, and Mac OS X.

```
SESSION_TIMEOUT <seconds>
```

SIGNAL_DOWN
Default: OFF

Ability to launch an application when the **c-tree Server** comes down. This keyword takes as its argument the name of an executable that will be launched when the **c-tree Server** has been successfully terminated. Supported by c-tree Server for Unix and Windows.

```
SIGNAL_DOWN <executable>
```

SIGNAL_READY
Default: `OFF`

Ability to launch an application when the **c-tree Server** comes up. This keyword takes as its argument the name of an executable, which will be launched when the **c-tree Server** is ready (i.e., automatic recovery is completed). Supported by c-tree Server for Unix and Windows.

```
SIGNAL_READY <executable>
```

SKIP_MISSING_FILES
Default: `NO`

This keyword is available for special **c-tree Server** startup conditions. If a user file required by the **c-tree Server** during automatic recovery was deleted, an error 12 might be returned and the **c-tree Server** would not continue. By adding "`SKIP_MISSING_FILES`" to the default *ctsrvr.cfg* file, the error will be logged and the **c-tree Server** will successfully start up. However, "`SKIP_MISSING_FILES`" is not recommended as a permanent setting. Deleting files under transaction processing control adversely affects database integrity.

```
SKIP_MISSING_FILES <YES | NO>
```

SKIP_MISSING_LOG_MIRRORS
Default: `NO`

Accepts a `YES`/`NO` argument. With an argument of `YES`, the **c-tree Server** does NOT terminate automatic recovery if it cannot find a log mirror to be recovered along with the primary file.

```
SKIP_MISSING_LOG_MIRRORS <YES | NO>
```

SKIP_MISSING_MIRRORS
Default: `NO`

Accepts a `YES`/`NO` argument. With an argument of `YES`, the **c-tree Server** does NOT terminate automatic recovery if it cannot find a mirror to be recovered along with the primary file.

```
SKIP_MISSING_MIRRORS <YES | NO>
```

SORT_MEMORY
Default: 16000

Specifies the size of sort buffers used by the **c-tree Server**. To conserve memory, set this value to 8192 or 4096. If large amounts of memory are available, the value can be increased significantly beyond the default. This value must be less than the maximum segment size in segmented architectures.

```
SORT_MEMORY <bytes>
```

SPECIAL_CACHE_FILE
Default: None

Dedicates a specified amount of cache memory to a particular Extended data file. This allows the Server Administrator to specify files that are critical to maintain in cache memory at the expense of the "least-recently-used" (LRU) approach, where a new cache page replaces the LRU existing page.

The server configuration file specifies the amount of dedicated cache for specific data files with a configuration file entry in the form:

```
SPECIAL_CACHE_FILE  <datafilename>#<bytes to cache>
```

For example, the following keywords specify 100,000 bytes of dedicated cache for *customer.dat* and 400,000 bytes for *data\inventory.dat*:

```
SPECIAL_CACHE_FILE   customer.dat#100000
SPECIAL_CACHE_FILE   data\inventory.dat#400000
```

SPECIAL_CACHE_PERCENT
Default: 50%

To specify the percentage of the overall data cache space that may be dedicated to individual files, use a configuration entry of the form:

```
SPECIAL_CACHE_PERCENT   <percentage>
```

For example, the following keyword would permit up to 10% of the total data cache pages to be assigned to files on the special cache file list:

```
SPECIAL_CACHE_PERCENT    10
```

To disable any special cache, enter `-1` for the percentage. The percentage defaults to 50% and the maximum amount that can be specified with the keyword is 90%.

START_EVEN
Default: S

The alternative name for even numbered start file. The start file contains the location at which the automatic recovery routines begin to scan the transaction logs. There are two start files (numbered zero and one) to reduce the risk of losing the starting point for automatic recovery. This name must be in the form of a directory path and the single character "S" (e.g., "C:\START\S"). The **c-tree Server** appends a seven-digit even number and the extension "FCS" to the name provided.

```
START_EVEN <full_path>S
```

START_EVEN_MIRROR
Default:
No start mirrors

The alternative name for even numbered secondary start file. The start file contains the location at which the automatic recovery routines begin to scan the transaction logs. There are two start files (numbered zero and one) to reduce the risk of losing the starting point for automatic recovery. This name must be in the form of a directory path and the single character "S" (e.g., "C:\START\S"). The **c-tree Server** appends a seven-digit even number and the extension "FCS" to the name provided.

```
START_EVEN_MIRRORS <full_path>S
```

START_ODD
Default: S

The alternative name for odd numbered start file. The start file contains the location at which the automatic recovery routines begin to scan the transaction logs. There are two start files (numbered zero and one) to reduce the risk of losing the starting point for automatic recovery. This name must be in the form of a directory path and the single character "S" (e.g., "C:\START\S"). The **c-tree Server** appends a seven digit odd number and the extension "FCS" to the name provided.

```
START_ODD <full_path>S
```

START_ODD_MIRROR
Default:
No start mirrors

The alternative name for odd numbered secondary start file. The start file contains the location at which the automatic recovery routines begin to scan the transaction logs. There are two start files (numbered zero and one) to reduce the risk of losing the starting point for automatic recovery. This name must be in the form of a directory path and the single character "S" (e.g., "C:\START\S"). The **c-tree Server** appends a seven digit odd number and the extension "FCS" to the name provided.

```
START_ODD_MIRRORS <full_path>S
```

STARTUP_BLOCK_ LOGONS
Default: Allow logons.

`STARTUP_BLOCK_LOGONS` prevents non-`ADMIN` user logons when the server is started. Only users in the `ADMIN` group are allowed to logon.

This feature allows the server to start for `ADMIN` purposes before authorizing access by non-`ADMIN` users. This could include creating or adjusting files, adjusting security options, or any other operations that require a functioning Server but are more conveniently accomplished when users are not connected to the Server.

`SUPPRESS_LOG_FLUSH`
Default: `NO`

Causes transaction begin and commit operations to skip the flushing of the log file when its argument is `YES`. The default is `NO`. Suppressing the log flush makes it impossible to perform a proper automatic recovery. However, a dynamic dump will capture the necessary log information to restore `TRNLOG` files to a clean, consistent state. Using this keyword without the `PREIMAGE_DUMP` keyword is not recommended

By turning on `PREIMAGE_DUMP` and using `PREIMG` files, your system can run much faster than with full transaction processing, and still perform on-line dynamic dumps which will permit restoring files to the time of the dump in a clean, consistent state. However, it will NOT be possible to roll forward from the restored files because transaction log entries are not maintained outside of the dump process. See also `PREIMAGE_DUMP` and Section 6.7.

```
SUPPRESS_LOG_FLUSH <YES | NO>
```

`TASKER_SLEEP`
Default: 0

Reduces **c-tree Server** CPU activity level in non-preemptive environments, where the **c-tree Server** performs its own task switching, thus controlling when to put itself to sleep and when to wake up. The value of `TASKER_SLEEP` controls whether the **c-tree Server** puts itself to sleep and, if it does, when it will wake itself up. Setting this keyword to other than default will diminish the **c-tree Server** performance. This keyword is normally used in the following environments: SCO Unix, QNX, Apple A/UX, Interactive Unix and Motorola 88 OPEN.

Valid values are:

Value	Description
-1	Never sleep.
0	Relinquish control of the CPU if another process is ready to run; otherwise, resume processing
> 0	Sleep for that amount of time (in milliseconds).

```
TASKER_SLEEP <milliseconds | 0 | -1>
```

`TMPNAME_PATH`

Default: current Server directory.

The `TMPNAME_PATH` location becomes the default directory for temporary files. On startup, if a `TMPNAME_PATH` entry is encountered in *ctsrvr.cfg*, the **c-tree Server** tests the validity of the path. If there is an error, the **c-tree Server** terminates. Whether or not successful, the **c-tree Server** enters the path name in the *CTSTATUS.FCS* file.

```
TMPNAME_PATH <path>
```

`TOT_MEMORY`

Default: 0

If greater than zero, the total number of bytes the system will attempt to allocate for all uses (including index and data caches specified by the `IDX_MEMORY` and `DAT_MEMORY` keywords). If the system usage exceeds this level, attempts will be made to reduce discretionary allocations. If zero, no memory limit is imposed.

NOTE: Systems with virtual memory do not benefit from a memory limit unless you wish to limit the **c-tree Server** for other reasons. Systems with fixed memory may use a limit to keep the **c-tree Server** within a desired limit.

```
TOT_MEMORY <bytes>
```

TRANSACTION_FLUSH

Default: 100

This keywords provides control for the maximum number of updates to a buffer (data or index) before it is flushed. The buffer may well be flushed prior to this number of updates because of the LRU (Least Recently Used) scheme or because of the checkpoint limit. This system parameter affects only buffers holding images for transaction controlled files. Reducing this value reduces the amount of buffering, slowing system performance; but decreases the amount of work to be performed during recovery. A value of zero causes the buffer to be flushed upon update.

```
TRANSACTION_FLUSH <# of updates>
```

USR_MEM_RULE

Default: `GUIDELINE`

The system default rule for **c-tree Server** action when a user exceeds his/her memory limit. This rule is employed only if the System Administrator has not assigned a rule specifically to the user or the user's primary group.

Valid values are:

ABSOLUTE - the memory limit set for a user is to be applied as given, so no additional memory beyond the established limits will be allocated if it is requested.

GUIDELINE - the memory limit set for a user guides memory allocation as follows: allow the user to have requested memory beyond the limit set, if it is available, and when another user needs that memory, then it reduces the amount of memory used back down toward the guideline as soon as possible (e.g., by moving memory resident information to disk).

```
USR_MEM_RULE <GUIDELINE | ABSOLUTE>
```

USR_MEMORY

Default: 0

If greater than zero, this is the system default limit of memory (in bytes) available to each user. Zero means no user system default limit is imposed. The Administrator overrides this setting for a particular user by assigning a different value to the user or to the user's primary Group.

```
USR_MEMORY
```

6.5 Server System Event Log Keywords

The **c-tree Server** optionally maintains two system files: *SYSLOGDT.FCS* and *SYSLOGIX.FCS*. *SYSLOGDT.FCS* is a **c-tree Plus** data file with a record for each recordable system event. Unlike the *CTSTATUS.FCS* file, the system log files can be encrypted so that entries cannot be added, deleted, or modified with a simple text editor, and vendors can add application specific entries to the log.

The System Event Log contents are controlled by `SYSLOG` configuration keywords in *ctsrvr.cfg*, *ctsrvr.set*, or from the command line. They are entered as pairs in the form of: `SYSLOG <keyword>`. As many of these pairs as desired may be used at the direction of your vendor. The current `SYSLOG` keywords are:

`ADMIN_API`	Only allow users in the `ADMIN` group to use the **SystemLog** function to create vendor-defined entries in the log.
`CTSTATUS`	Log each entry to *CTSTATUS.FCS* in the System Event Log, except for those entries which occur before or after the system logging monitor is in operation.
`DELETE_FILE`	Log file deletes and restores.
`DISABLE_API`	Do not allow any calls to the **SystemLog** function for user defined entries.
`DYNAMIC_DUMP`	Log the beginning and end of dynamic dumps and a result for each file dumped.
`ENCRYPT`	Encrypt the `SYSLOG` files.
`USER_INFO`	Log all logons, logoffs, and changes to user logon profiles.
`NONE`	Used in a settings file to eliminate additional `SYSLOG` entries in a server configuration file.
`LOGFAIL_PURGE`	Causes an automatic purge of the oldest entries in the log if the system cannot add a record to *SYSLOGDT.FCS*. All the entries occurring on the oldest day are deleted unless there are only entries for the current day in which case no entries are purged. After a successful purge, an attempt is made to add the new entry that triggered the automatic purge. If this add succeeds, the system log operation continues in its usual fashion.
`LOGFAIL_CTSTATUS`	If there is no `LOGFAIL_PURGE` entry in the configuration file, or if the purge fails, the log entries will be rerouted to *CTSTATUS.FCS* if `LOGFAIL_CTSTATUS` is in the configuration file. This disables `SYSLOG CTSTATUS`; i.e, no more entries are made to the system log.
`LOGFAIL_TERMINATE`	If there is no automatic purge or it fails, and if there is no rerouting to *CTSTATUS.FCS*, either the system log will stop operation, or if `LOGFAIL_TERMINATE` is in the configuration file, the **c-tree Server** will shutdown. **USE `LOGFAIL_TERMINATE` WITH CAUTION!**

6.6 Server Memory Calculations

The **c-tree Server** startup memory requirements can be reasonably approximated with the following equation:

```
Base line Memory =
    Server EXE size + 1 MB +
    Communications Library size (if applicable) +
    Data cache (DAT_MEMORY) +
    Index buffer (IDX_MEMORY) +
    (900 bytes * Number of files (FILES)) +
    (325 bytes * Maximum number of connections (CONNECTIONS)) +
    (4 bytes * Maximum number of connections (CONNECTIONS)
             * Maximum number of connections (CONNECTIONS)) +
    (16000 bytes * Number of actual connections) +
    (256 bytes per connection per file actually opened))
```

Note the following points:

- DAT_MEMORY and IDX_MEMORY default to 225,000 bytes each, with a minimum of 600 * PAGE_SIZE.
- FILES defaults to 100.
- CONNECTIONS default to 128.
- IDX_MEMORY is the MAX of:
    - IDX_MEMORY or
    - 3 * CONNECTIONS * (PAGE_SIZE + 400), where PAGE_SIZE defaults to 2K.

The following locking/transaction processing related items should be considered when approximating the **c-tree Server** dynamic memory requirements:

- Each record locked consumes 24 bytes.

For transaction processing files only:

- Each data record written consumes (record length + 42) bytes.
- Each index operation consumes (key length + 42) bytes.

Note: Some operating systems offer virtual RAM (VRAM) which swaps data from memory to the hard drive. VRAM is usually automatically invoked if RAM gets full. Since data is being moved to and from the hard drive by VRAM, applications will often slow. If your system suddenly slows, examine this possible cause with your system administrator.

6.7 Advanced - Faster Auto-Recovery

The following keywords described in Section 6.4 reduce the **c-tree Server** disaster recovery time. Reducing the **c-tree Server** auto-recovery time is done at the expense of **c-tree Server** throughput during normal operation. **c-tree Server** auto-recovery is typically a vital consideration in time sensitive, mission critical applications, such as a PBX controller or embedded automation control application. If your application requires the fastest possible data access during normal operations, this section will not be of interest.

RECOVER_MEMLOG
`<# of logs>`
Default: 0

Loads one or more transaction logs into memory during automatic recovery to speed the recovery process. The argument specifies the maximum number of memory logs to load into memory during automatic recovery.

RECOVER_DETAILS
`<YES | NO>`
Default: `NO`

Sends detailed information about the automatic recovery process to the **c-tree Server** console. The time spent for each phase of automatic recovery in addition to the number of transactions processed for each phase is provided. This keyword adds minimal overhead.

CHECKPOINT_ INTERVAL
`<byte interval>`
Default: 2MB

Speed up recovery at the expense of performance during updates. The interval between checkpoints is measured in bytes of log entries. It is ordinarily about one-third, (1/3), the size of one of the active log files. Reducing the interval speeds automatic recovery at the expense of performance during updates.

For example, adding "`CHECKPOINT_INTERVAL 150000`" to the configuration file will cause checkpoints about every 150,000 bytes of log file.

CHECKPOINT_IDLE
Default: 300

Specifies the time in seconds between checkpoint checks. A checkpoint is an entry in the transaction log which lists open files, active transactions and transactions which are vulnerable due to pending buffer flushes. By default, every 300 seconds the **c-tree Server** checks if there has been any transaction activity, and if so, if there are any current active transactions. If there has been activity since the last checkpoint, but currently no active transaction, a checkpoint occurs. This strategy will not create extra checkpoints when the **c-tree Server** is idle, with respect to transactions, or when the **c-tree Server** is busy with transactions.

It is important to note that if an application routinely calls **Begin** whether or not updates are imminent, this "idle" checkpoint will be inhibited because appears to be an active transaction. The purpose of this feature is to increase the likelihood of a clean checkpoint occurring in the transaction log, thus speeding automatic recovery. Ordinarily, checkpoints occur at predetermined intervals in the transaction log. A value of negative one (-1) will disable the idle checkpoint feature.

```
CHECKPOINT_IDLE <# of seconds |-1>
```

FORCE_LOGIDX
Default: NO

Allows `LOGIDX` support to be forced on, off, or disabled. `ON` forces all indices to use `LOGIDX` entries. `OFF` forces all indices not to use `LOGIDX` entries. `NO` uses existing file modes to control `LOGIDX` entries.

`LOGIDX` is an index file mode permitting faster index automatic recovery during **c-tree Server** startup. Transaction controlled indices with this file mode are recovered more quickly than with the standard transaction processing file mode `TRNLOG`. This feature can significantly reduce recovery times for large indices and has not noticeably degraded the speed of index operations. `LOGIDX` is only applicable if the file mode also includes `TRNLOG`.

NOTE: `LOGIDX` is intended for index files only, and is ignored in data files. When adding the `LOGIDX` file mode to an existing index, be sure to rebuild the index!

```
FORCE_LOGIDX <ON | OFF | NO>
```

TRANSACTION_FLUSH
Default: 100

Controls the maximum number of updates to a data or index buffer before it is flushed. This system parameter affects only buffers holding images for transaction controlled files. Reducing this value reduces the amount of buffering, slowing system performance; but decreases the amount of work performed during recovery. Zero causes the buffer to be flushed upon update. The buffer may be flushed prior to this number of updates because of the LRU (Least Recently Used) scheme or because of the checkpoint limit.

```
TRANSACTION_FLUSH <# of updates>
```

CHECKPOINT_FLUSH
Default: 2

The maximum number of checkpoints to be written before a data or index buffer holding an image for a transaction controlled file is flushed. The default value is 2. A value of zero causes the buffer to be flushed at least by the first checkpoint written after the buffer update. Reducing the values of this system parameter reduces the amount of buffering, slowing system performance; but decreases the amount of work to be performed during recovery.

```
CHECKPOINT_FLUSH <# of checkpoints>
```

6.8 Advanced - Performance Optimization

The following items can optimize **c-tree Server** throughput and are intended for use by advanced users only. In addition, the suggestions in the "Optimizing Transaction Processing" section make disaster recovery difficult or impossible!

I/O caching

If the computer running the **c-tree Server** has sufficient memory and the size of the files controlled by the **c-tree Server** are relatively large, increasing the `DAT_MEMORY` and `IDX_MEMORY` could increase performance. In general, the larger the data and index cache sizes, the better the performance for heavy use situations. The **c-tree Servers** use a hashed caching algorithm, so there is no need for concern with having the cache sizes set too large.

Fastest Platform

A commonly asked question is which **c-tree Server** platform offers the fastest response times. The performance of the **c-tree Server** is largely dependant on the host hardware and the communication protocol. The faster the CPU and disk I/O subsystem, the faster the **c-tree Server** responds. The internal performance differences for the **c-tree Server** across platforms are negligible.

Base the decision for which hardware platform to choose for the **c-tree Server** on the optimum hardware specifications using the following order of priority:

Fastest Disk I/O Subsystem

Fastest CPU

Fastest and most supported RAM

Communication Protocol

c-tree Servers support many communication protocols in addition to many operating system/hardware combinations. Typically, the largest I/O bottleneck with the client/server model is the communications between the server and the clients. Choosing the best suited communication protocol for the database server can play a crucial role in the client side response times. Due to all the variables affecting response times, (record size, quantity of records, number of users, network traffic, speed of the network cards . . .), it is impossible to provide an absolute guideline for which protocol to use. The best way to determine the optimal protocol for a particular platform is to conduct time trials with the available protocols. However, as a rule, the platform's protocols will typically be the fastest.

Optimizing Transaction Processing - ADVANCED

This section is intended for advanced users only! The c-tree transaction processing logic offers three modes of operation:

1. No transaction control at all.
2. `PREIMG` - partial transaction control (atomicity only).
3. `TRNLOG` - full transaction control (atomicity and automatic recovery).

Note: Atomicity and automatic recovery are defined in the glossary in Appendix C.

Transaction Control Background

No transaction control With no transaction control defined for a data file, read/write access to the file will be very quick. However, no guarantee of data integrity will be available through atomicity or automatic recovery.

PREIMG If the `PREIMG` file mode is used, database access will still be fast, and some data integrity will be provided through atomicity. With atomicity only, (`PREIMG`), changes are made on an all or nothing basis, but no automatic recovery is provided.

TRNLOG If your application files have been setup with the `TRNLOG` file mode, all the benefits of transaction processing will be available, including atomicity and automatic recovery. Automatic recovery is available with `TRNLOG` only, because `TRNLOG` is the only mode where all changes to the database are written immediately to transaction logs. The presence of the transaction log (history of changes to the files) allows the Server to guarantee the integrity of these files in case of a catastrophic event, such as a power failure. Recovering files without a `TRNLOG` file mode from a catastrophic event will entail rebuilding the files. File rebuilding will not be able to recover data not flushed to disk prior to the catastrophic event.

Transaction Options

SUPPRESS_LOG_FLUSH Full transaction processing offers the best possible data integrity, but at some expense to performance. Using the `SUPPRESS_LOG_FLUSH` keyword reduces overhead with transaction processing log flushes, but at the expense of automatic recovery. This keyword is typically used only with the `PREIMAGE_DUMP` keyword described below.

PREIMAGE_DUMP Although automatic recovery is not available to `PREIMG` files, it is possible to perform periodic dynamic backups. By using the `PREIMAGE_DUMP` keyword, it is possible to promote `PREIMG` files to full `TRNLOG` files during the Server dynamic dump process (see Section 5.3). The promotion to a `TRNLOG` file means a full transaction log (history of the file changes) will be maintained during the dump process. This guarantees that any changes made to the files while the backup is occurring will be saved in these specially maintained transaction logs. The ability to dynamically backup the user data files somewhat minimizes the loss of the automatic recovery feature with this mode.

Transaction Log Flush Delay

Increasing the amount of time before the transaction logs are flushed can increase the number of transactions sent to disk with each flush. This activity is controlled with the `COMMIT_DELAY` keyword. See `COMMIT_DELAY` for additional information.

6.9 Advanced - Broadcast

It is possible for your vendor to create client applications that listen for an available **c-tree Server** without knowing the Server Name in advance. A **c-tree Server** can be configured to broadcast its Server Name and IP address over a TCP/IP port. With this method, it is possible for a client to detect the various **c-tree Servers** operating on the network and obtain their Server Names, including IP addresses. Your vendor will notify you if and when you should use these settings and what values should be used.

Three Server keywords support the Broadcast feature: `BROADCAST_PORT`, `BROADCAST_INTERVAL`, and `BROADCAST_DATA`. See the examples in Section 6.4, "Advanced Configuration Options".

`BROADCAST_PORT` specifies the TCP/IP port used for the broadcast. The default, 5596, is used when DEFAULT is specified, but any valid four-byte integer greater than 5000 that is not in use by another process may be specified. This should NOT be the port for the **c-tree Server**, which is displayed at startup and is based on the Server Name. See the examples below.

`BROADCAST_INTERVAL` determines the number of seconds between broadcasts. The default is 10 seconds, otherwise the token should be a number. If the number is negative, each broadcast is also sent to the **c-tree Server** standard output.

`BROADCAST_DATA` specifies a token to be broadcast following the Server Name. The token must not contain spaces. The Server Name will be followed by a vertical bar character, '|', which is followed by the token. There is no default token.

Using the following sample keywords and assuming the host IP address was 127.0.0.1, the **c-tree Server** broadcasts "`SAMPLE@127.0.0.1|FAIRCOM_SERVER`" on port 6329 every 90 seconds:

```
SERVER_NAME         SAMPLE
BROADCAST_PORT      6329
BROADCAST_INTERVAL  90
BROADCAST_DATA      FAIRCOM_SERVER
```

6.10 Advanced - Compatibility and Diagnostics Keywords

The following keywords should be used ONLY on the advice of your application developer. They can seriously alter the operation of the **c-tree Server**.

Compatibility Keywords

ABORT_ON_CLOSE	NO_COMMAND_LINE
ADMIN_STOP	NO_CONFIG_FILE
BLOCK_DDSFM_CREATE	NO_SHUTDOWN_DELAY
BLOCK_DDSFM_DELETE	NO_SPCMGT_QUEUE
EXACT_FILE_NAMES	NO_UNIQFILE
FORCE_WRITETHRU	NON_ADMIN_SHUTDOWN
NLM_DEFER_THREADSWITCH	TCPIP_CHECK_DEAD_CLIENTS
NLM_LONG_FILE_NAMES	WTHRU_UPDFLG
NO_BLOCK_KILL	

Diagnostics Keywords

DYNDUMP_LOG	NO_EXCEPTION_HANDLER
FILE_LOGON	QUEUE_LOGON
LOCK_DUMP	TRACK_LOGON
LOCK_LOGON	TRAP_COMM
LOWL_FILE_IO	WRITETHRU

COMPATIBILITY ABORT_ON_CLOSE
Default: Defer close.

Disable 'deferred close' capability. Force a transaction abort when a file altered in a transaction is closed before the transaction is committed.

```
COMPATIBILITY ABORT_ON_CLOSE
```

COMPATIBILITY ADMIN_STOP
Default: Only ADMIN group may stop the **c-tree Server**

If no users are logged on, any user can shutdown a **c-tree Server** unless COMPATIBILITY ADMIN_STOP is present.

```
COMPATIBILITY ADMIN_STOP
```

COMPATIBILITY BLOCK_DDSFM_CREATE BLOCK_DDSFM_DELETE
Default: Do not block creates and deletes.

c-tree Server can block adds and deletes of superfile members during the course of a dynamic dump using two server configuration COMPATIBILITY keywords:

BLOCK_DDSFM_CREATE blocks superfile member creation during a dynamic dump. Attempts to create a superfile member return DDCR_ERR(740) with this keyword activated.

BLOCK_DDSFM_DELETE blocks superfile member deletion during a dynamic dump. Attempts to remove a superfile member return DDDR_ERR(741) with this keyword activated.

IMPORTANT NOTE: An application may create or delete superfile members in a superfile host waiting to be dumped while the overall dump is going on. Once the dynamic dump begins dumping the superfile host, blocked operation cannot occur **until the end of the entire dump**, not just the end of the dump of the superfile host itself. Therefore, the last superfile host listed in the dump script file list will have the shortest blocking period.

COMPATIBILITY EXACT_FILE_NAMES
Default: Allow different names.

Force different references to the same file to use the same names. For example: C:\data\temp.dat and \data\temp.dat would be considered different even if they referred to the same file.

```
COMPATIBILITY EXACT_FILE_NAMES
```

COMPATIBILITY FORCE_WRITETHRU
Default: Not present.

Forces the automatic addition of the `WRITETHRU` mode to all files opened without the `TRNLOG` file mode.

```
COMPATIBILITY FORCE_WRITETHRU
```

COMPATIBILITY NLM_DEFER_ THREADSWITCH
Default: Defer

This option can improve the performance of the c-tree Server for Novell at the cost of decreased performance in other processes. Consult with your application developer and Novell system administrator to determine if this switch is appropriate for your system

```
COMPATIBILITY NLM_DEFER_THREADSWITCH
```

COMPATIBILITY NLM_LONG_FILENAMES
Default: Not supported.

Instructs the **c-tree Server** to use OS/2 namespace support, provided OS/2 namespace support is enabled on the working volume. If the keyword is not used, or if the volume does not support OS/2 namespace, long file names are not supported. FairCom recommends that when using long file name support all volumes provide OS/2 namespace support to prevent an error. This keyword is only required by the NLM c-tree Server and is ignored in all other versions.

```
COMPATIBILITY NLM_LONG_FILENAMES
```

COMPATIBILITY NO_BLOCK_KILL
Default: Allow kill.

Disable the `ADMIN` ability to kill currently connected clients.

```
COMPATIBILITY NO_BLOCK_KILL
```

COMPATIBILITY NO_SHUTDOWN_DELAY
Default: Wait for client.

Forces an instant shutdown without pause for client disconnect. Not valid on NLM.

```
COMPATIBILITY NO_SHUTDOWN_DELAY
```

COMPATIBILITY NO_SPCMGT_QUEUE
Default: Manage Superfile deleted space.

By default, the **c-tree Server** reclaims the space from deleted member files of a Superfile. A dedicated background thread performs the space reclamation. A permanent queue permits the space reclamation to be interrupted at Server shutdown, and resumed when the Server restarts. The Server file *D0000001.FCS* holds queue entries for the space reclamation of deleted superfile members.

To disable this feature, add the following keyword to the Server configuration.

```
COMPATIBILITY NO_SPCMGT_QUEUE
```

COMPATIBILITY NO_UNIQFILE
Default: Check file identity.

Disables attempts to determine if files accessed with different file names (or paths) and identical **c- tree Plus** file IDs are the same file or different files.

```
COMPATIBILITY NO_UNIQFILE
```

COMPATIBILITY NON_ADMIN_SHUTDOWN
Default: `ADMIN` group only may shutdown.

Allow non-ADMIN users to shut down the Server.

```
COMPATIBILITY NON_ADMIN_SHUTDOWN
```

COMPATIBILITY TCPIP_CHECK_DEAD_ CLIENTS
Default: No check.

The **c-tree Server** normally recognizes when a client disconnects. However, the Server relies on a chain of events controlled by the operating system in order to recognize the disconnection. The client computer must notify the Server host computer that the connection has been dropped. For example: When a user closes an application, the socket is closed by the operating system, which sends a message to the Server host machine. However, if the network connection is temporarily interrupted or if the client machine is powered down suddenly, this message is not sent and the Server host machine can't recognize that the client connection has dropped. See also `SESSION_TIMEOUT`.

With the `COMPATIBILITY TCPIP_CHECK_DEAD_CLIENTS` keyword in the Server configuration, the **c-tree Server** detects when a TCP/IP client has dropped. Every 120 seconds, the client connection socket is rechecked to ensure that it is still a valid communications channel. If a connection is found to be invalid, the Server terminates the connection. This functionality is not currently supported on the **c-tree Server** for the Mac, OS/2, or early Linux (kernel earlier than v2.036) platforms.

COMPATIBILITY WTHRU_UPDFLG
Default: Not present.

Disables the 'update' flag on files with the `WRITETHRU` file mode. If `COMPATIBILITY WTHRU_UPDFLG` is not in its configuration file and if non-transaction files are opened without `WRITETHRU` in the file mode, a warning is issued in *CTSTATUS.FCS* concerning the vulnerability to `FCRP_ERR(14)`.

```
COMPATIBILITY WTHRU_UPDFLG
```

DIAGNOSTICS DYNDUMP_LOG
Default: `OFF`

The `DIAGNOSTICS DYNDUMP_LOG` keyword generates dynamic dump status info, sending progress entries during each dynamic dump to *CTSTATUS.FCS*, including an entry for each file it attempts to dump.

```
DIAGNOSTICS DYNDUMP_LOG
```

DIAGNOSTICS FILE_LOGON
Default: `OFF`

When the `DIAGNOSTICS FILE_LOGON` keyword is added to the **c-tree Server** configuration file, upon each logon and logoff, four file counters are output: physical files open, logical files open, File Control Blocks (FCBs) in use, and FCBs available. The logical files count will be greater than physical files if superfiles are in use. FCBs in use count will be greater than logical files if index files contain additional index members. These values reflect counts generated by all applications using the **c-tree Server**.

```
DIAGNOSTICS FILE_LOGON
```

DIAGNOSTICS LOCK_DUMP
Default: Disabled.

Enable the use of the **LockDump** function.

```
DIAGNOSTICS LOCK_DUMP
```

DIAGNOSTICS LOCK_LOGON
Default: `OFF`

The `DIAGNOSTICS LOCK_LOGON` keyword displays the net lock count upon each **c-tree Server** logon and logoff. This count is system wide, not just for the process logging off or on and incurs very low overhead.

```
DIAGNOSTICS LOCK_LOGON
```

DIAGNOSTICS LOWL_FILE_IO
Default: Disabled

The `DIAGNOSTICS LOWL_FILE_IO` Server keyword logs low-level system errors into the Server status file, *CTSTATUS.FCS*. Although client applications have access to system errors through *sysiocod*, it is useful to see these errors logged on the Server side.

```
DIAGNOSTICS  LOWL_FILE_IO
```

DIAGNOSTICS NO_EXCEPTION_HANDLER
Default: Handler Enabled.

The c-tree Server for Windows includes a Win32 Exception Handler to take care of any error situations. This keyword disables the Exception Handler in case the addition of this error handler created any unexpected behavior. FairCom cannot foresee any situation where the keyword will be needed, but in the interest of safety, added a method for disabling this Exception check.

```
DIAGNOSTICS NO_EXCEPTION_HANDLER
```

DIAGNOSTICS QUEUE_LOGON
Default: `OFF`

The `DIAGNOSTICS QUEUE_LOGON` provides the current number of items in the **c-tree Server** queues. Three system wide counts are given for `QUEUE_LOGON.` These three counts, listed below, are preceded by the letters M, C and D, respectively. This option requires very low overhead.

1. The number of pending messages in c-tree Server monitors.
2. The number of pending messages in checkpoint queue.
3. The number of pending messages in the delete node.

```
DIAGNOSTICS QUEUE_LOGON
```

DIAGNOSTICS TRACK_LOGON
Default: `OFF`

The `DIAGNOSTICS TRACK_LOGON` option provides a net count of memory allocation requests. This count is system wide, not just the particular process logging off or logging on and requires very little overhead. See also `MEMORY_TRACK`.

```
DIAGNOSTICS TRACK_LOGON
```

DIAGNOSTICS TRAP_COMM

Default: Disabled

When activated, the `DIAGNOSTICS TRAP_COMM` keyword instructs the **c-tree Server** to log incoming communications packets to *TRAPCOMM.FCS* prior to execution. This log can be played back using the cttrap utility and a debug Server to observe the results of the client requests, allowing the developer to exactly duplicate and repeat client activities. The trap file, *TRAPCOMM.FCS*, is created in the Server directory by default. To prepend a path onto the trap file name (say to route it to a separate disk), add an entry of the form `DIAGNOSTIC_STR  <trap file path>`. For example, if `DIAGNOSTIC_STR  /bigdisk/` were in the configuration file, then the trap file would be */bigdisk/TRAPCOMM.FCS*.

```
DIAGNOSTICS TRAP_COMM
```

DIAGNOSTICS WRITETHRU

Default: Disabled

The **c-tree Server** writes a warning message to the *CTSTATUS.FCS* file if it detects a file that is not under transaction control and does not have `WRITETHRU` defined. This warning is to notify the developer that the file is being maintained in a vulnerable mode. Because of the overhead of writing this message to the log, and because FairCom does allow this "dangerous-cached-buffered-type" mode, the warning message is only issued once, for the first file detected. In other words, it simply tells the developer that there was "at-least-one" vulnerable file.

To help developers detect the file names for all vulnerable files, a `DIAGNOSTICS` keyword has been added. By placing the following in your *ctsrvr.cfg*, all file names will be listed in the *CTSTATUS.FCS* file:

```
DIAGNOSTICS  WRITETHRU
```

6.11 Advanced - Configuration

Settings File

Your vendor may supply a settings file, *ctsrvr.set*, that is not user configurable. Configuration options in *ctsrvr.set* are critical to the vendor's application, so it should not be deleted. It is possible for the vendor to set the **c-tree Server** to require *ctsrvr.set* for startup. Review your vendor's documentation for guidance on the settings file.

Environment Variables for Configuration

The **c-tree Server** supports environment variables specifying the location and file name of the Server Configuration and Settings files, `FCSRVR_CFG` and `FCSRVR_SET`. By default, the **c-tree Server** looks for the configuration and settings files named *ctsrvr.cfg* and *ctsrvr.set* in the **c-tree Server** working directory. This feature allows the Administrator to change the location and name of these two files.

The environment variables, `FCSRVR_CFG` and `FCSRVR_SET`, override the default names and locations when the **c-tree Server** is launched. The environment variable should contain a complete file name for the configuration file. For example, to direct the **c-tree Server** to use *.\work\my_config.001* as its configuration file on a Windows platform, define `FCSRVR_CFG` as:

```
set FCSRVR_CFG=.\work\my_config.001
```

Likewise, `FCSRVR_SET` overrides the name and location of the Server Settings file:

```
(Unix example)
set FCSRVR_SET=/usr/production/my_set.abc
export FCSRVR_SET
```

Server Command Line Parameters

The **c-tree Server** has been improved to accept configuration information from the command line in addition to the settings and configuration files. Configuration keywords and values listed as command line arguments take affect after a settings file, *ctsrvr.set*, if any, and before the standard configuration file, *ctsrvr.cfg*. A command line entry cannot override a settings file entry, and a configuration file entry cannot override a command line entry (or a settings entry). For more information on settings files (ctsrvr.set), see additional documentation included in these notes related to Server Configuration File Security.

All valid configuration file keywords are supported and may be listed on the command line followed by an appropriate value. No special switch symbols or syntax is required. Simply enter each keyword followed by a value as follows:

```
ctsrvr FUNCTION_MONITOR YES LOCAL_DIRECTORY C:\MYDATA\
```

To specify the name and location of your server configuration file, *ctsrvr.cfg*, when launching the **c-tree Server** from the command line, use the new command line keyword `CTSRVR_CFG` followed by a fully qualified configuration file name as follows:

```
ctsrvr CTSRVR_CFG C:\myServer\ctsrvr.cfg
```

or

```
ctsrvr CTSRVR_CFG  /usr/myserver/myinfo.cfg
```

The `CTSRVR_CFG` command line keyword is typically used when running two Servers on the same machine, described elsewhere.

NOTE: The `FCSRVR_CFG` environment variable supersedes the `CTSRVR_CFG` keyword, so the file specified in the environment variable will always be the file used.

Appendix A. User's Control of Security Options

Users, including the super-user `ADMIN`, can add or change their own password. The user who is owner of a file can change file security information for the file. The utilities for implementing user security options are described here.

The User's Password

The following steps are required for a user to change the password associated with their own User ID:

1. Run the utility program *ctpass* as any other program in the environment.
2. Enter your current User ID.
3. Enter the current password for your User ID, if you have one.
4. Continue by entering the current name of the **c-tree Server** (i.e., the default name or another name, supplied in the **c-tree Server** configuration file).
5. Now change your password by entering the new password.
6. To be sure to enter the new password, you may be asked to enter it twice before it will be accepted. If the same name is not entered both times, try again.

NOTE: Whenever input is requested, the user may enter a question mark (?) to receive HELP.

After the new password is entered and confirmed, a message saying your User ID password has been successfully updated will be displayed. After being updated successfully, the new password must be used with the User ID to log on to the **c-tree Server**.

NOTE: The Administrator can always change, or view, the password for any User ID allowed on the **c-tree Server**.

File Security Controls

The owner of a file can change the security information for their file, as follows:

1. Run the utility program *ctfile* the same way as any other program in the environment.
2. Enter current User ID.
3. Enter the current password for the User ID, if one has been assigned.
4. Continue by entering the **c-tree Server's** current name, which is either the default name or another name specified by the Server configuration.
5. Now give the name of the file whose security information is to change.
6. If the named file has a file password, the next step is to enter the password.

NOTE: Whenever input is requested, the user may enter a question mark (?) to receive HELP.

The file owner may change the following security options for their file:

- Change the file's password.
- Change the file's permission mask.
- Change the file's Group.
- Change the file's Owner (Caution: be careful of this! Once the owner has been changed, then the original owner may no longer use the utility, *ctfile*, to access the file and change security information).

NOTE: The Administrator can always use the Administration Utility to change, or view, the file security information for any file controlled by the **c-tree Server**.

Appendix B. Overview of The c-tree Server

This Appendix focuses on items helpful in understanding the benefits of the **c-tree Server**.

Traditional file and data management operations allowed application programs to interact with groups of files and information in those files. The focus was on controlling files, and data, directly from an application.

In response to new environments, developments in information management are rapidly transforming ways applications interact with files and data within files. For example, if files and user applications reside on physically separate systems, as in a LAN environment, and multiple users need access to the same data, the traditional methods of file and data management become cumbersome, unstable, or impossible to implement and maintain in a practical manner.

A common design solution to these new environments is to add a new element, a server, which coordinates the activities of separate applications and files with interacting needs and effects. The first widely available servers were file servers, which are software products (often running on machines dedicated to the server functions) that control the links between applications and files.

Another popular kind of server is the database server, a program coordinating much more sophisticated operations between application programs, files, data within files, and even between different database servers. When using a database server, an application does not interact directly with files, but with the database server. The **c-tree Server** is a sophisticated database server, which communicates with applications using c-tree's flexible low-level API (Application Programming Interface), fast ISAM (Indexed Sequential Access Method) API or both concurrently.

Finally, most **c-tree Servers** can be accessed by Windows based ODBC compliant applications by using the **c-tree ODBC Driver**. The multiple levels of file access offered by the **c-tree Server** is one of the many features setting the **c-tree Server** apart from other database servers.

Client/server Computing

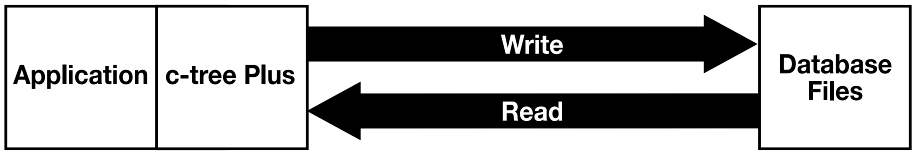
Client/server computing removes most of the difficult and tedious issues of database management from application programs and assigns these operations to a separate program, called a data server, which operates between the application program and its data.

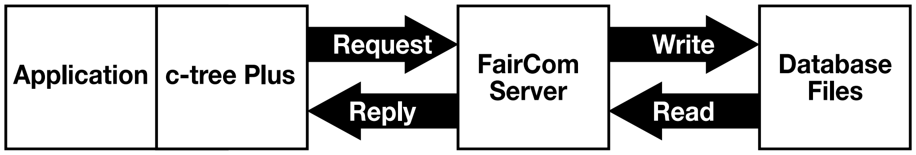
In client/server computing, application programs are clients making requests of a server, which goes to the relevant files, executes all operations needed to carry out the request, and sends back a response to the client application.

To implement this design, an application program needs to communicate with the database server. Exact details about which information-processing tasks are carried out by the application and which are carried out by the server depend on the server involved and on the "client-side" data management code used in the application.

The following diagrams illustrate how client/server computing is different from direct database manipulation.

In the "old" method of database operations, applications deal with files directly, using functions supplied by a third party (e.g., the **c-tree Plus** multi-user non-server library), or user supplied functions. All responsibility for security, coherence, and speed of access, in a single user or a multi-user environment, are the responsibility of the application code using data management functions.





In the "new" architecture, client/server computing, database operations function in a client application program interfaced to a database server. The database server contains the "intelligence" needed to process requests from clients, interact with the relevant database files, wherever they are located, and respond to those requests.

The database server in client/server computing plays a number of roles all of which add power to applications with a minimum of effort, including:

- Minimizes the flow of information from one place to another. By processing and responding to communication "request" and "response" messages between application programs and database files, the data server eliminates the need to send whole files of information from place to place. Only relevant data moves across the network.
- Manages multi-user issues. The Server manages requests so users don't get in one another's way, or create inconsistencies in the database.
- Coordinates sending and receiving of information over networks even where database files and/or applications reside on different types of machines and operating systems (heterogeneous support).
- Implements transaction processing (described below).
- Implements security features (described in the section on "Controlling Access to the Server").
- Offers absolute data integrity through file mirroring.

Client/server computing is more and more important for the most basic of reasons: it offers increased speed, control, and efficiency in data management.

Transaction Processing

Transaction processing extends client/server computing beyond the "request/response" model where a **c-tree Server** stands between an application and database files.

Transaction processing involves:

- Grouping together all individual actions involved in carrying out a task (e.g., all operations involved in taking an order for a product from a customer) as a transaction.
- Using the **c-tree Server** to carry out this group of actions, assuming either:
    1. All the actions will be done with all the effects of the actions reflected in the database files.
    2. The transaction will be terminated if there is any problem and none of the actions will be reflected in the database files.

The **c-tree Server** supports "all or nothing" (i.e., atomic) transaction processing. Applications using the **c-tree Server** can be designed to have transaction processing with:

- Fully logged transaction processing on given files.
- Transaction processing without logging.
- No transaction processing on less important files.

Heterogeneous Server Network Support

c-tree Servers automatically provide sophisticated network support allowing dissimilar machines to share data across the same network. The FairCom term for this type of logic is "Netformat". Netformat logic automatically controls all aspects of data byte ordering (big endian/little endian). The Server process defines the ordering of the data (High/Low (big endian) or Low/High (little endian)) while the client process dictates the alignment of the file.

Example: To further illustrate the power of the Netformat logic, review the following network scenario:

Server: c-tree RS/6000 TCP/IP Server

Clients: DOS Microsoft C word aligned client application, Macintosh System 7 Think C byte aligned client application, Windows Borland C byte aligned client application.

All data files will be stored in a High/Low (Most significant byte/Least significant byte) format used by the IBM RS/6000 CPU. Files created by the DOS application will be word aligned (the default with Microsoft C). Files created by the Macintosh and Windows Borland C application will be byte aligned. The three applications will all be able to share the same files (assuming the application developer has aligned all numeric fields on at least a 2-byte boundary for this example - a good C programming practice).

File Mirroring

The FairCom mirroring facility makes it possible to create and maintain copies of important files on different drive volumes, partitions or physical drives. If the primary storage location is lost due to some form of catastrophe (i.e., head crash) the mirroring logic can automatically detect the lost connection and switch to the secondary or "mirrored" storage area.

To mirror a file, the **c-tree Plus** file name must be modified by appending a vertical bar ('|') followed by the mirror name. For example, to mirror the file "customer.dat" to the file "mirror.dat", the file name would appear as follows:

```
"customer.dat|mirror.dat"
```

The mirrored file can be automatically created at file creation time by using the "primary_name|mirror_name" string whenever an ordinary **c-tree Plus** file name is supplied to the **c-tree Plus** create routine (e.g., in parameter files, IFIL structures, or in the file name string for low level creates).

Mirroring is available for client/server and single-user operations. It applies to all **c-tree Plus** file modes including transaction processed files.

By default, read and write operations on mirrored files will continue without returning an error if one of the files fail, but the other succeeds. When this happens, the failed file is shut down, all subsequent I/O operations continue only with the remaining file and the file name for the shut down file is logged in *CTSTATUS.FCS*.

When a primary and mirror file get out-of-sync, beyond the ability of automatic recovery to make them both good, the most up-to-date file is recovered.

Encryption

The **c-tree Server** supports encryption of data, index and transaction log files. This technology provides the means to add an extra level of confidentiality to an application's data. Once encrypted, it becomes impossible for a casual user to "dump" or "inspect" the data. Only your application vendor can add this functionality.

Appendix C. Glossary

This Appendix provides definitions for some of the terms found in this guide. Most terms are discussed in further detail throughout the rest of the guide. The definitions considered for the advanced user are depicted by listing "(advanced)" at the beginning of the definition.

Administrator	Individual typically responsible for installing, configuring, starting, stopping and maintaining the database Server and the files controlled by the **c-tree Server**.
AppleTalk	A communication protocol developed by Apple Computer Co. for the Apple Macintosh platform. The c-tree Server for Macintosh supports this protocol.
ASCII Text File	"ASCII" is an industry code for representing characters as binary values. An ASCII Text File is a special type of file that can be copied from computer to computer and read by most word processors and editors. If this file type is unfamiliar, consult the word processor or editor documentation for additional information.
atomicity	(advanced) A term meaning an all or nothing criteria applied to data inserts, deletes and updates; with the principal goal of keeping a group of files synchronized. For example, if a record is to be updated in a series of five files as follows: enable transaction file 1 update - successful file 2 update - successful file 3 update - error, not updated file 4 update - successful file 5 update - successful commit transaction In this example, the failed update for file 3 will cause the commit transaction to fail. Atomicity indicates this failure on file 3 will prevent any updates from occurring. Without atomicity, files 1, 2, 4 and 5 would have been updated, but file 3 would not, causing the five files to be out of sync.
automatic recovery	The process of restoring files back to a pristine state after some type of catastrophe (i.e., loss of power to the computer). Automatic recovery is available only for files using full transaction processing (`TRNLOG` file mode). The `TRNLOG` file mode causes all of the changes to the particular file to be logged immediately to a special high-speed transaction log. The presence of this complete history of the changes to the data files with the `TRNLOG` file mode is what makes automatic recovery possible.
byte	The amount of computer memory required to store one character. To store the word "computer" takes 8 bytes. Computer memory and hard disk space are often measured in kilobytes or megabytes. 1 kilobyte = 1024 bytes (characters) = 2^{10}; 1 megabyte = 1048576 bytes (characters) = 2^{20}.
cache	A storage location where data can be more quickly accessed. For example, a disk cache will store frequently accessed data in memory to prevent reading from the slower disk drive.

c-tree Plus	The c-tree file handler API that is the foundation of the c-tree database Server. **c-tree Plus** also serves as the client side development kit for the c-tree client/server model. **c-tree Plus** gives a client side application the ability to communicate (connect) with the **c-tree Server**.
checkpoint	(advanced) An entry placed in **c-tree Server** logs identifying a starting point during the Server's automatic disaster recovery.
client	The second half of the client/server model. The client process is typically a third party application performing a specific task. For example, in a client/server based accounting system, the program prompting the user for input and displaying results would be considered a client.
ctadmn	**c-tree Server** utility program allowing the Server Administrator to grant access to the Server through user identification names. Controls high-level access to the Server and should be executed by the Server Administrator only.
ctdump	**c-tree Server** utility program for creating file backups. This unique utility allows the backup of user data to be performed without restricting user access in any way.
ctfdmp	**c-tree Server** utility program for rolling forward from a specific point in the transaction log history. The **c-tree Server** has the ability to allow users to reset files to specific points in time. This utility allows older backups of user data to be brought up to date.
ctldmp	(advanced) **c-tree Server** utility program for displaying detailed information from the Server transaction logs.
ctrdmp	**c-tree Server** utility that restores the backups made using *ctdump*. This utility also performs data rollbacks.
ctpass	**c-tree Server** utility to allow users to change their password.
ctstop	**c-tree Server** utility for stopping the **c-tree Server**.
data file	A collection of similar pieces of information stored in one location.
deadlock	(advanced) A situation in which two users are prevented from obtaining access to the same record. An example of this situation is if user A locks record 1 for update. At the same time, user B locks record 2 for update. User A now needs to lock record 2 and user B needs to lock record 1. If user A and B both need locks on their target records before processing can continue, a deadlock occurs. The **c-tree Server** automatically handles this situation by returning error messages to the users involved.
directory	A location where files are stored on disk. A directory can be thought of as a hanging folder in a file cabinet. Each file folder within the hanging folder can be thought of as a separate file, or collection of information.
dynamic dump	A method for backing up specified files without restricting user access to the files.
encryption	Disguising information making it difficult for a casual user to inspect the actual contents.

F_TCPIP.DLL	A c-tree Plus communication DLL supporting the TCP/IP transport protocol.
FETCPIP.DLL	A c-tree Plus communication DLL supporting the TCP/IP transport protocol with encryption.
FNETBIOS.DLL	A c-tree Plus communication DLL supporting the NetBIOS transport protocol.
FSHAREMM.DLL	A c-tree Plus communication DLL supporting the shared memory transport protocol.
FSPX.DLL	A c-tree Plus communication DLL supporting the Novell Netware communication protocol IPX/SPX.
file	A collection of related information, referred to as records. See the definitions for directory and record for further information.
folder	The Apple Macintosh term for a location of files on disk. Similar to directory defined above.
hash bins	(advanced) A mathematical algorithm for storing data in memory so it can be quickly located and retrieved. The **c-tree Servers** all use sophisticated hashing routines for data and index caches.
index file	A special type of file used by c-tree Plus and the **c-tree Server** for quickly locating information (records) within a data file.
IPX/SPX	A communication protocol provided by Novell typically used on Novell Netware networks. This communication protocol allows a client process to talk to a Server.
log **(ctstatus, transaction)**	A special purpose file containing important information about a specific process. For example, the c-tree *CTSTATUS* log will contain status information about the Server. Storing items such as when the Server was last started and stopped and what files have been backed up using the dynamic dump facility, etc.
logging	The process of keeping a permanent record of the changes made during a transaction.
message queues	A communication protocol typically used on Unix based operating systems. This communication protocol allows a client process to talk to a c-tree Server process.
mirroring	A mechanism for duplicating important files on different hard drive volumes, partitions or physical devices. If the primary storage location is lost due to some form of catastrophe (i.e., hard disk crash) the mirroring logic can automatically detect the lost connection and switch to the secondary or "mirrored" storage area without any user intervention.
NetBIOS	(Network Basic Input/Output System) A communication protocol typically used on DOS, Windows, OS/2, and Windows NT based operating systems. This communication protocol allows a client process to talk to a Server process.
page	(advanced) A unit of measure for electronic data. At the lowest level operating system read and write, data is manipulated in page sizes. A common page size for DOS is 512 bytes, 2048 bytes for OS/2 and 4096 bytes for Unix.

PREIMG	(advanced) A transaction processing file mode supporting atomicity, but not automatic recovery.
process	For purposes of this Guide, a process is equivalent to a user or connection. One process equals one user.
record	A piece of information stored within a file. Expanding on the file cabinet example used in the directory definition, each piece of paper found within a file folder can be thought of as a record. A record is a unique piece of information similar to other pieces of information (papers) within the same file folder.
roll back	A process made possible with transaction processing allowing file transactions to be reset back to a specific point in time. For example, if a data entry operator enters two hours worth of transactions with incorrect information, these transactions can be removed or rolled back.
roll forward	Similar to roll back, except moving forward in time. The roll forward process is typically for applying transactions to "out dated" (old) files. To do this transaction processing logs containing the desired transactions must be available.
semaphore	(advanced) An operating system level counter used for controlling access to a limited pool of shared resources, such as shared memory.
server	The term server can refer to many different items. For example a "file server" is a specific piece of software for sharing files and hardware devices among users. A good example of a "file server" is the popular Novell network. A "hardware server" is a special computer on which the file server operates. The **c-tree Server** is a "database server", special software efficiently managing multiple users accessing common data.
shared memory	A communication protocol typically used on Unix, OS/2 and Windows NT based operating systems. This communication protocol allows a client process to talk to a **c-tree Server** process.
superfile	A physical file containing any number of logical data and index files.
tar	The tar command is used for creating and managing file backups. c-tree Unix Servers are shipped in tar format. The tar command is used to copy the files from the distribution floppies to the hard drive.
TCP/IP	Transport Control Protocol/Interface Program. A communication protocol available on most operating systems. This communication protocol allows a client process to talk to a Server process.
transaction	A specific operation on a file (i.e., adding a record, deleting a record, updating a record - are all examples of a transaction).
transaction processing	A mechanism by which several data integrity issues are handled. Two of the most important issues are atomicity and automatic recovery.
user	A client process (application) connected to a **c-tree Server**. Each process connected to a **c-tree Server** is counted as a user.

Index