

isCOBOL Evolve: c-tree RTG



Key Topics

- [Installing c-tree](#)
- [Configuring the c-tree Server](#)
- [Configuring the client](#)
- [Accessing from isCOBOL](#)
- [c-tree Utilities](#)
- [Bound Server mode](#)
- [Backup options](#)
- [Troubleshooting](#)

Copyrights

Copyright (c) 2023 Veryant
6390 Greenwich Drive, #225, San Diego, CA 92122, USA

All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution and recompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Veryant and its licensors, if any.

Table of Contents

1. Installing c-tree	1
Installing on Windows	1
Installing on Unix	4
2. SQL Engine Licensing	6
3. Configuring the c-tree Server	7
4. Server Startup	9
Startup on Windows	9
Startup on Unix	10
ctreesql command-line	11
5. Configuring the client	12
Configuring the client through Framework properties	12
Configuring the client through CTREE_CONF	17
<config>	19
<instance>	19
<file>	20
<automkdir>	22
<batchaddition>	22
<bulkaddition>	23
<ctfixed>	24
<datacompress>	24
<datafilesuffix>	26
<detectlock>	26
<encrypt>	26

<hugefile>	27
<ignorelock>	27
<indexfilesuffix>	28
<keycheck>	28
<keycompress>	29
<log>	30
<locktype>	31
<maxlencheck>	31
<map>	31
<memoryfile>	32
<optimisticadd>	32
<prefetch>	33
<retrylock>	35
<rpc>	35
<runitlockdetect>	36
<skiplock>	37
<smartcopy>	37
<sqlize>	38
<transaction>	40
<trxholdslocks>	40
<writethru>	41

6. Accessing from isCOBOL 42

The URL syntax	42
Registering existing c-tree files in c-tree SQL Server	43
Creating c-tree files into SQL Server from the COBOL Program	45
Accessing c-tree files through SQL from the COBOL Program	47
Multithread programs	48

7. c-tree Utilities 49

Command-line utilities	49
ctadmn	50
Server administration	50
Online and offline backups	51
ctdump	51
ctfdmp	51

ctfileid	51
ctldmp	52
ctquiet	52
ctrdmp	53
ctsqlcdb	53
ctsqlutl	53
ctstat	54
ctstop	55
cttctx	56
cttrnmod	56
ctunf1	57
ctutil	58
-info	59
-param	59
-profile	59
-copy	59
-setpath	60
-tron	60
-rename	60
-remove	61
-check	61
-rebuild	61
-getimg	61
-rblimg	62
-makeimg	63
-checkimg	63
-compact	64
-unload	64
-unloadtext	65
-load	65
-loadtext	65
-segment	66
-make	66
-sqlinfo	66
-sqllink	67

-sqlunlink	67
-sqlize	68
-conv	68
-compress	69
-uncompress	69
dbdump	69
dbload	70
dbschema	70
isql	71
sa_admin	72
Graphical utilities	73
c-treeACEMonitor	73
A Java version of this utility is provided through the library c-treeACEMonitor.jar.	74
c-treeConfigManager	74
Refer to c-tree Server Administrator's Guide for additional information on these utilities.	76
c-treeGauges	76
c-treeISAMExplorer	77
c-treeLoadTest	78
c-treeLogAnalyzer	79
c-treePerfMon	80
c-treeQueryBuilder	81
c-treeSecurityAdmin	82
c-treeSqlExplorer	83
c-treeTPCATEST	84
DrCtree	85
ErrorViewer	86
8. Bound Server mode	87
9. Troubleshooting	88
Problems working with the c-tree Server Windows Service	88
Problems starting the c-tree Server Service	88
Problems connecting to the c-tree Server Service	89
Problems stopping the c-tree Server Service	89
Problems connecting from a COBOL program	89
Error Codes	90

10.Drivers	110
ODBC Driver	110
JDBC Driver	112
ADO.NET Driver	112
PHP Driver	112
PYTHON Driver	112
11.External References	113

Chapter 1

Installing c-tree

c-treeRTG (also called simply c-tree) is a file server. It's composed of a server (the c-tree Server) and a client (the c-tree client library) that can be installed on the same machine or on different machines.

Each time an i/o operation is issued, the client sends a request to the server and the server part operates locally on the server machine.

This kind of approach provides a fast and secure access to remote files and it's a strongly suggested alternative to shared directories.

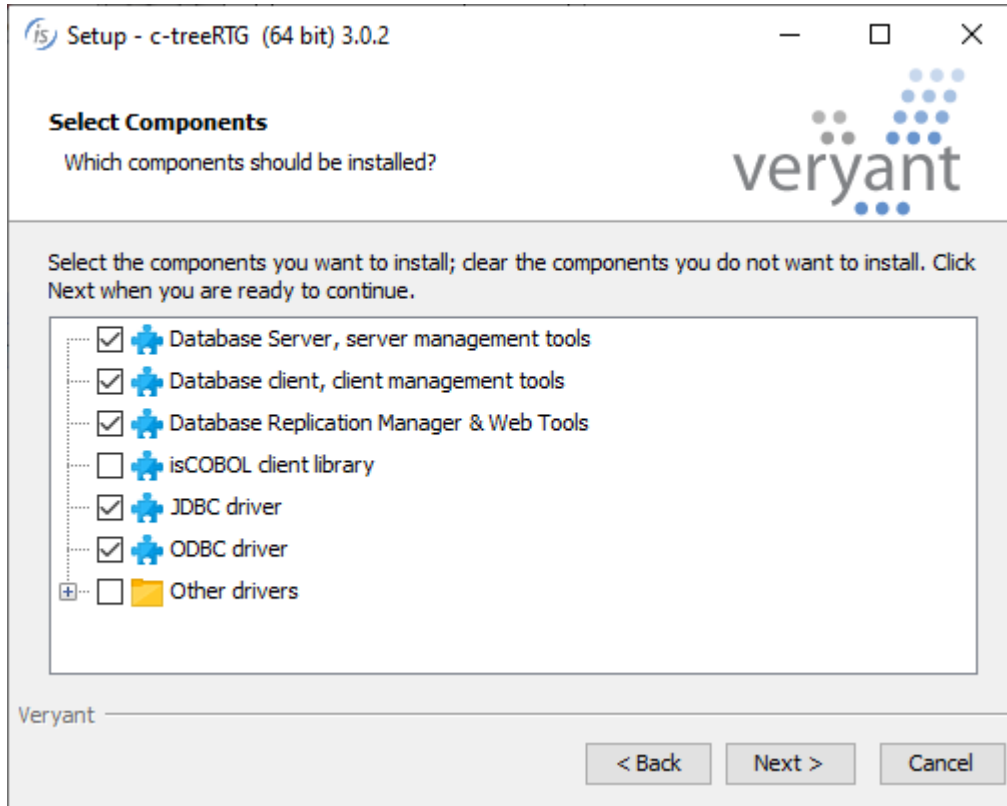
c-tree can also work as a database. isCOBOL accesses its files with standard i-o statements, but other tools, like Ms Office applications and other ODBC and JDBC clients can work on these files through SQL.



Installing on Windows

c-tree provides a graphical setup wizard for the installation on Windows.

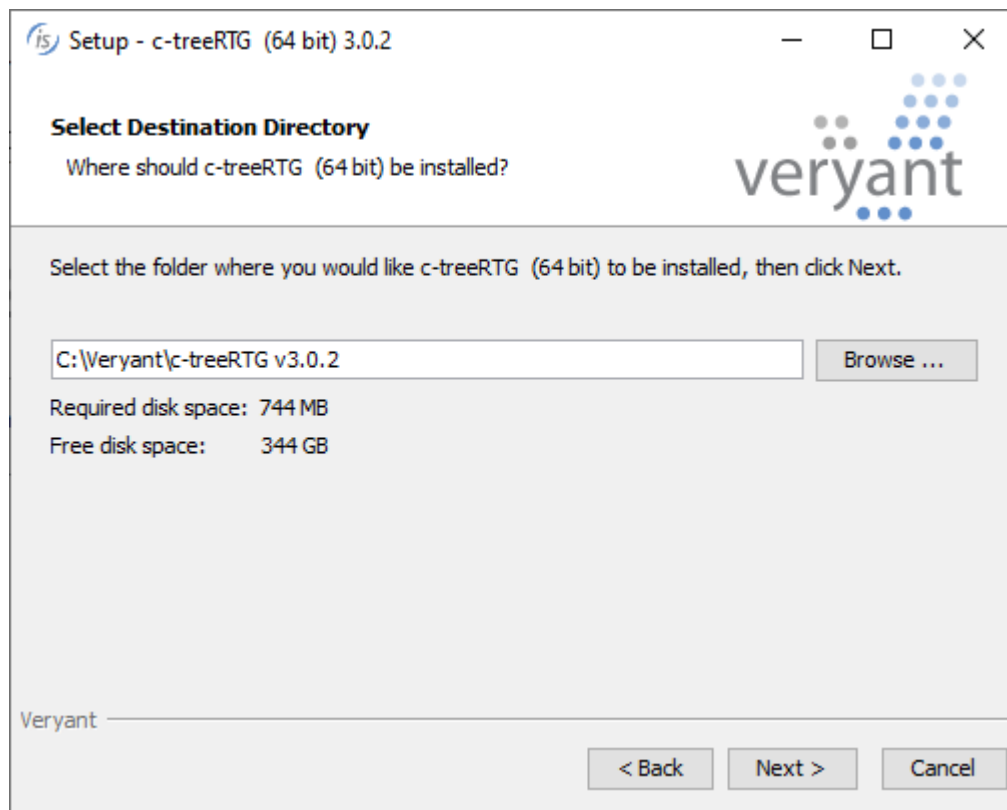
First of all, you have to choose which kind of installation to perform.



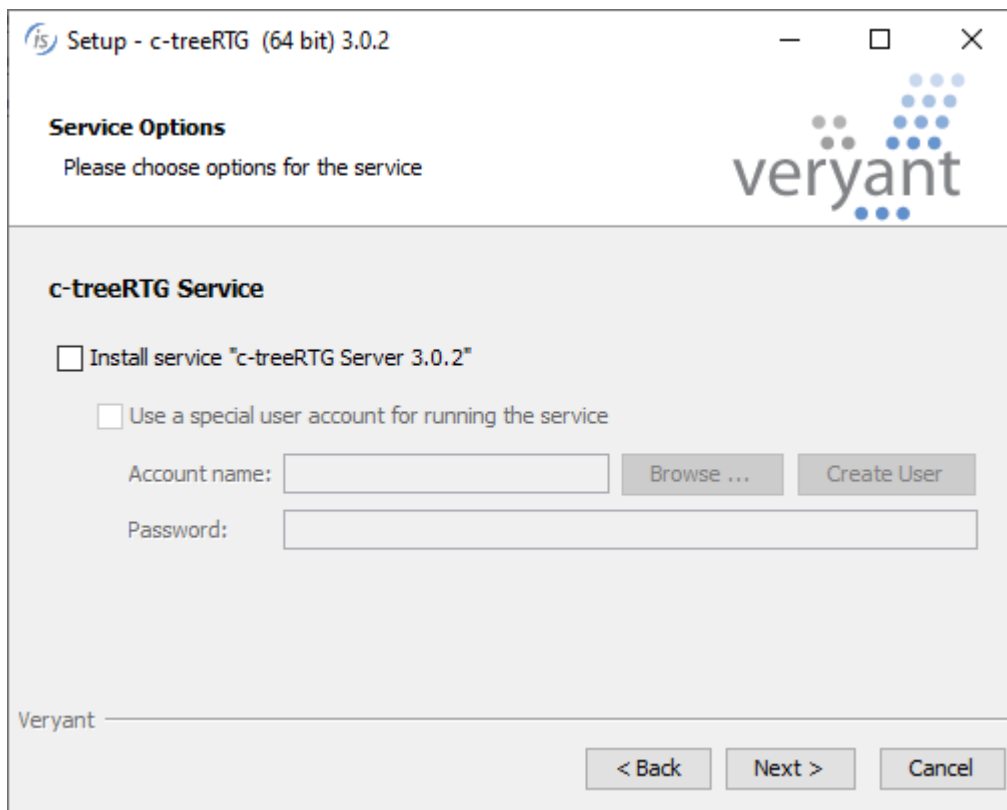
- *Database Server* is the full server installation. If this option is checked, server products will be installed on the PC.
- *Database Client* is the client installation. If this option is checked, client database tools are installed.
- *Database Replication Manager & Web Tools* is the HTML interface that allows you to manage the c-tree Server and the Replication Agent. If this option is checked, the HTML interface is installed.
- *isCOBOL client library* is the c-tree client installation. If this option is checked, the client library (ctree.dll) will be installed. The setup will ask for the destination directory later.
- *ODBC driver*, if checked, installs the ODBC driver for c-tree.
- *JDBC driver*, if checked, installs the JDBC driver for c-tree.
- *Other drivers* is a collection of drivers to interface c-tree with other COBOLs and other programming languages. If this option is checked, all the drivers will be installed on the PC. If you wish to install only some specific drivers, expand the item and check only the desired drivers.

If *isCOBOL client library* was checked, the next step will let you choose which version of isCOBOL will be updated with the c-tree client library.

The setup lets you select the path in which the c-tree and its utilities will be installed.



If Database Server was checked, the setup lets you create a Windows service for the c-tree Server



The service must be managed via the sc command or the Windows Service Manager, as explained in [Windows Service](#).

Installing on Unix

c-tree for other platforms comes in a gzipped tar file. The name of the file is: c-treeRTG_vMajor.Minor.Build_Platform.tar.gz

Current version of c-tree is:

- 3.0.2

Currently available platforms are:

- IBM AIX 5 (32 bit)
- IBM AIX 5 (64 bit)
- Linux x86 32-bit
- Linux x64 64-bit
- Mac OS 64-bit
- Sun Solaris (SPARC - 32 bit)
- Sun Solaris (SPARC - 64 bit)
- OpenServer 10

Note - If you need a different porting, ask to Veryant support.

Unzip and extract the tar file with the following command:

```
gunzip c-treeRTG_vMajor.Minor.Build_Platform.tar.gz  
tar -xvf c-treeRTG_vMajor.Minor.Build_Platform.tar
```

A directory called *c-treeRTG_vMajor.Minor* will be created.

Chapter 2

SQL Engine and Replication Licensing

c-tree is provided with an initial license that allows:

- unlimited ISAM server features
- limited SQL server features
- no data replication features

The SQL Engine works for three hours from the c-tree startup and allows a maximum of 4 concurrent connections.

In order to unlock the SQL server and take advantage of SQL features without limitations, you need to purchase a license.

The same is applicable to the enablement of data replication features.

Contact your Veryant representative to purchase a license. You will be provided with a file named *ctsrvr#####.lic* . In order to install the license

1. stop the c-tree service if it's running
2. replace *ctsrvr#####.lic* in the c-tree installation folder with the new one
3. (re)start the c-tree service

Chapter 3

Configuring the c-tree Server

The c-tree Server looks for a file called *ctsrvr.cfg* located in the *config* directory. If this file is missing, the default settings are used.

It's possible to use an alternate configuration file through the command line option `CTSRVR_CFG`.

Example for Windows:

```
faircom.exe CTSRVR_CFG C:\etc\customer1.cfg
```

Example for Unix/Linux:

```
./faircom CTSRVR_CFG /etc/customer1.cfg
```

Keywords in the configuration file must be followed by a value and are not case-sensitive.

Commented lines have a semicolon as first digit.

The table below lists configuration keywords included in the default *ctsrvr.cfg* file installed with c-tree..

CONNECTIONS	The maximum number of connections to the c-tree Server. Typically, 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.
-------------	---

COMM_PROTOCOL	<p>Specifies a communications module loaded by the Server. Possible values:</p> <p>F_TCPIP F_TCIPV6 FSHAREMM DISABLE</p> <p>If no protocol is specified, then the default system communication protocol is used.</p> <p>An application can override the default protocol by specifying the protocol in the connection string with the following syntax: FAIRCOMS@myhost^F_TCPIP or FAIRCOMS@myhost^F_TCIPV6</p> <p>The c-tree Server can support TCP/IP and shared memory simultaneously. For example, the server could be communicating with some users locally through shared memory and others using TCP/IP over an Ethernet connection. Each protocol to be loaded by the c-tree Server requires a separate COMM_PROTOCOL line in the configuration script. The following example loads TCP/IP and Shared Memory Model protocols:</p> <pre>COMM_PROTOCOL F_TCPIP COMM_PROTOCOL FSHAREMM</pre>
SUBSYSTEM COMM_PROTOCOL SSL	<p>Enables SSL communication. The following entries are required: SERVER_CERTIFICATE_FILE to specify the path to the pem file SSL_CONNECTIONS_ONLY to specify if only SSL is accepted or standard connections are accepted as well.</p> <p>Example:</p> <pre>SUBSYSTEM COMM_PROTOCOL SSL { SERVER_CERTIFICATE_FILE ctreessl.pem SSL_CONNECTIONS_ONLY YES }</pre>
COMPATIBILITY TCPIP_CHECK_DEAD_CLIENTS	<p>When this flag is present, the c-tree server recognizes when a connection between the client and the server, for some reason is lost. The c-tree server can automatically remove the connection and release the files and records related with the lost connection. See also DEAD_CLIENT_INTERVAL <seconds>.</p>

DAT_MEMORY	<p>The memory allocated to the data cache, in bytes.</p> <p>It's possible to measure this settings in Kilobytes, Megabytes or Gigabytes by specifying the KB, the MB and the GB suffix respectively. For example, the following values are equivalent:</p> <ul style="list-style-type: none"> • 200 MB • 204800 KB • 209715200 <p>The default value is 100 MB.</p>
DELAYED_DURABILITY	<p>When specified with a value greater than 0, the c-tree Server does not sync its log buffer simply because a transaction commits. Instead, it syncs the log buffer if it becomes full, or if a write to the data cache requires the log buffer to be flushed to disk, or if the maximum defer time in seconds specified by this keyword is exhausted.</p>
LOG_SPACE	<p>The size of the transaction log.</p>
DEAD_CLIENT_INTERVAL <seconds>	<p>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).</p> <p>COMPATIBILITY TCPIP_CHECK_DEAD_CLIENTS is required to enable the checking. Otherwise, no check is made.</p>
FILES	<p>The maximum number of data files and indices where each index 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 towards the FILES total.</p> <p>Every time the c-tree server opens a new file, the counter is increased. If the c-tree server opens the same file for different clients, instead, the counter is not increased.</p> <p>The default value is 1000.</p>
IDX_MEMORY	<p>The memory allocated to the index cache, bytes.</p> <p>It's possible to measure this settings in Kilobytes, Megabytes or Gigabytes by specifying the KB, the MB and the GB suffix respectively. For example, the following values are equivalent:</p> <ul style="list-style-type: none"> • 200 MB • 204800 KB • 209715200 <p>The default value is 100 MB.</p>
MAX_DAT_KEY	<p>Maximum number of indices per data file.</p> <p>The default value is 64.</p>

MAX_KEY_SEG	<p>Maximum number of key segments allowed per index.</p> <p>The default value is 16.</p>
MAX_FILES_PER_USER	<p>The maximum number of data files and indices (where each index counts toward this total) per client connection.</p> <p>The default value is 2048.</p>
PAGE_SIZE	<p>The number of bytes per buffer page in bytes. Only the following values are accepted (all other values generate an error):</p> <ul style="list-style-type: none"> • 1024 • 2048 • 4096 • 8192 • 16384 • 32768 • 65536 <p>A file created with a larger PAGE_SIZE cannot be opened by a server with a smaller PAGE_SIZE.</p> <p>The default value is 32768 in v12 and later (before v12, the default was 8192).</p> <p>Since this setting affects also FCS files and SQL databases, after changing it you should restart the c-tree Server with a clean data folder. Alternatively you can convert the files in the data folder by following the procedure described in Faircom's documentation at Adjusting PAGE_SIZE.</p>
SERVER_NAME	<p>A name assigned to c-tree Server, instead of the default name.</p> <p>The default value is FAIRCOMS.</p>
SERVER_PORT	<p>A port assigned to c-tree Server, instead of the default port.</p> <p>The default value is 5597.</p>
SERVER_DIRECTORY	<p>Working directory of the c-tree Server. This directory is used to resolve all relative paths of c-tree files. By default the working directory is the directory in which the c-tree Server has been started.</p>
LOCAL_DIRECTORY	<p>Base directory for c-tree files. This directory is used to resolve all relative paths of c-tree files. If LOCAL_DIRECTORY is a relative path, then it's appended to SERVER_DIRECTORY. If omitted, only SERVER_DIRECTORY is used to resolve relative paths of c-tree files.</p>
CONSOLE NO_SHUTDOWN_PROMPT	<p>If present, ctsrvr will not prompt for administrator password when closed (Windows only).</p>
CONSOLE TOOL_TRAY	<p>If present, ctsrvr will start minimized in the system tray (Windows only).</p>

SHMEM_DIRECTORY	<p>On Unix systems, the c-treeRTG shared memory communication protocol creates a file that clients use to find the shared memory identifier for its shared memory logon region, and it creates named pipes for initial communication between client and server.</p> <p>This option sets the directory in which c-tree stores files used for connecting using the Unix shared memory protocol.</p> <p>This configuration option can include an environment variable name that will be substituted with its value when the configuration file is read.</p> <p>Note - If SHMEM_DIRECTORY is set, clients must be able to find this non-default directory. Client processes will check the environment variable CTREE_SHMEM_DIRECTORY. A client looking in the wrong location for shared memory information may take extra time to connect waiting for the shared memory protocol to timeout before falling back to TCP.</p>
SQL_DATABASE	<p>Name of the database. By default this entry is set to "ctreeSQL". Database files are automatically created during the first startup. This name must be used by ctutil -sql* op-codes and external SQL tools to connect and operate with isCOBOL ISAM Server.</p>
SQL_OPTION DROP_TABLE_DICTIONARY_ONLY	<p>Affects SQL DROP TABLE command operation. This option retains files on disk when tables are dropped from the SQL dictionary. Use this option when you want to undo a DROP TABLE command. You can also use this option when you want files/tables to remain available in ISAM and not in SQL.</p> <p>This option is particularly useful when using c-tree features, such as <i>ctutil -sqlize</i>, which may bring extra files into the SQL data dictionaries that later need to be dropped from SQL without removing them from disk.</p>

Chapter 4

Server Startup

The c-tree Server can run both in background or foreground mode.

Startup on Windows

In order to start the c-tree Server in foreground mode, launch 'ctreesql.exe'. You can run it from the Windows Start menu: *Start > Programs > c-treeRTG v3.0.2 > c-treeRTG Server*

Unless the entry `CONSOLE TOOL_TRAY` is present in the `ctsrvr.cfg` file, the following panel will appear.



Also, a new icon will appear in the system tray.

From the *View* menu you can hide the Message Monitor or show the Function Monitor (list of all internal functions called from clients). From the *Control* menu you can shut down the server. The same result is obtained by clicking on its window close button.

The c-tree Server can also be started from a command prompt by running the *faircom.exe* file installed in the c-treeRTG "server" folder. It is the same executable file pointed by the link mentioned above. The faircom executable file supports command-line parameters allowing you to specify configuration options or a custom configuration file directly on the command-line; see [The faircom executable command-line](#) for details.

Windows Service

In order to start the c-tree Server in background mode, you must create a service.

If you didn't create the service during the setup process, you can start an MS-DOS prompt with Administrator privileges and run:

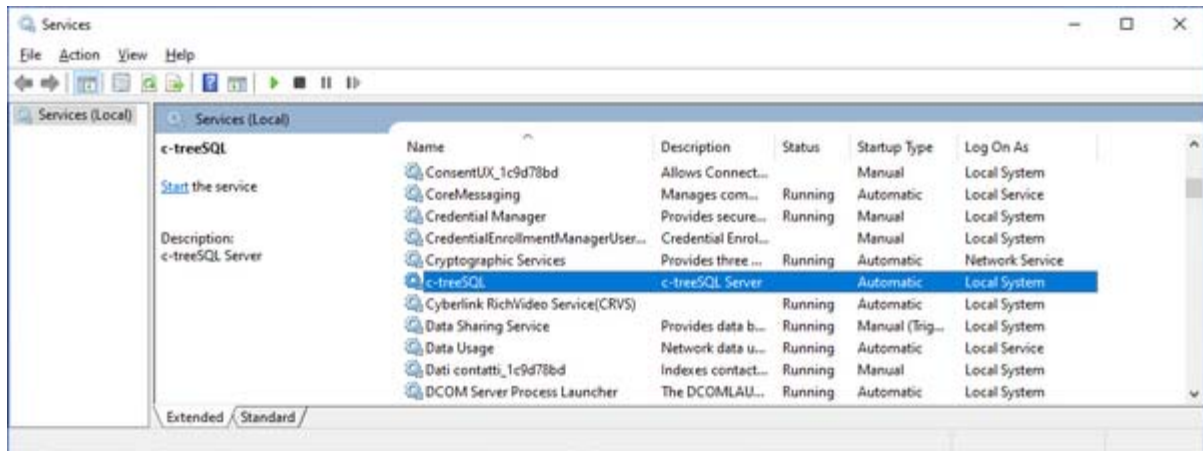
```
sc create c-treeSQL start= auto binPath= "C:\Veryant\c-treeRTG  
v3.0.2\server\faircom.exe"
```

Note - "c-treeSQL" is just a suggested name, but any name can be used. The path to *faircom.exe* might be different on your machine.

After it, you can optionally assign a description to the service as follows:

```
sc description c-treeSQL "c-treeSQL Server"
```

If the above commands are successful, you will find this new service available in the Service Control Manager:



You can manage the c-treeSQL service from the Service Control Manager or by running other sc commands in the MS-DOS prompt.

For more information about the sc command, refer to Microsoft documentation: <https://technet.microsoft.com/en-us/library/bb490995.aspx>.

To remove the service from the system, use this command:

```
sc delete c-treeSQL
```

Starting multiple services on the same machine

In this example we show how to start two c-tree Server services listening on different ports on the same machine.

1. Create two configuration files, e.g.

ctsrvr1.cfg:

```
SERVER_NAME FAIRCOMS1
SERVER_PORT 6661
LOCAL_DIRECTORY ./data1/
```

ctsrvr2.cfg:

```
SERVER_NAME FAIRCOMS2
SERVER_PORT 6662
LOCAL_DIRECTORY ./data2/
```

Note - the two snippets above shows the basic configuration, they should be completed with the rest of settings that usually appear in a c-tree configuration. We notice that different ports and different data folders are referenced in the two configuration files.

2. Create the two services as follows:

```
sc create c-treeSQL-1 start= auto binPath= "C:\Veryant\c-treeRTG
v3.0.2\server\faircom.exe CTSRVR_CFG \path\to\ctsrvr1.cfg"

sc create c-treeSQL-2 start= auto binPath= "C:\Veryant\c-treeRTG
v3.0.2\server\faircom.exe CTSRVR_CFG \path\to\ctsrvr2.cfg"
```

Startup on Unix

The Unix version of c-tree does not show any graphical window. All messages are printed on the console.

To start the server in foreground mode, change to the "server" directory and launch the following command:

```
./faircom
```

To start the server in background mode, change to the "server" directory and launch the following command:

```
./faircom &
```

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 faircom &
```

To stop the server, you can use either ctadmn or ctstop utilities, that will be discussed later.

See [The faircom executable command-line](#) for information about the available command-line options.

Starting multiple Servers on the same machine

In this example we show how to start two c-tree Server processes listening on different ports on the same machine.

1. Create two configuration files, e.g.

ctsrvr1.cfg:

```
SERVER_NAME FAIRCOMS1
SERVER_PORT 6661
LOCAL_DIRECTORY ./data1/
```

ctsrvr2.cfg:

```
SERVER_NAME FAIRCOMS2
SERVER_PORT 6662
LOCAL_DIRECTORY ./data2/
```

Note - the two snippets above shows the basic configuration, they should be completed with the rest of settings that usually appear in a c-tree configuration. We notice that different ports and different data folders are referenced in the two configuration files.

2. Start the two processes as follows:

```
faircom CTSRVR_CFG /path/to/ctsrvr1.cfg
faircom CTSRVR_CFG /path/to/ctsrvr2.cfg
```

Note - On Unix platforms, the c-treeRTG server shares a directory with the c-treeRTG clients when they communicate via the shared memory protocol. It is possible to configure the c-treeRTG server to change the default shared memory directory with configuration keyword [SHMEM_DIRECTORY](#). Using different shared memory directories allows running multiple c-treeRTG servers with the same name on the same machine. To configure the c-treeRTG clients to set the shared memory directory, it is necessary to set [iscobol.file.index.ctshmemdir](#) in the runtime configuration file or [ctshmemdir](#) in the CTREE_CONF file.

The faircom executable command-line

The faircom executable accepts 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 before the standard configuration file, *ctsrvr.cfg*.

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:

```
faircom 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 command-line keyword CTSRVR_CFG followed by a fully qualified configuration file name as follows:

```
faircom CTSRVR_CFG C:\myServer\ctsrvr.cfg
```

3. The CTSRVR_CFG command line keyword is typically used when running two Servers on the same machine.

Chapter 5

Configuring the client

Configuring the client through Framework properties

The isCOBOL Framework looks for the c-tree configuration between configuration properties if

- the c-tree version is 9.5.49790 or greater
- the property `iscobol.ctree.new_config (boolean)*` is set to true

Properties are evaluated at the first open of a c-tree file. Changing them after the first c-tree file has been opened will have no effect.

Global properties

The following properties affect the whole session, they can't be set for a specific instance:

Property	Description
<code>iscobol.file.index.filepool</code>	True = the Filepool feature is enabled. False = the Filepool feature is disabled. If omitted, <i>False</i> is assumed. Note - this property just enables the Filepool feature; you need to set <code>iscobol.file.index.inpool</code> to true in order to actually include files in the pool.
<code>iscobol.file.index.filepool.size</code>	Specifies the maximum number of files that can be included in the pool.
<code>iscobol.file.index.log.debug.batchaddition (boolean)</code>	True = trace batchadditions details. False = do not trace batchaddition details. If omitted, <i>False</i> is assumed.
<code>iscobol.file.index.log.debug.config (boolean)</code>	True = trace config details, limited to the configuration set by the user. False = do not trace config details. If omitted, <i>False</i> is assumed.

Property	Description
<code>iscobol.file.index.log.debug.config.full</code> (boolean)	<p>True = trace complete config details, including the default configuration. False = do not trace complete config details.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.log.debug.prefetch</code> (boolean)	<p>True = trace prefetch details. False = do not trace prefetch details.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.log.error</code> (boolean)	<p>True = trace errors in the log file. False = do not trace errors in the log file.</p> <p>If omitted, <i>True</i> is assumed.</p>
<code>iscobol.file.index.log.error.atend</code> (boolean)	<p>True = trace EOF errors. False = do not trace EOF errors.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.log.error.notfound</code> (boolean)	<p>True = trace NOTFOUND errors. False = do not trace NOTFOUND errors.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.log.file</code>	<p>Specifies the name of the log file where to trace c-tree client operations. When this property is set, some log is enabled by default: error, info and warning. Set the other <code>iscobol.file.index.log</code> properties to true or false if you wish to disable or enable a particular log.</p> <p>If omitted, no log is created.</p>
<code>iscobol.file.index.log.info</code> (boolean)	<p>True = trace generic information in the log file. False = do not trace generic information in the log file.</p> <p>If omitted, <i>True</i> is assumed.</p>
<code>iscobol.file.index.log.profile</code> (boolean)	<p>True = trace profile information in the log file. False = do not trace profile information in the log file.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.log.warning</code> (boolean)	<p>True = trace warning information in the log file. False = do not trace warning information in the log file.</p> <p>If omitted, <i>True</i> is assumed.</p>

Instance properties

The following properties can be set for a specific instance as explained in [Configuring multiple instances](#) below. If you don't configure multiple instances, they affect the default instance that is created by isCOBOL when the program opens the first c-tree file:

Property	Description
<code>iscobol.file.index.anyunlock</code> (boolean)	<p>True = Any UNLOCK performed in the run unit on a file will remove the lock on the file even if the record was locked by another OPEN instance.</p> <p>False = The lock is removed only if the UNLOCK is performed on the same OPEN instance where the READ WITH LOCK was performed.</p> <p>This property can be set only if iscobol.file.index.autolock_allowed (boolean) is set to true.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.autolock_allowed</code> (boolean)	<p>True = More locks on the same record can be acquired in the same run unit.</p> <p>False = Only one lock on the same record can be acquired in the same run unit.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.batchaddition</code> (boolean)	<p>True = write operations are buffered in order to speed up performance.</p> <p>False = write operations are not buffered.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.batchaddition.records</code>	<p>Specifies the number of write operations that must be buffered at a time.</p> <p>If omitted, <i>10</i> is used.</p>
<code>iscobol.file.index.bulkaddition</code> (boolean)	<p>True = files open for output or extend are managed with bulk addition (indexes are written at the close of the file) to speed up performance.</p> <p>False = files open for output or extend are managed without bulk addition.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.connect</code> (boolean)	<p>True = connect to c-tree server during runtime initialization.</p> <p>False = connect to c-tree server at the first open of a c-tree file.</p> <p>If omitted, <i>False</i> is assumed.</p>

Property	Description
<code>iscobol.file.index.ctshmemdir</code>	<p>Configure the c-treeRTG server to change the default shared memory directory.</p> <p>This setting is ignored under Windows as it is meaningful only for Unix platforms.</p> <p>On Unix platforms, the c-treeRTG server shares a directory with the c-treeRTG clients when they communicate via the shared memory protocol. It is possible to configure the c-treeRTG server to change the default shared memory directory with configuration keyword SHMEM_DIRECTORY. Using different shared memory directories allows running multiple c-treeRTG servers with the same name on the same machine. To configure the c-treeRTG clients to set the shared memory directory, it is necessary to set this property.</p>
<code>iscobol.file.index.data_suffix</code>	<p>Specifies the extension for the data file.</p> <p>If omitted, then <i>.dat</i> is assumed.</p>
<code>iscobol.file.index.datacompress (boolean)</code>	<p>True = data records are compressed using zlib algorithm. False = data records are not compressed.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.datacompress.level</code>	<p>This property selects the zlib compression level. Valid values are between 1 and 9 where 1 gives best speed but use more disk space and 9 gives best compression at the expenses of performance.</p> <p>If omitted, then <i>1</i> is assumed.</p>
<code>iscobol.file.index.datacompress.strategy</code>	<p>This property specifies the strategy for data compression. Possible values are:</p> <ul style="list-style-type: none"> 0 = Use the default zlib compression strategy. 1 = Use the zlib filtered compression strategy. 2 = Use zlib Huffman only compression strategy. 3 = Use zlib RLE compression strategy. 4 = Use zlib fixed compression strategy. <p>If omitted, <i>0</i> is assumed.</p>
<code>iscobol.file.index.datacompress.type</code>	<p>Specifies the algorithm for data compression. Valid values are:</p> <ul style="list-style-type: none"> zlib = zlib algorithm is used. rle = rle algorithm is used. <p>If omitted, then <i>rle</i> is assumed.</p>
<code>iscobol.file.index.encrypt (boolean)</code>	<p>True = files are encrypted using ctCAMO encryption algorithm. False = files are not encrypted.</p> <p>If omitted, <i>False</i> is assumed.</p>

Property	Description
<code>iscobol.file.index.endian check (boolean)</code>	<p>True = check the byte endianness at connection time False = don't check the byte endianness at connection time</p> <p>If omitted, <i>True</i> is assumed.</p> <p>This logic checks at connection time to ensure that the byte endianness of both client and server is the same (if it is not, the connection is terminated and the operation fails). In rare cases where this logic is not necessary (e.g., if files have no numeric data types), the byte endianness check can be skipped so the connection can succeed.</p>
<code>iscobol.file.index.file_m apping</code>	<p>This property specifies the list of files that should be included in the current instance. Multiple names must be separated by comma. Wildcards are supported. See Configuring multiple instances for details.</p>
<code>iscobol.file.index.file_m apping_p</code>	<p>This property specifies the list of files that should be included in the current instance. Multiple names must be separated by comma. Wildcards are supported.</p> <p>At the end of each file name you can put an ordinal number that is the number of the path between <code>iscobol.file.index_path_mapping</code> paths. In this way it's possible to specify a file mapping for each path mapping.</p> <p>Example:</p> <pre>iscobol.file.index.1.path_mapping=/dir1,/dir2 iscobol.file.index.1.file_mapping_p=customers1,invoices1,reports2,discounts</pre> <p>The instance mapping is valid for the files customers and invoices when they're are under /dir1, for reports when it's is under /dir2 and for discounts, wherever it is.</p> <p>See Configuring multiple instances for details.</p>
<code>iscobol.file.index.fileve rsion</code>	<p>This property specifies the backward compatibility for every new file created in the current c-tree instance. It affects default attributes such as the default compression algorithm.</p> <p>Set this property if you plan to use the same c-tree files also with previous c-tree major versions.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> 1 = create files compatible with c-tree v1 (all versions before v11) 2 = create files compatible with c-tree v2 (also known as v11) 3 = create files compatible with c-tree v3 <p>The default value is 3.</p> <p>Note - the compatibility with previous versions might be affected also by the PAGE_SIZE setting in the server configuration. Be sure to use the same page size between the current c-tree version and the previous c-tree version you want to be compatible with.</p>

Property	Description
<code>iscobol.file.index.fixed_length</code> (boolean)	<p>True = files are created with fixed length records. False = files are created with variable length records.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.forced_elete</code> (boolean)	<p>True = if a file segment is missing, DELETE FILE deletes the other segment. False = if a file segment is missing, DELETE FILE returns a 'file corrupted' error.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.index_suffix</code>	<p>Specifies the extension for the index file.</p> <p>If omitted, then <i>.idx</i> is assumed.</p>
<code>iscobol.file.index.inpool</code>	<p>True = files are included in a pool to speed up the next OPEN operations. False = files are not included in a pool.</p> <p>If omitted, <i>False</i> is assumed.</p> <p>Note - the <code>iscobol.file.index.filepool</code> global setting must be set to true, otherwise this property will be ignored.</p>
<code>iscobol.file.index.keycheck</code> (boolean)	<p>True = extra keys in the file are not allowed. False = extra keys in the file are allowed.</p> <p>If omitted, <i>True</i> is assumed.</p>
<code>iscobol.file.index.keycompress</code> (boolean)	<p>True = index files are compressed using zlib algorithm. Leading and padding compression are applied. False = index files are not compressed.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.keycompress.leading</code> (boolean)	<p>True = leading compression is applied on index files. False = leading compression is not applied on index files.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.keycompress.padding</code> (boolean)	<p>True = padding compression is applied on index files. False = padding compression is not applied on index files.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.keycompress.rle</code> (boolean)	<p>True = RLE compression is applied on index files. False = RLE compression is not applied on index files.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.keycompress.vlennod</code> (boolean)	<p>True = Only RLE and PADDING compression are allowed on index files. False = Only LEADING and PADDING compression are allowed on index files.</p> <p>If omitted, <i>True</i> is assumed.</p>

Property	Description
<code>iscobol.file.index.lock_read_anyhow</code> (boolean)	True = locked records are returned. False = locked records are not returned. If omitted, <i>False</i> is assumed.
<code>iscobol.file.index.lock_wait</code> (boolean)	True = if a record is locked, wait until lock is released. False = if a record is locked, return a record locked error. If omitted, <i>False</i> is assumed.
<code>iscobol.file.index.log.debug.batchaddition</code> (boolean)	True = trace batchadditions details. False = do not trace batchaddition details. If omitted, <i>False</i> is assumed.
<code>iscobol.file.index.log.debug.prefetch</code> (boolean)	True = trace prefetch details. False = do not trace prefetch details. If omitted, <i>False</i> is assumed.
<code>iscobol.file.index.log.error</code> (boolean)	True = trace errors in the log file. False = do not trace errors in the log file. If omitted, <i>False</i> is assumed.
<code>iscobol.file.index.log.error.atend</code> (boolean)	True = trace EOF errors. False = do not trace EOF errors. If omitted, <i>False</i> is assumed.
<code>iscobol.file.index.log.error.notfound</code> (boolean)	True = trace NOTFOUND errors. False = do not trace NOTFOUND errors. If omitted, <i>False</i> is assumed.
<code>iscobol.file.index.log.file</code>	Specifies the name of the log file where to trace c-tree client operations. If omitted, no log is created.
<code>iscobol.file.index.log.info</code> (boolean)	True = trace generic information in the log file. False = do not trace generic information in the log file. If omitted, <i>False</i> is assumed.
<code>iscobol.file.index.log.profile</code> (boolean)	True = trace profile information in the log file. False = do not trace profile information in the log file. If omitted, <i>False</i> is assumed.

Property	Description
<code>iscobol.file.index.maxinstance</code>	<p>This property specifies how many instances can be created. It is mandatory in order to work with multiple instances. The value must range between 2 and 99. Any value less than 2 is considered as 2 while any value greater than 99 is considered 99. Note that the default instance is counted, so you should calculate the value of this property by adding 1 to the highest instance number you put in your property names. See Configuring multiple instances for details.</p>
<code>iscobol.file.index.maxlencheck</code> (boolean)	<p>True = return an error if record is larger than maxlen bytes.. False = record larger than maxlen bytes is allowed.</p> <p>If omitted, <i>False</i> is assumed</p>
<code>iscobol.file.index.memoryfile</code> (boolean)	<p>True = files are created in memory rather than on disk. False = files are created on disk.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.memoryfile.persist</code> (boolean)	<p>True = memory files are removed when the c-tree Server is shut down. False = memory files are removed when the client session that creates them terminates.</p> <p>If omitted, <i>True</i> is assumed.</p>
<code>iscobol.file.index.optimisticadd</code> (boolean)	<p>True = use optimistic strategy when adding records. Unique key violation is not checked before writing the whole record. This provides better performance. False = use pessimistic strategy when adding records.. Unique key violation is checked before writing the whole record.</p> <p>If omitted, <i>True</i> is assumed</p>
<code>iscobol.file.index.password</code>	<p>This property specifies the password for the connection to the c-tree Server.</p>
<code>iscobol.file.index.path_mapping</code>	<p>This property specifies the list of paths that should be included in the current instance. Multiple paths must be separated by comma. See Configuring multiple instances for details.</p>
<code>iscobol.file.index.prefetch</code> (boolean)	<p>True = read operations are buffered in order to speed up performance. False = read operations are not buffered.</p> <p>If omitted, <i>False</i> is assumed.</p> <p>The number of buffered records and the amount of time they're kept in memory is controlled by iscobol.file.index.prefetch.records and iscobol.file.index.prefetch.ttl respectively.</p> <p>Prefetched records are always locked by default. You can change this behavior by setting iscobol.file.index.prefetch.allowwriters (boolean) to true.</p>

Property	Description
<code>iscobol.file.index.prefetch.allowwriters</code> (boolean)	<p>True = prefetched records are not locked. False = prefetched records are locked.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.prefetch.records</code>	<p>Specifies the number of read operations that must be buffered at a time.</p> <p>If omitted, <i>10</i> is used.</p> <p>Note - this value should be set carefully according to the number of read operations between a Start and another. Having an high value for this setting and few read operations between a Start and another might slow down performance as useless records are loaded in memory even if the program will not read them.</p>
<code>iscobol.file.index.prefetch.ttl</code>	<p>This property specifies how many seconds the prefetched records should be kept in memory.</p>
<code>iscobol.file.index.read_lock_test</code>	<p>True = a READ WITH NO LOCK returns a lock condition if the record is locked. False = a READ WITH NO LOCK reads the record even if it's locked.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.rowid</code>	<p>True = enable support for creating a ROWID field/index. False = disable support for creating a ROWID field/index.</p> <p>ROWID is required for creating Full-Text Indexes, so this option is necessary if Full-Text Search is desired.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.rowid.size</code>	<p>This property sets the size of the rowid hidden field in the record header. Accepted values are 8 and 4.</p> <p>The default value is 8.</p>
<code>iscobol.file.index.rpc</code> (boolean)	<p>True = use RPC calls to communicate with c-tree server False = use standard c-tree calls to communicate with c-tree server (recommended only for diagnostic purposes)</p> <p>If omitted, <i>True</i> is assumed</p>
<code>iscobol.file.index.rpc.crc</code> (boolean)	<p>True = enable CRC checks on the RPC data buffers that are sent/received over the network. Recommended only for diagnostic purposes. False = disable CRC checks on the RPC data buffers that are sent/received over the network.</p> <p>If omitted, <i>False</i> is assumed</p>

Property	Description
<code>iscobol.file.index.server</code>	<p>This property specifies the server name and IP address. The value can be one of the following syntaxes:</p> <ul style="list-style-type: none"> • <code>servername</code> • <code>servername@hostname</code> • <code>servername@IPaddress</code> • <code>#port@hostname</code> • <code>#port@IPaddress</code> <p>If omitted, then <i>FAIRCOMS</i> is assumed.</p>
<code>iscobol.file.index.skiplock</code> (boolean)	<p>True = advance to the next record when a locked record is encountered. False = do not advance to the next record when a locked record is encountered.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.ssl</code> (boolean)	<p>True = use SSL (Secure Socket Layer) to connect to the c-tree Server. False = do not use SSL (Secure Socket Layer) to connect to the c-tree Server.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.sslcert</code>	<p>This property specifies the path and name of the pem file for SSL certificate.</p>
<code>iscobol.file.index.transaction</code> (boolean)	<p>True = transaction support is on. False = transaction support is off.</p> <p>If omitted, <i>True</i> is assumed.</p>
<code>iscobol.file.index.transaction.deferautocommit</code>	<p>True = turn on optimization that improves performance for functions that use autocommit. False = turn off optimization that improves performance for functions that use autocommit.</p> <p>If omitted, <i>False</i> is assumed.</p> <p>When set to <i>True</i>, it still guarantees atomicity and consistency of transaction but not durability because the last transaction could be lost.</p>
<code>iscobol.file.index.transaction.dependent</code>	<p>True = file operations are transaction dependent. False = file operations performed within an active transaction are not affected by the transaction ending operation (commit or abort).</p> <p>If omitted, <i>False</i> is assumed</p> <p>Note - this setting doesn't match with the 'dependent' information shown by the command <i>ctutil -info</i>.</p>

Property	Description
<code>iscobol.file.index.transaction.logging</code> (boolean)	<p>True = turns on transaction logging. It is indicated when data safety is more important than performance. Files are created with c-tree file mode <code>ctTRNLOG</code> and are automatically recovered after a crash.</p> <p>False = turns off transaction logging. It is indicated when performance is more important than data safety.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.trxholdlocks</code> (boolean)	<p>True = only the locks on records updated during the transaction are released at transaction commit while all other locks not specifically released are held. At transaction rollback instead records locks are held unconditionally.</p> <p>False = all pending locks are released during a transaction commit or rollback.</p> <p>If omitted, <i>False</i> is assumed.</p>
<code>iscobol.file.index.user</code>	<p>This property specifies the user name for the connection to the c-tree Server.</p> <p>If omitted, then <i>guest</i> is assumed</p>
<code>iscobol.file.index.versioncheck</code> (boolean)	<p>True = check if c-tree client and server versions match during the connection at runtime initialization.</p> <p>False = do not check if c-tree client and server versions match during the connection at runtime initialization.</p> <p>If omitted, <i>False</i> is assumed.</p>

Configuring multiple instances

It is possible to define multiple instances of the c-tree client in the same runtime session. Each instance can work on a different server and can have a specific configuration. For example you might want all files whose name starts with "TMP" to be treated as memory files, or you might want to distribute your archives on different c-tree servers depending on the directory where they reside.

In order to create multiple instances, you can repeat the above `iscobol.file.index` settings by putting the instance number between before the setting name. For example:

```
iscobol.file.index.server=FAIRCOMS
iscobol.file.index.1.server=FAIRCOMS
iscobol.file.index.2.server=FAIRCOMS@192.168.0.102
```

The above snippet defines three instances: the first two will connect to the default server on localhost and the third will connect a remote server whose IP address is 192.168.0.102.

Note that the `iscobol.file.index.maxinstance` property must be set accordingly or an error will occur. According to the above snippet, since we have three instances defined, we will set

```
iscobol.file.index.maxinstance=3
```

The runtime chooses which instance to use by checking the file name and path against the properties `iscobol.file.index.file_mapping` and `iscobol.file.index.path_mapping`.

Example:

```
iscobol.file.index.server=FAIRCOMS
iscobol.file.index.1.server=FAIRCOMS
iscobol.file.index.1.path_mapping=/tmp
iscobol.file.index.2.server=FAIRCOMS@192.168.0.102
iscobol.file.index.2.path_mapping=/remote
```

According to the above configuration, all files under the /tmp folder will be bound to instance number 1 and all files under the /remote folder will be bound to instance number 2. All the other files will be bound to the default instance.

Once multiple instances are defined, specific properties can be specified for each instance.

Example

```
iscobol.file.index.server=FAIRCOMS
iscobol.file.index.1.server=FAIRCOMS
iscobol.file.index.1.path_mapping=/tmp
iscobol.file.index.1.memoryfile=true
iscobol.file.index.1.transaction=no
iscobol.file.index.2.server=FAIRCOMS@192.168.0.102
iscobol.file.index.2.path_mapping=/remote
```

According to the above snippet, all the files on instance number 1 will be memory files without transaction support.

Note - The default instance includes a default file matching rule: `file_mapping=*`. This rule ensures that c-treeRTG can handle any possible file, including files that do not match any other rules. A default file matching rule is required when using multiple instances.

When setting configurations made by multiple properties (e.g. the log or the key compression), all the properties should be related to the instance. For example, the following configuration is not correct:

```
iscobol.file.index.1.log.file=/tmp/ctree.log
iscobol.file.index.log.error=1
iscobol.file.index.log.profile=1
iscobol.file.index.log.info=1
```

because only the file has been specified for the instance number 1, but all the other settings will go on the default instance and will be ignored by the instance number 1. The correct configuration would be:

```
iscobol.file.index.1.log.file=/tmp/ctree.log
iscobol.file.index.1.log.error=1
iscobol.file.index.1.log.profile=1
iscobol.file.index.1.log.info=1
```

Configuring the client through CTREE_CONF

Faircoms utilities always look for the c-tree configuration in the c-tree configuration file.

The isCOBOL Framework looks for the c-tree configuration in the c-tree configuration file if

- the c-tree version is less than 9.5.49790

- the property `iscobol.ctree.new_config (boolean)*` is set to false

The configuration file is defined by the environment variable `CTREE_CONF`. If `CTREE_CONF` is not defined then a XML file named `ctree.conf` is searched in the current directory. The XML configuration file `ctree.conf` is an XML file that maintains the same syntax for both Windows and Unix.

If the file cannot be found, the c-tree client library will use default settings, that means it will connect to a c-tree Server named FAIRCOMS on the localhost.

The XML configuration file is subject to a precise hierarchy as follows:

`<config>` root element : within it there can be zero or more global settings and zero or more `<instance>` elements. Within an `<instance>` element there can be zero or more global settings and zero or more `<file>` elements.

Setting elements specified as direct children of the `<config>` root element apply to all the `<instance>` elements specified as direct children of the `<config>` root element.

Setting elements specified as direct children of the `<instance>` element and inherited from the `<config>` root element apply to all the `<file>` elements specified as children of the current `<instance>` element.

Setting elements specified as direct children of `<file>` elements and inherited from the parent elements apply to any c-tree file matching the criteria defined by the current `<file>` elements.

Specified setting elements overwrite the inherited values.

Setting elements can be specified as children of CONF INST and FILE but actually apply only to file elements if a setting element is not specified as children of the file.

Option elements can be used inside the `<config>` root element, the `<instance>` and the `<file>` element to configure c-tree behaviors. The following option elements are available:

- `<automkdir>`
- `<batchaddition>`
- `<bulkaddition>`
- `<ctfixed>`
- `<datacompress>`
- `<datafilesuffix>`
- `<detectlock>`
- `<encrypt>`
- `<filepool>`
- `<fileversion>`
- `<forcedelete>`
- `<hugefile>`
- `<ignorelock>`
- `<indexfilesuffix>`
- `<keycheck>`
- `<keycompress>`
- `<log>`
- `<locktype>`
- `<maxlencheck>`
- `<map>`

- <memoryfile>
- <optimisticadd>
- <prefetch>
- <retrylock>
- <rowid>
- <rpc>
- <runitlockdetect>
- <skiplock>
- <smartcopy>
- <sqlize>
- <transaction>
- <trxholdslocks>
- <writethru>

Examples

In the following example the RPC setting will be active for all specified instances but not for instance number 3 which is specifying its own RPC setting.

```
<config>
  <rpc>yes</rpc>
  <instance server="FAIRCOMS">
    <file name="CUSTORDR"/>
    <file> </file> <!-- default file matching rule -->
  </instance>
  <instance server="FAIRCOMS@192.168.0.2">
    <file name="ITEMMAST"/>
  </instance>
  <instance server="FAIRCOMS@10.0.0.4">
    <rpc>no</rpc>
    <file name="CUSTMAST"/>
  </instance>
</config>
```

Note - The first instance includes a default file matching rule: <file> </file>. This rule ensures that c-treeRTG can handle any possible file, including files that do not match any other rules. Because it matches all files, it should be placed after all other rules. A default file matching rule is required when using multiple instances.

The following example is more simple. It just tells to use the custom extension ".cix" for the key files working on the c-tree server named FAIRCOMS and started on localhost (as default).

```
<config>
  <indexfilesuffix>.cix</indexfilesuffix>
</config>
```

<config>

<config> is the root element of the XML configuration file. It does not have any attributes and it's used only as a container of all other configuration elements.

<instance>

The instance element specifies instance-wide configurations for the client/server driver. Each instance represents a connection to the c-tree server.

Attributes

Attribute	Description	Default value
server	<p>Specifies the server name and the host name of the c-tree to connect to. The format can be one of the following syntaxes:</p> <ul style="list-style-type: none"> • servername • servername@hostname • servername@IPAddress • #port@hostname • #port@IPAddress <p>If the host name or the IP address is omitted, host name defaults to localhost.</p>	FAIRCOMS
user	Specifies the c-tree user name.	n/a
password	Specifies the c-tree user password.	n/a
connect	<p>Indicates whether to connect to c-tree at runtime initialization or wait for the first OPEN operation.</p> <p>It accepts the following values:</p> <p>"yes" : Indicates to connect at runtime initialization.</p> <p>"no" : Indicates to connect during the first OPEN operation. This is the default value.</p>	no
versioncheck	<p>Indicates whether to check that c-tree version matches the c-tree version. This option is turned off by default.</p> <p>It accepts the following values:</p> <p>"yes" : Turn on version matching check. Versions must match otherwise an error is returned at runtime initialization.</p> <p>"no" : Turn off version matching check. This is the default value.</p>	no
ssl	<p>Indicates whether or not an SSL connection should be established.</p> <p>It accepts the following values:</p> <p>"yes" : Perform an SSL connection.</p> <p>"no" : Perform a standard connection.</p>	n/a
sslcert	Specifies the path to a pem file for the SSL certificate.	n/a

Attribute	Description	Default value
endiancheck	<p>By default, the byte endianness is checked at connection time. When set to "no", that check can be skipped.</p> <p>This logic checks at connection time to ensure that the byte endianness of both client and server is the same (if it is not, the connection is terminated and the operation fails). In rare cases where this logic is not necessary (e.g., if files have no numeric data types), the byte endianness check can be skipped so the connection can succeed.</p>	yes
ctshmemdir	<p>Configure the c-treeRTG server to change the default shared memory directory.</p> <p>This setting is ignored under Windows as it is meaningful only for Unix platforms. On Unix platforms, the c-treeRTG server shares a directory with the c-treeRTG clients when they communicate via the shared memory protocol. It is possible to configure the c-treeRTG server to change the default shared memory directory with configuration keyword SHMEM_DIRECTORY. Using different shared memory directories allows running multiple c-treeRTG servers with the same name on the same machine. To configure the c-treeRTG clients to set the shared memory directory, it is necessary to set this attribute.</p>	n/a

Example

```
<!-- instance 1 on a c-tree server named COBOL running on a server machine whose IP
address is 192.168.0.2 -->
<instance server="COBOL@192.168.0.2" user="admin" password="ADMIN" connect="yes" versi
oncheck="no">

...

</instance>
<!-- instance 2 on a c-tree server running on a server machine whose IP address is
192.168.0.3 listening on port 1234 -->
<instance server="#1234@192.168.0.3" user="admin" password="ADMIN" connect="yes" versi
oncheck="no">

...

</instance>
```

<file>

The file element specifies file-wide configurations. It is used to delimit options to a specific file identified by the name attribute or to all the files contained in a directory identified by the dir attribute.

The name and dir setting may contain the * wildcard to match multiple files with a single section. See [Wildcard matching rules](#) below for details.

Attributes

Attribute	Description	Default value
name	Specifies the string to match the file name portion of the file path passed by the COBOL application.	n/a
dir	Specifies the string to match the directory portion of the file path passed by the COBOL application. If attribute name is also specified, the options applies to the files that match both the directory and the name.	n/a

Example

```
<file name="CUSTMAST" dir=".">

...

</file>
```

Wildcard matching rules

If the name/dir setting does not contain a *, the <file> options apply to the files that match exactly the name/dir setting.

If the name/dir setting begins with a *, the <file> options apply to the files for which name/dir ends with the string on the right of the *. For instance <file name="*mast">, matches file name "custmast" but does not match file name "master".

If the name/dir setting ends with a *, the <file> options apply to the files for which name/dir begins with the string on the left of the *. For instance <file name="mast*">, matches file name "master" but does not match file name "custmast".

If the name/dir setting begins with a * and ends with a *, the <file> options apply to the files which name/dir contains the string enclosed between the *. For instance <file name="*mast*">, matches file name "master" and file name "custmast".

If the name/dir setting contains only a *, the <file> options apply to all files.

In case a file matches multiple <file> rules, the following precedence rules apply:

An exact match has precedence over any wildcard match. For instance, file name "custmast" matches <file name="custmast"> and does not match <file name="cust*">.

In case of multiple wildcard matches, the one with most matching characters takes precedence: For instance, file name "custmast" matches <file name="cust*"> and does not match <file name="*ast">.

In case both name and dir are specified, the sum of the matching characters of both name and dir setting is considered.

In case the sum of matching characters is equal, the rule with the longest matching characters takes precedence: For instance, file name "custmast" matches <file name="cust*" dir="data"> and does not match <file name="*mast" dir=".">.

In case the lengths of the settings are equal, alphabetical order is used: For instance, file name "custmast" matches <file name="cust*" dir="."> and does not match <file name="*mast" dir=".">.

<automkdir>

The automkdir option indicates whether to automatically create missing directories when creating new files. If this option is set to yes, any missing directory of the file path is automatically created. If this option is set to no, an error is returned if the file path does not exist. This option is disabled by default.

Accepted Values

Value	Effect	Synonyms
yes	Missing directories are automatically created.	y, true, on, 1
no	An error is returned if path does not exist. This is the default value.	n, false, off, 0

Example

```
<automkdir>yes</automkdir>
```

<batchaddition>

The batchaddition option enables batch record writes to improve performance of consecutive record additions. This is achieved by caching the records being added in the client side before they are sent in one batch operation to the server to be written to disk. The records are sent to the server after a given number of records (specified by the records attribute) have been added or when the file is closed. Since the records are actually written when they are sent to the server, a duplicate key error is returned by the operation that triggered the batch write operation. For this reason, batchaddition is recommended only for specific operations such as data import where duplicate errors are not expected.

Writing records in batches improves performance of large record additions in environments where client and server reside on different systems connected with a network. In such environments any request from the client to the server is sent over the network which is generally a time expensive operation.

The batchaddition option is available only for files opened with OUTPUT or EXTEND mode. The number of records to cache is defined by the records attribute. The batchaddition option is disabled by default.

Accepted Values

Value	Effect	Synonyms
yes	Enable the record batch addition.	y, true, on, 1
no	Disable record batch addition. This is the default value.	n, false, off, 0

Attributes

Attribute	Description	Synonyms
records	Indicates the number of records to cache. This value is set to 100 by default.	recs

Example

```
<batchaddition records="50">yes</batchaddition>
```

<bulkaddition>

The bulkaddition option enables deferred key writes to improve performance of consecutive record additions. This is achieved by writing only the data record and postponing the key addition until the file is closed. When the file is closed, all pending keys are written in one single operation by an index rebuild routine. This technique of adding records has two benefits:

- 1) The index rebuild operation is faster than the sum of each key addition operation.
- 2) The index rebuild operation creates an index that is efficiently organized resulting in faster key searches.

The bulkaddition option is available only for files opened with OUTPUT or EXTEND mode. The bulkaddition option is disabled by default.

Accepted Values

Value	Effect	Synonyms
yes	Enable the bulkaddition technique.	y, true, on, 1
no	Disable the bulkaddition technique. This is the default value.	n, false, off, 0

Example

```
<bulkaddition>yes</bulkaddition>
```

<ctfixed>

The ctfixed option indicates whether to create fixed record length files with c-tree file mode ctFIXED. If this option is set to yes, files with fixed record length are created with c-tree file mode ctFIXED. If this option is set to no, all files are created with c-tree file mode ctVLENGTH. This option is disabled by default.

Setting ctfixed to yes imposes a minimum record length of 9 bytes for huge files or 5 bytes for non-huge files (based on the setting of the [<hugefile>](#) configuration option).

Accepted Values

Value	Effect	Synonyms
yes	Fixed record length files are created with ctFIXED file mode.	y, true, on, 1
no	All files are created with ctVLENGTH file mode. This is the default value.	n, false, off, 0

Example

```
<ctfixed>yes</ctfixed>
```

<datacompress>

The datacompress option indicates whether to create files with data compression enabled. When data compression is enabled, data records are compressed using the compression algorithm specified by the type attribute. Data compression reduces disk space utilization but it may also impact performance. This feature is turned off by default.

Accepted Values

Value	Effect	Synonyms
yes	Files are created with data compression enabled.	y, true, on, 1
no	Files are created without data compression. This is the default value.	n, false, off, 0

Attributes

Attribute	Description	Synonyms
type	Selects the type of compression. Values: "rle" : Use a simple RLE compression algorithm. "zlib" : Use the zlib compression algorithm. The default value is "rle".	n/a
strategy	Selects the compression strategy. Depending on the compression type the possible values for strategy are: Valid values for type "rle": "0" : Use the default simple RLE compression strategy. This is the default value for <i>type="rle"</i> . Valid values for type "zlib": "0" : Use the default zlib compression strategy. "1" : Use the zlib filtered compression strategy. "2" : Use zlib Huffman only compression strategy. "3" : Use zlib RLE compression strategy. This is the default value for <i>type="zlib"</i> . "4" : Use zlib fixed compression strategy.	n/a
level	Selects the compression level. Valid values are 0 and the range between 1 and 9: "0" : Use the compression algorithm default level. "1" : Provides best speed but uses more disk space "9" : Provides best compression at the expense of performance. The default value is "0".	n/a

Example

```
<datacompress type="zlib" strategy="3" level="9">yes</datacompress>
```

<datafilesuffix>

The datafilesuffix option defines the string to append to data file names. It can be used to define the default data file extension.

If not specified the default data file extension is ".dat".

Any string is accepted. Make sure your operating system file system accepts the value you specify.

Please remember to specify the dot if you want to use this as part of your file extension.

If you want your files to be created with no file extension, you can set datafilesuffix to a SPACE character as follows:

```
<datafilesuffix> </datafilesuffix>
```

Example

```
<datafilesuffix>.cdt</datafilesuffix>
```

<detectlock>

The detectlock option indicates whether to return an error trying to retrieve a record locked by another user if the READ operation does not have an explicit WITH LOCK option. This feature is turned off by default.

Accepted Values

Value	Effect	Synonyms
yes	the record locked error is returned	y, true, on, 1
no	the record locked error is not returned	n, false, off, 0

Example

```
<detectlock>yes</detectlock>
```

<encrypt>

The encrypt option indicates whether to create files with data encryption support enabled. When encryption is enabled, files are encrypted using c-tree ctCAMO encryption algorithm.

Accepted Values

Value	Effect	Synonyms
yes	Create file with data encryption support enabled.	y, true, on, 1
no	Create file with data encryption support disabled. This is the default value.	n, false, off, 0

Example

```
<encrypt>yes</encrypt>
```

<filepool>

In COBOL applications it is common practice to close and re-open files when entering procedures. This can cause unnecessary overhead and performance issues in c-treeRTG.

The Filepool feature introduces support for file pooling to keep files open when the COBOL application requests to close it. This allows the file handle to be returned immediately when the COBOL application request to re-open it.

The <filepool> global configuration keyword enables and disables support for file pooling and optionally sets the size of the file pool with attribute *size*.

The configuration keyword <inpool> defines if a file can be included in the file pool.

Note - <filepool> is client specific and not shared among clients. Consider user A adds a file to a filepool and user B attempts to operate on that file with ctutil -copy. This fails with error 12 (-8). Automatically removal from a pool during the copy can only be done for the user that requested the pool.

Attributes

Attribute	Description	Synonyms
size	Specifies the maximum number of files to keep in the pool.	n/a

Accepted Values

Value	Effect	Synonyms
yes	Enables the Filepool feature.	y, true, on, 1
no	Disables the Filepool feature. This is the default value.	n, false, off, 0

Example

```
<config>
  <filepool size="40">yes</filepool>
  <instance server="FAIRCOMS">
    <file>
      <inpool/>
    </file>
  </instance>
</config>
```

<fileversion>

The fileversion option can be set to a major version number of c-treeRTG (i.e. 2). The default value of fileversion is the major version number of the current c-treeRTG version (e.g. 3). If fileversion is specified in the configuration file, the default values of all the other configuration options are reset to the default value of the c-treeRTG version specified.

For example, if `<fileversion>2</fileversion>` is specified in a configuration file while running c-treeRTG version 3, the default value for the `<rowid size>` attribute is changed from 8 to 4. Also the default value of the `<keycompress vlnnod>` attribute is changed from 1 to 0 and the implicit value of `<keycompress/>` is changed from `<rle/>` to `<leading/><padding/>`.

Note - the compatibility with previous versions might be affected also by the [PAGE_SIZE](#) setting in the server configuration. Be sure to use the same page size between the current c-tree version and the previous c-tree version you want to be compatible with.

Accepted Values

Value	Effect	Synonyms
1	Create files compatible with c-tree v1 (all versions before v11)	n/a
2	Create files compatible with c-tree v2 (also known as v11)	n/a
3	Create files compatible with c-tree v3 (default)	n/a

Example

```
<fileversion>2</fileversion>
```

<forcedelete>

The isCOBOL default behavior of DELETE FILE is to return a "corrupted file" error and not delete the orphan file if the index file is missing. This configuration keyword allows to enable Micro Focus DELETE FILE behavior, which is to delete the orphan file.

Accepted Values

Value	Effect	Synonyms
yes	Allow the program to delete a file even if it's an orphan file.	y, true, on, 1
no	Return a "corrupted file" error if the program tries to delete an orphan file. This is the default value.	n, false, off, 0

Example

```
<forcedelete>yes</forcedelete>
```

<hugefile>

This configuration keyword allows the huge file mode to be turned off so that non-huge files are created. The default is to use c-tree support for huge files, which accommodate up to 16 exabytes of data when segmented files are used. Non-huge files accommodate up to 2 or 4 gigabytes of data depending on the operating system.

Accepted Values

Value	Effect	Synonyms
yes	Create huge file (c-tree files larger than four gigabytes). This is the default value.	y, true, on, 1
no	Create non-huge file.	n, false, off, 0

Example

```
<hugefile>yes</hugefile>
```

<ignorelock>

The ignorelock option specifies whether READ operations should ignore locks. This option is turned off by default.

Accepted Values

Value	Effect	Synonyms
yes	Turns off record lock detection.	y, true, on, 1
no	Turns on record lock detection. This is the default value.	n, false, off, 0

Example

```
<ignorelock>yes</ignorelock>
```

<indexfilesuffix>

The indexfilesuffix option defines the string to append to index file names. It can be used to define the default index file extension.

If not specified the default data file extension is ".idx".

Any string is accepted. Make sure that your operating system file system accepts the value you specified.

Please remember to specify the dot if you want to use this as part of your file extension.

If you want your files to be created with no file extension, you can set indexfilesuffix to a SPACE character as follows:

```
<indexfilesuffix> </indexfilesuffix>
```

Example

```
<indexfilesuffix>.cix</indexfilesuffix>
```

<keycheck>

The keycheck option specifies whether to check that all the keys belonging to an indexed file are defined in the COBOL program. If this option is set to yes, the keys must match the key definitions otherwise an error is returned. If this option is set to no, files that contain more keys than described in the COBOL program are allowed. Files that contain fewer keys than the key definitions are never allowed regardless of the keycheck setting. This feature is enabled by default.

Accepted Values

Value	Effect	Synonyms
yes	All key definitions must match otherwise an error is returned during OPEN operations. This is the default value.	y, true, on, 1
no	Open the file even if it contains more indexes than the key definitions.	n, false, off, 0

Example

```
<keycheck>no</keycheck>
```

<keycompress>

The keycompress option indicates whether to create files with key compression enabled. This feature is turned off by default. It may impact performance but reduces disk space usage.

Accepted Values

Value	Effect	Synonyms
yes	Turns on key compression combining both leading character and padding compression. This provides the maximum key compression.	y, true, on, 1
no	Turns off key compression.	n, false, off, 0

Additionally the keycompress option may accept the following sub-options to specify which compression type to use:

Option	Description
<leading>	Indicates to compress the leading characters of key values.

Option	Description
<padding>	Indicates to compress the padding characters of key values.
<rle>	Indicates to use RLE compression algorithm.

Attributes

Attribute	Description	Default value
vlennod	Use the new RLE compression or the legacy LEADING compression. Values: "yes" : Enable RLE, disable LEADING. "no" : Disable RLE, enable LEADING.	yes

Examples

The following example turns on default key compression that implicitly uses leading and padding compression:

```
<keycompress>yes</keycompress>
```

The following example turns on padding key compression only:

```
<keycompress>
  <padding>yes</padding>
</keycompress>
```

<log>

The log option indicates whether to log events such as errors that occur in c-tree. This feature might be helpful for diagnostics purposes.

Attributes

Attribute	Description	Synonyms
file	Specifies the log file name. If omitted the log messages are redirected to the standard error stream (stderr).	n/a

Accepted Values

Value	Effect	Synonyms
yes	Turns on logging of errors and generic information.	y, true, on, 1
no	Turns off logging. This is the default value.	n, false, off, 0

Additionally the log option may accept the following sub-options to specify which event to log:

Option	Description
<debug>	Indicates to log debug information. The following children elements are supported: <config>: include explicit configuration in the log <config full="yes">: include full configuration in the log
<error>	Indicates to log errors. The following children elements are supported: <atend>: log EOF errors <notfound>: log NOTFOUND errors <duplicate>: log DUPLICATE errors
<info>	Indicates to log generic information.
<profile>	Indicates to log performance profiling information.
<warning>	Indicates to insert a warning into the log file.

Examples

The following example turns on implicit logging of errors and generic information to standard error stream:

```
<log>yes</log>
```

The following example turns on explicit logging of errors and generic information to file *mylog.txt*:

```
<log file="mylog.txt">  
  <error>yes</error>  
  <info>yes</info>  
</log>
```

<locktype>

The locktype option allows to configure if the record data is returned to the program even if the record is locked.

Note - this setting is ignored by the c-tree File Connector (fscsc).

Accepted Values

Value	Effect	Synonyms
0	Locked records are returned.	n/a
1	Locked records are not returned. This is the default.	n/a

Example

```
<locktype>0</locktype>
```

<maxlencheck>

The maxlencheck option specifies whether to check that the record being read from disk fits entirely into the record buffer for which length is equal to the maxlen file definition. If this option is set to yes, an error is returned if the record read from disk is larger than maxlen bytes. If this option is set to no, the record is truncated at maxlen bytes before it is returned to COBOL. This option is enabled by default.

Accepted Values

Value	Effect	Synonyms
yes	Return an error if record is larger than maxlen bytes. This is the default value.	y, true, on, 1
no	Return record truncated at maxlen bytes.	n, false, off, 0

Example

```
<maxlencheck>no</maxlencheck>
```

<map>

The map option replaces the file name or directory passed by the COBOL application with the specified values.

The map option may contain the following sub-options:

Option	Description
<name>	Specifies the string that replaces the file name portion of the file path passed by the COBOL application.
<dir>	Specifies the string that replaces the directory portion of the file path passed by the COBOL application.

Examples

The following example replaces only the file name portion of the passed file path:

```
<map>  
  <name>CUSTMAST2010</name>  
</map>
```

The following example replaces both the name and directory portion of the passed file path:

```
<map>  
  <name>CUSTMAST2010</name>  
  <dir>/Data2010</dir>  
</map>
```

<memoryfile>

The memoryfile option indicates whether to create files in memory rather than on disk. This feature is indicated for temporary files. This feature is turned off by default.

Accepted Values

Value	Effect	Synonyms
yes	Create file as memory file.	y, true, on, 1
no	Create file as disk file. This is the default value.	n, false, off, 0

Attributes

Attribute	Description	Synonyms
persist	By default memory files are kept in memory until the c-tree Server is shut down. By disabling this attribute (persist="no") memory files created by this client session will be removed when the session terminates.	n/a

Example

```
<memoryfile>yes</memoryfile>
```

<optimisticadd>

The optimisticadd option specifies the strategy used by c-tree to add new records. This feature is turned on by default.

Accepted Values

Value	Effect	Synonyms
yes	c-tree uses an optimistic strategy which provides optimal performance when adding non-existing records. This is the default value.	y, true, on, 1
no	c-tree uses a pessimistic strategy that is attempts to add unique keys before adding the data record.	n, false, off, 0

Example

```
<optimisticadd>no</optimisticadd>
```

<prefetch>

The prefetch option enables batch record retrieval to improve performance of consecutive sequential reads. This is achieved by retrieving a given number of records (specified by the records attribute) from the server to the client side at the second consecutive sequential read operation. The next sequential read operations do not need to contact the server to retrieve the records as the records are now cached on the client side. Once all the records in the client cache are processed, the next block of records is requested to the server. By default, the records cached on the client side cannot be updated by other users (the allowwriters attribute overrides this default). This avoids situations in which the cached records do not match the records on the server.

Prefetching records improves performance of sequential reads in environments where the client and server reside on different systems connected with a network. In such environments any request from the client to the server is sent over the network which is generally a time-expensive operation.

The number of records to prefetch is defined by the records attribute.

Accepted Values

Value	Effect	Synonyms
yes	Enable record prefetch.	y, true, on, 1
no	Disable record prefetch. This is the default value.	n, false, off, 0

Attributes

Attribute	Description	Synonyms
records	Indicates the number of records to prefetch. This value is set to 10 by default.	recs
allowwriters	<p>When a set of records is prefetched, the prefetched records are locked to prevent other users from updating them. When this attribute is enabled, the read lock is not used so that other users can alter the prefetched records.</p> <p>This attribute is disabled by default (prefetch allowwriters="no").</p> <p>Note - It is the programmer's responsibility to ensure that the application does not have issues with prefetched records that have been modified. Enable allowwrites only when data is not critical and/or seldom changes.</p>	writers
ttl	Indicates the number of seconds prefetched records are kept in memory.	n/a

Examples

To enable prefetch and set the number of prefetched records to 50:

```
<prefetch records="50">yes</prefetch>
```

To enable prefetch and allow prefetched records to be modified by other users:

```
<prefetch allowwriters="yes">yes</prefetch>
```

<retrylock>

The retrylock option indicates whether a READ operation that fails because a record is locked should wait until the lock is released rather than returning an immediate lock error. When this option is enabled, records are locked with the c-tree lock mode of ctENABLE_BLK instead of ctENABLE.

Accepted Values

Value	Effect	Synonyms
yes	Wait until the lock is released.	y, true, on, 1
no	Do not wait and return immediately with an error. This is the default value.	n, false, off, 0

Note - When this option is explicitly set in the configuration file, it overrules the WAIT LOCK clause in READ operations. READ WAIT LOCK statements will return a "record locked" error if retrylock is set to n, false, off or 0 in the configuration file.

Example

```
<retrylock>yes</retrylock>
```

<rowid>

The <rowid> option enables support for creating a ROWID field/index. ROWID is required for creating Full-Text Indexes, so this option is necessary if Full-Text Search is desired.

Accepted Values

Value	Effect	Synonyms
yes	Enable support for ROWID field/index.	y, true, on, 1
no	Disable support for ROWID. Default for c-treeRTG COBOL Edition.	n, false, off, 0

Attributes

Attribute	Description	Default value
size	Sets the size of the rowid hidden field in the record header. The default value is 8, accepted values are 8 and 4.	8

Example

```
<rowid>yes</rowid>
```

<rpc>

The rpc option indicates whether to use standard c-tree client/server calls or c-tree RPC calls. Using RPC calls results in better performance due to an optimized communication protocol that reduces network traffic. This feature is turned on by default.

Accepted Values

Value	Effect	Synonyms
yes	Use RPC calls to communicate with c-tree.	y, true, on, 1
no	Use standard c-tree client/server calls to communicate with c-tree. This option is recommended only for diagnostics purposes.	n, false, off, 0

Attributes

Attribute	Description	Default value
crc	Indicates whether to enable CRC checks on the RPC data buffers that are sent/received over the network. It is recommended only for diagnostic purposes to detect faulty networks. Values: "yes" : Turns on CRC checks. "no" : Turns off CRC checks.	no

Example

```
<rpc crc="no">yes</rpc>
```

<runitlockdetect>

The runitlockdetect option specifies whether to check if a record or file has been locked by a separate OPEN statement issued from within the same run unit. By default it is permitted to lock a record multiple times within the same instance. When this option is turned on, a lock error is returned while reading a record that was locked within the same instance or by opening in exclusive mode a file that was locked within the same instance.

Accepted Values

Value	Effect	Synonyms
yes	Detect record locks. This is the default value.	y, true, on, 1
no	Do not detect record locks.	n, false, off, 0

Attributes

Attribute	Description	Default value
anyunlock	When set to 'yes', any UNLOCK performed in the run unit on a file will remove the lock on the file even if the record was locked by another OPEN instance. Otherwise, the lock is removed only if the UNLOCK is performed on the same OPEN instance where the READ WITH LOCK was performed.	no

Example

```
<runitlockdetect anyunlock="no">no</runitlockdetect>
```

<skiplock>

The skiplock option advances the current record pointer when a locked record is encountered. This feature is off by default.

Accepted Values

Value	Effect	Synonyms
yes	Advance the current record pointer when a locked record is encountered.	y, true, on, 1
no	Do not advance the current record pointer. This is the default value.	n, false, off, 0

Example

```
<skiplock>yes</skiplock>
```

<smartcopy>

The smartcopy option enables the use of batched read/write techniques to improve performance of file copy operations. If this option is set to yes, the file copy reads and writes records in blocks in order to reduce the number of read and write operations. If this option is set to no, the records are copied one by one. This option is enabled by default.

Accepted Values

Value	Effect	Synonyms
yes	Records are copied in blocks. This is the default value.	y, true, on, 1
no	Records are copied one by one.	n, false, off, 0

Example

```
<smartcopy>yes</smartcopy>
```

<sqlize>

The sqlize option indicates whether to attempt linking the table to c-treeSQL. If this option is set to yes, the necessary operations to make the file accessible from c-treeSQL are performed when the file is open with OUTPUT or EXTEND mode. This option is disabled by default.

Accepted Values

Value	Effect	Synonyms
yes	Files opened with OUTPUT or EXTEND mode are linked to c-treeSQL.	y, true, on, 1
no	No attempt to link table is made. This is the default value.	n, false, off, 0

Attributes

Attribute	Description	Synonym
xfd	Path to ISS data definition file. If a directory is specified, a file with same name but ".iss" extension is searched.	n/a
database	Database name to add the file to. The default value is "ctreeSQL".	db
password	c-tree ADMIN password. The default value is "ADMIN".	pw
symbolic	Optional table name to use when adding file to a database.	n/a
prefix	Optional prefix for table or symbolic name.	n/a
owner	Optional user name to assign table ownership.	n/a
public	Optionally grant public access permissions. Values: "yes" : Grant public permissions. "no" : Turns off CRC checks.	n/a
numformat	Numeric format digit ID. Values: "A" : Set numeric format to ACUCOBOL type. "D" : Set numeric format to Data General type. "I" : Set numeric format to IBM type. "M" : Set numeric format to Micro Focus type. The default value is "A".	n/a

Example

```
<sqlize xfd="custmast.iss" symbolic="customers">yes</sqlize>
```

<transaction>

The transaction option indicates whether to create files with transaction support enabled. This feature is turned on by default.

Please note that it is possible to enable/disable transaction support and transaction logging on existing files using the `-tron` option of the ctutil utility.

Accepted Values

Value	Effect	Synonyms
yes	Turns on transaction support. This is the default value.	y, true, on, 1
no	Turns off transaction support.	n, false, off, 0

Attributes

Attribute	Description	Default value
logging	Enable/disable transaction logging. Values: "yes" : Turns on transaction logging. It is indicated when data safety is more important than performance. Files are created with c-tree file mode ctTRNLOG and are automatically recovered after a crash. "no" : Turns off transaction logging. It is indicated when performance is more important than data safety. Files are created with c-tree file mode ctPREIMG. This is the default value.	no
fileoperations (or fileops)	Determines whether file operations (such as file create, delete, and rename) performed within an active transaction are affected by the transaction ending operation (commit or abort): "yes" : File operations are transaction dependent. "no" : File operations performed within an active transaction are not affected by the transaction ending operation (commit or abort).	no
deferautocommit	Turn on optimization that improves performance for functions that use autocommit. Similar to the c-tree DELAYED_DURABILITY keyword, guarantees atomicity and consistency of transaction but not durability because the last transaction could be lost.	no

Example

```
<transaction logging="no">yes</transaction>
```

<trxholdslocks>

The `trxholdslocks` option indicates whether to hold or release locks during a transaction commit or rollback. This feature is turned off by default.

It is a global-only option and can be set only as a child of `<config>`. It is not valid if specified as child of `<instance>` or `<file>`.

Accepted Values

Value	Effect	Synonyms
yes	Only the locks on records updated during the transaction are released at transaction commit while all other locks not specifically released are held. At transaction rollback instead records locks are held unconditionally.	y, true, on, 1
no	All pending locks are released during a transaction commit or rollback. This is the default value.	n, false, off, 0

Example

```
<trxholdslocks>yes</trxholdslocks>
```

<writethru>

The `writethru` option specifies whether to enable write synchronization.

Accepted Values

Value	Effect	Synonyms
yes	Turn on write synchronization.	y, true, on, 1
no	Turns off write synchronization. This is the default value.	n, false, off, 0

Example

```
<writethru>yes</writethru>
```

Chapter 6

Accessing from isCOBOL

isCOBOL uses the c-tree interface in one of the following cases:

1. when the runtime framework property [iscobol.file.index](#) is set either to "ctreej"; or
2. when the runtime framework property [iscobol.file.index.FileName](#) is set either to "ctreej"; or
3. when CLASS "com.iscobol.io.DynamicCtreeJ" is specified in the [SELECT Clause](#).

The isCOBOL Framework becomes a client process that connects to a listening c-tree Server. To provide this connection, isCOBOL loads the ctree client library (ctree.dll on Windows, libctree.so on Linux) which must be available in the library path and must be the same version of the listening c-tree Server. A copy of this library is provided along with isCOBOL and it's of the same version as the latest c-tree server available for download at Veryant's web site. The library is installed in the bin folder on Windows and in the native/lib folder on Linux/Unix). You can replace this library if you need to work with a different version of c-tree.

isCOBOL can create c-tree files directly into the c-tree SQL Server so they are available as database tables for external SQL tools. Consult [Creating c-tree files into c-tree SQL Server from the COBOL Program](#) chapter for details.

In order to bring existing c-tree files into the SQL Server, instead, the [ctutil](#) utility must be used. Consult [Registering existing c-tree files in c-tree SQL Server](#) chapter for details.

Accessing through a File Connector

The c-tree File Connector allows you to work on c-tree files managed by c-tree by separating ISAM native access from java process. This solution is suggested when working with third party application servers (e.g. Apache Tomcat).

isCOBOL uses the file connector interface in one of the following cases:

1. when the runtime framework property [iscobol.file.index](#) is set either to "fscsc"; or
2. when the runtime framework property [iscobol.file.index.FileName](#) is set either to "fscsc"; or
3. when CLASS "com.iscobol.io.DynamicConnector" is specified in the [SELECT Clause](#).

Accessing previous c-tree versions

Client library

isCOBOL includes a copy of the c-tree client library in the *bin* directory on Windows and in the *native/lib* directory on Linux/Unix. This library is always up to date with the latest c-tree server distributed by Veryant. If it doesn't suit your needs, consider replacing this library with the one found in the c-tree server's installation, under the *drivers/ctree.cobol* folder, with the following criteria.

Until version 11.9 a c-tree client library was able to connect to a c-tree server of an older version, as long as the major release is the same. Since version 11.9, the default *ctree.cobol/iscobol* is no longer able to connect to c-tree servers of a previous version (e.g. 11.6).

If you need to connect to a c-tree server of a previous version, use the library installed under *drivers/ctree.cobol/iscobol.noGT32Kfiles* instead.

If you need to connect to a c-tree server version 10 or lower, then use the c-tree client library installed along with that server and use the *ctree2* file handler rather than the *ctreej* file handler.

File handler class

The available file handler classes and their aliases may be different when working with previous c-tree versions. The following table lists the file handlers available for the various c-tree versions:

c-tree version	available file handlers	
	values for <i>iscobol.file.index</i>	classes usable in FILE-CONTROL
9.5.51961 or previous	• <i>ctree</i>	• <i>com.iscobol.io.DynamicCtree</i>
between 9.5.53702 and 10.4.0.39110	• <i>ctree2</i>	• <i>com.iscobol.io.DynamicCtree2</i>
	• <i>fscsc</i>	• <i>com.iscobol.io.DynamicConnector</i>
10.4.0.39701 or later	• <i>ctreej</i> ^[A]	• <i>com.iscobol.io.DynamicCtreeJ</i> ^[A]
	• <i>ctree2</i> ^[B]	• <i>com.iscobol.io.DynamicCtree2</i> ^[B]
	• <i>fscsc</i>	• <i>com.iscobol.io.DynamicConnector</i>

^[A] Supported since isCOBOL 2016 R1.

^[B] Deprecated since isCOBOL 2016 R1.

The URL syntax

It is possible to specify c-tree Server connection information in the physical file name through the URL syntax as follows:

```
ctree://userID:password@hostname:servername/filename?config1?config2...?configN
```

where:

- *userID* is the c-tree user ID.
- *password* is the c-tree user password.
- *hostname* is the host name or IP address of the machine running the c-tree Server.
- *servername* is the name of the c-tree Server to connect to. As an alternative, you can specify the server port with the syntax *#port* (see snippets below for details).
- *filename* is the name of the file to open. It can be either a full path or a path relative to c-tree Server current directory.
- *config1* thru *configN* are optional configuration keywords

Examples

A typical example based on a default installation:

```
ctree://admin:ADMIN@localhost:FAIRCOMS/myfile
```

userID and password can be omitted:

```
ctree://localhost:FAIRCOMS/myfile
```

the port can be specified instead of servername:

```
ctree://localhost:#5597/myfile
```

servername can also be omitted in which case the default "FAIRCOMS" is used:

```
ctree://localhost/myfile
```

hostname can also be omitted in which case the default "localhost" is used:

```
ctree://:FAIRCOMS/myfile
```

If both servername and hostname are omitted, the syntax is used as follows:

```
ctree:///myfile
```

URL in File Prefix

The URL syntax can be used also as [iscobol.file.prefix](#) value.

Suppose that your c-tree files reside in the folder C:\data on the server machine Server1 where the c-tree server is running.

In this case the configuration should include a similar entry:

```
iscobol.file.prefix=ctree://admin:ADMIN@Server1:FAIRCOMS/C:/data
```

SQL Features

c-tree indexed files can be accessed via SQL after being "sqlized".

Registering existing c-tree files in c-tree SQL Server

In order to work on c-tree files via SQL, it's necessary to link the files into the c-tree SQL database. The ctutil utility provides a function for this purpose: [-sqlize](#).

The first step is to create special dictionary files for the archives that need to be installed in the c-tree SQL Server. In order to create the dictionary files, COBOL programs containing file descriptions must be compiled with [-efc](#) option.

For example:

```
iscc -efc myProg.cbl
```

The isCOBOL Compiler will generate a file named *<filename>.iss* for each file described in the COBOL program.

Note -c-tree SQL doesn't allow identifiers whose name begins with underscore. If one of the field names in your FD begins with underscore, you should change that name for the iss dictionary by using the EFD [NAME Directive](#).

Note - the way numeric fields are described in the dictionary is affected by the configuration property [iscobol.compiler.efc_field_name_num](#) (boolean).

When dictionary files are available, the following command can be launched:

```
ctutil -sqlize c-tree-file iss_file database_name [-sign=convention] [-symb=table_name]
```

Where:

- *c-tree-file* is the name of the physical c-tree file. This name is resolved server-side, so if a relative path is specified, the path will be relative to the ctrsvr working directory.
- *iss-file* is the name of the iss file generated by the isCOBOL Compiler. This file is loaded client-side, if a relative path is specified, the path will be relative to the ctutil working directory.
- *database_name* is the name of the database in which the file must be installed. c-tree SQL provides a default database named ctreesQL. The user can create new databases using DBA utilities.
- *convention* specifies the data convention to be used for the c-tree file. Use "A" if COBOL programs are compiled with -dca or without options. Use "M" if programs are compiled with -dcm, use "I" for -dci and "D" for -dcd. If omitted, "A" is assumed.
- *table_name* is an optional parameter that allows to specify a different name for the table on the database. By default the table on the database has the same name of the c-tree file.

For example:

```
ctutil -sqlize file1 file1.iss ctreesQL
```

The administrator password is required.

The administrator password is included in the configuration file.* This file can be:

- in the working directory in a file named *ctree.conf*
- in the directory specified in the environment variable CTREE_CONF
- passed to ctutil using the -c option in the command line

This configuration file should contain this statement:

```
<config>
  <instance server="FAIRCOMS" user="admin" password="ADMIN">
  </instance>
</config>
```

Other statements can be added to the configuration file as needed.

*To be compatible with older c-tree versions that allowed you to pass the administrator password in the

command line, you can add the new configuration variable `COMPATIBILITY SQLIMPORT_ADMIN_PASSWORD` to the c-tree Server configuration file (`ctsrvr.cfg`). However, adding the admin and password to the configuration file as shown above is considered the best practice.

The `sqlize` option has the same effect as running in sequence the following deprecated options:

<code>-sqlunlink</code>	Unlink the file from the c-tree SQL database. The file is no more available as table for SQL clients despite it still exists on disk.
<code>-sqlinfo</code>	Include the iss dictionary in the c-tree DAT file header.
<code>-sqlllink</code>	Link the file to the c-tree SQL database. The file is now available as table for SQL clients.

The above options are still supported for backward compatibility and are documented in the [ctutil](#) manual.

How to deal with homonymous files

When working with ISAM disc files you can have the same file name in different paths. Let's think about a multi-company application that keeps the history of some archives among the years. On disc you could find:

- `compay1\2020\invoices.dat`
- `compay1\2021\invoices.dat`
- `compay2\2020\invoices.dat`
- `compay2\2021\invoices.dat`
- etcetera

In order to reflect this situation in SQL, you might consider having multiple databases (one database per company) and use a prefix before the table name (use the year as prefix).

To create additional databases in c-treeSQL you can use either the [c-treeSqlExplorer](#) graphical utility or the [ctsqlcdb](#) command line utility.

Assuming that you have created two databases, `DBCompany1` and `DBCompany2`, you can `sqlize` the above archives as follows:

```
ctutil -sqlize /path/to/company1/2020/invoices invoices.iss DBCompany1 -prefix=2020_  
ctutil -sqlize /path/to/company1/2021/invoices invoices.iss DBCompany1 -prefix=2021_  
ctutil -sqlize /path/to/company2/2020/invoices invoices.iss DBCompany2 -prefix=2020_  
ctutil -sqlize /path/to/company2/2021/invoices invoices.iss DBCompany2 -prefix=2021_
```

Creating c-tree files into c-tree SQL Server from the COBOL Program

In order to create new files directly in c-tree SQL Server, the following property must be set in the configuration:

```
iscobol.sqlserver.iss=1
```

Only c-tree files will be affected, all other indexed files will not.

Setting the `iscobol.sqlserver.iss` (boolean) to true is not enough. `isCOBOL` needs to locate the iss dictionary for

the file.

In order to creating the iss dictionary files, COBOL programs containing file descriptions must be compiled with `-efc` option.

For example:

```
iscc -efc myProg.cbl
```

The isCOBOL Compiler will generate a file named `<filename>.iss` for each file described in the COBOL program.

Note - c-tree SQL doesn't allow identifiers whose name begins with underscore. If one of the field names in your FD begins with underscore, you should change that name for the iss dictionary by using the EFD [NAME Directive](#).

The path where iss files are located is specified through the following property in the configuration:

```
iscobol.sqlserver.isspath=path_where_iss_files_are_located
```

After isCOBOL has read the file information from the iss file, it needs to connect to the c-tree SQL database. The database name and the user credentials are passed through the following properties in the configuration:

```
iscobol.sqlserver.database=database_name  
iscobol.file.index.user=user_name  
iscobol.file.index.password=user_password
```

Note - If COMPATIBILITY SQLIMPORT_ADMIN_PASSWORD is present in the c-tree Server configuration (ctsrvr.cfg) then `iscobol.sqlserver.password=user_password` should be set instead of `iscobol.file.index.user` and `iscobol.file.index.password`. Note that the use of COMPATIBILITY SQLIMPORT_ADMIN_PASSWORD is discouraged for security reasons.

Among your indexed files there could be some files with the same record definition but different file name. In this case, to avoid creating an iss dictionary for each one of them, you can associate more files to the same iss by creating a mapping.

```
iscobol.sqlserver.iss.mapping.physical_name_pattern=iss_basename
```

Wildcards are supported.

For example, suppose you have a file named "customer" for each year, so on disc there are "customer2009", "customer2008", "customer2007", etc. If their FD is the same, you just have to create customer.iss and then set:

```
iscobol.sqlserver.iss.mapping.customer*=customer
```

Another important issue that should be considered is the sign convention. Ensure that the property [iscobol.sqlserver.convention](#) is set to the proper value depending on the data compatibility options in your compiler command line.

Compiler option	sqlserver.convetion
-dca	A
-dcd	D

-dci	I
-dcm	M
-dcn	N
-dcr	R

It's possible to include part of the file path in the table name by setting `iscobol.sqlserver.dirlevel` to a value different than zero. The following table shows some examples:

physical file name: /home/user1/data/file1	
dirlevel	table name on c-tree SQL database
0	file1
1	datafile1
2	user1datafile1

It's possible to customize the name of the table by configuring the replacement performed on the physical file name through the property `iscobol.sqlserver.iss.replacement_rules`. The following table shows some examples:

physical file name: /home/user1/data/f-cust.cdt	
rules	table name on c-tree SQL database
0	f_cust_cdt
1	f_custcdt
2	fcust_cdt
3	fcustcdt

This is a sample configuration that summarizes all the above settings.

```
iscobol.file.index=ctreej
iscobol.sqlserver.iss=1
iscobol.sqlserver.convention=A
iscobol.sqlserver.isspath=/develop/iss
iscobol.sqlserver.database=ctreeSQL
iscobol.file.index.user=admin
iscobol.file.index.password=ADMIN
iscobol.sqlserver.iss.mapping.customer*=customer
iscobol.sqlserver.dirlevel=0
```

Note - `iscobol.sqlserver.iss`, `iscobol.sqlserver.isspath` and `iscobol.sqlserver.database` must always be set, otherwise the automatic link procedure is not performed.

Once all the properties have been correctly set in the configuration, you can create the file in c-tree SQL

Server by just opening it for output in the program.

```
OPEN OUTPUT logical_file_name.
```

EFD directives and mapping rules

The definition of the table is based on the extended file description (ISS) generated by the Compiler.

In order to generate ISS dictionaries, compile the program with the [-efc](#) compiler option.

Mapping rules

The sqlize process will map only elementary fields to table columns and only the larger record type when there are multiple record definitions. Default rules used to map COBOL fields to table columns can be changed by using EFD directives.

Default Rules

COBOL file name	RDBMS table
COBOL record	table row
COBOL field	table column
COBOL key	table index

Redefines must be managed using the [WHEN Directive](#) or they will be hidden to SQL.

FILLER data items are not mapped to table columns. You can overwrite this behavior using the [NAME Directive](#) in order to associate a column name with COBOL FILLER.

Cobol group items are not mapped to table columns, instead they are mapped to elementary fields of the group. You can change this approach using the [USE GROUP Directive](#).

There is not a corresponding RDBMS table definition for the COBOL OCCURS statement. For this reason the default behavior is to append a sequential number to the field name. For example:

```
01 myoccurs occurs 10 times.  
  03 customer-code pic 9(5).  
  03 customer-name pic x(30).
```

Will be mapped in the table as

```
customer_code_1  
customer-name_1  
...  
...  
customer_code_3  
customer-name_3  
customer_code_4  
customer-name_4  
customer_code_5  
customer-name_5  
...  
...
```

Using EFD directives

Directives are used when a COBOL file descriptor is mapped to a database field. The \$EFD prefix indicates to the compiler that the proceeding command is used during the generation of the data dictionary.

Consult [EFD Directives](#) for a detailed description of the available EFD directives.

Invalid Data

Not all COBOL data is valid for SQL rules, because COBOL allows you to store any value in any field while SQL expects that the value is consistent with the field type. For this reason, for example, spaces or low-values in Numeric and Date fields are not a problem in COBOL, but generate a conversion error in SQL.

Invalid data is managed by the c-tree SQL engine as follows: when invalid data is found in a field, NULL is returned for that field.

Accessing c-tree files through SQL from the COBOL Program

The COBOL programs can access c-tree SQL database in two ways.

- Using standard COBOL i-o statements.
This is the same approach used for the standard isCOBOL ISAM Server. The following configuration property must be set:

```
iscobol.file.index=ctreej
```

- Using Embedded SQL.
For this approach Veryant suggests that you use the official JDBC driver installed along with c-tree.

The *ctreeJDBC.jar* library must appear in the CLASSPATH and the following settings are necessary in the configuration:

```
iscobol.jdbc.driver=ctree.jdbc.ctreeDriver  
iscobol.jdbc.url=jdbc:ctree:6597@localhost:
```

Where *localhost* and *6597* are ip-address and port on which the SQL Server is listening.

Then, from the COBOL program:

```
exec sql connect to 'ctreeSQL' user 'admin' using 'ADMIN' end-exec.
```

Where *ctreeSQL* is the name of the database and *admin/ADMIN* are user and password.

Multithread programs

The runtime session's connection to the c-tree server is made when the first file is opened. This connection is bound to the life of the thread that opened this first file. Therefore, it's important that the thread that performs the first open of a c-tree file stays alive for the whole runtime session.

Consider the following sample situation: There is a main menu program from which the user can call subprograms. The subprograms are each called in a thread so that the user can switch from one to another any time. The subprograms perform i/o on c-tree files. In a situation like this, the user might perform the following steps:

1. Run the menu program as thread #1.
2. Launch a subprogram as thread #2 (e.g. Customer Handling).
3. Launch another subprogram as thread #3 (e.g. Statistics).
4. Close the first subprogram (thread #2), leaving only thread #1 and #3 active.

In this situation, if Customer Handling were to open the first file, then the connection to the c-tree server would be lost when Customer Handling was closed, and the other programs running in the runtime session, like Statistics, will encounter i/o errors.

The menu program must open the first file to create the connection to the c-tree server, as it is the only program that will stay alive for the whole runtime session. A quick and easy workaround to this problem would be to open and close a file in the menu program to establish the connection to the c-tree Server, for instance:

```
open input a-file.  
close a-file.
```

Good practice for transactions management

c-treeRTG supports transactions management via the START TRANSACTION, COMMIT and ROLLBACK statements. These statements don't trigger a connection to the c-tree server, though.

Although it's good practice to start a transaction only after the files have been opened, some programmers prefer to start the transaction before opening the files, e.g.:

```
start transaction.  
open i-o file1.
```

The START TRANSACTION in the above snippet may fail if the connection to c-tree hasn't been acquired yet. Only the OPEN statement triggers a connection to the c-tree server. The same workaround explained in [Multithread programs](#) can be applied, for instance:

```
open input a-file.  
close a-file.  
start transaction.  
open i-o file1.
```

Note, however, that the workaround would not be necessary if you start the transaction after opening the file, e.g:

```
open i-o file1.  
start transaction.
```


Chapter 7

c-tree Utilities

c-tree comes with a series of utilities that manage the c-tree Server ISAM and SQL features and the data files. There are [Command-line utilities](#) as well as [Graphical utilities](#) and [Web utilities](#).

Command-line utilities

The following command-line utilities are available:

- `ctadm`
- `ctcmdset`
- `ctdump`
- `ctfdmp`
- `ctfileid`
- `ctldmp`
- `ctpass`
- `ctquiet`
- `ctrdmp`
- `ctsqlcdb`
- `ctsqlutl`
- `ctstat`
- `ctstop`
- `cttctx`
- `cttrnmod`
- `ctunf1`
- `ctutil`
- `dbdump`
- `dbload`
- `dbschema`
- `isql`
- `sa_admin`

All the executable files are available in the “tools” folder except `ctutil` that can be found in the “drivers/ctree.cobol/iscobol” folder of c-tree as well as in the “bin” folder of the isCOBOL SDK.

ctadm

`ctadm` manages the c-tree Server. Its command-line interface offers a series of wizard procedures to monitor the server status, connected users, stop the c-tree Server and more.

When you launch `ctadm` you are prompted for login information and for the name of the c-tree Server you wish to manage.

The main menu lists the following options:

1. User Operations
2. Group Definitions
3. File Security
4. Monitor Clients
5. Server Information (IOPERFORMANCE)
6. Server Configuration (SystemConfiguration)
7. Stop Server
8. Quiesce Server
9. Monitor Server Activity
10. Change Server Settings

In this document only the most common operations are described. Refer to the c-tree Server Administrator's Guide for additional information.

Server administration

- To show server statics, type 6 and press ENTER.
- To show server-side i/o performance, type 5 and press ENTER.
- To list connected clients, type 4, press ENTER, type 1 and press ENTER.
In this list you find an entry for each client connection. A new client connection is created
 - o at the first OPEN performed by the main program of a stand-alone runtime session,
 - o at the first OPEN performed by the main program of a client session in thin client environment, and
 - o at the first OPEN performed by a program called through CALL RUN statement.
- To kill a client, type 4, press ENTER, type 2, press ENTER, provide client ID and press ENTER.
- To monitor the server activity, type 9 and press ENTER. The following options will appear:

Monitor Server Activity:

1. List all files open by the c-tree Server
2. List all files open by a particular connection
3. List connections that have a particular file open
4. List locks held on a particular file

- o Type the number of the desired option and press ENTER. You will be prompted for additional information depending on the chosen option.
- To stop the c-tree Server, type 7 and press ENTER. You will be prompted for a confirmation and for the number of seconds of delay before the shutdown.

ctcmdset

The ctcmdset utility creates an encrypted password file from a text file by encoding a *.cfg file into a *.set file. The file to be processed should have an extension of .cfg.

Usage:

```
ctcmdset [filename].cfg
```

The plain text form of the file should be:

```
; User Id
USERID ADMIN
; User Password
PASSWD <pass>
```

Note - Each line must end with a line-feed for the password to be encrypted correctly.

Refer to c-tree Server Administrator's Guide for additional information on this utility.

ctdump

The ctdump utility produces 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.

Usage:

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

Refer to c-tree Server Administrator's Guide for additional information on this utility.

ctfdmp

The forward dump utility, ctfdmp, is used to restore data to a given time following a [ctrdump](#) restore.

Usage:

```
ctfdmp
```

ctfileid

When a file open is attempted, c-tree 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. In either case, the match means that a physical file open is not required. Instead the open count for the file is incremented. Checking the unique file ID permits different applications and/or client nodes to refer to the same file with different names. For example, consider the situation where independent clients operate from different drive or path mappings.

If two different files have the same file ID, (a 12-byte value comprised of a Server ID, a time stamp, and a transaction log sequence number), problems will arise as the second file would not actually be opened. The ID is constructed such that no two files can have the same ID unless someone copies one file on top of another.

Copying a file to a new name is typically the only way the file ID's can match. The ctfileid utility is available to update the file ID number should this become absolutely necessary. This utility should only be run on a file when the server is stopped.

The ctfileid utility provides a convenient and safe way to update the fileid parameter of the file header.

Usage:

```
ctfileid file [-i] [-o] [-q] [-n size] [-s server] [-u uid] [-p pwd]
```

Where

- *file* is the c-tree dat file to be updated
- *-i* also updates indices related to the data file
- *-o* force open of corrupted file
- *-q* avoid printing output (quiet mode)
- *size* is the node size
- *server* is the c-tree server name, e.g. 'FAIRCOMS'
- *uid* is the user id, e.g. 'ADMIN'
- *pwd* is the user password, e.g. 'ADMIN'

ctldmp

The ctldmp utility performs a partial dump of the transaction logs to assist the developer in problem resolution.

Usage:

```
ctldmp [ DATE mm/dd/yyyy ] [ TIME hh:mm:ss ] [ LOG number ]
```

ctpass

The ctpass utility allows users to change their password.

It is an interactive utility.

1. Enter your current User ID.
2. Enter the current password for your User ID, if you have one. Maximum 63 characters are allowed.
3. Continue by entering the current name of the c-tree server (i.e. "FAIRCOMS").
4. Now change your password by entering the new password.

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.

Note - all users can change their own passwords. In addition, users who are members of the ADMIN group can change the password of all accounts that are not members of the ADMIN group. Only the super ADMIN account (named ADMIN) can change a password for an account that is a member of the ADMIN group.

ctquiet

The ctquiet utility allows an administrator to quiet the server from a script. An interactive option is available in the [ctadmn](#) utility.

Usage:

```
ctquiet [-s server] [-f] [-u] -p password
```

Where

- *server* is the c-tree server name, e.g. 'FAIRCOMS'
- *-f* Full consistency - files are flushed to disk from cache.
- *-u* Unquiet server
- *password* is the admin password, e.g. 'ADMIN'

Note - When you quiesce the server, as long as the connection that quiesced the server remains connected, all other connections are blocked. Only if that connection goes away c-tree allows the ADMIN user to logon again and undo the quiesce.

Examples

quiet the server

```
ctquiet -p ADMIN
```

wake the server

```
ctquiet -u -p ADMIN
```

ctrdump

The ctrdump utility is used to restore dumps created with [ctdump](#).

Usage

```
ctrdump [ dumpscript ]
```

A successful ctrdump completion always writes the following message to CTSTATUS.FCS:

```
DR: Successful Dump Restore Termination
```

A failed ctrdump writes the following message to CTSTATUS.FCS when ctrdump terminates normally:

```
DR: Dump Restore Error Termination...: <cterr>
```

where <cterr> is the error code.

ctsqlcdb

The ctsqlcdb utility creates a new, adds an existing, or drops a database from the c-tree SQL Server.

Usage:

```
ctsqlcdb { -add      } database_name [servername]
          { -create   }
          { -drop     }
```

Where:

- *-add* adds a reference to an existing database (be sure that the database dbs folder has been copied to the server's *data* directory before using this option)
- *-create* creates a new database
- *-drop* removes a reference to an existing database
- *database_name* is the name of the database you want to add, drop or create
- *servername* is the optional c-tree SQL Server name

ctsqlutl

The ctsqlutl utility allows to change the name of table columns.

Usage:

```
ctsqlutl [options] -rencol table_name column newcolumn
```

Where:

- *table_name* is the name of the table with the column you want to rename
- *column* is the name of the column you want to rename
- *newcolumn* is the new name for the column
- *options* can be one or more of the following:
 - o *-o owner_name*: owner of table
 - o *-d database_name*: database name (default: ctreeSQL)
 - o *-s server_name*: c-tree SQL Server name (default: FAIRCOMS)
 - o *-u userid*: userid for logging onto the c-tree SQL Server
 - o *-a password*: password for authentication

ctstat

The ctstat utility is used to display statistics collected by the c-tree Server. ctstat, provides valuable real time monitoring of critical server operations in the areas of cache usage, disk I/O, open files, established client connections, file locks, and transactions.

Usage:

```
usage: ctstat  report_type [-s svn] [-u uid] [-p upw] [-i int [cnt]] [-h frq]
        [-d] [-m] [-t]

reports:
  -vas          Admin-System Report
  -vts          Tivoli-System Report
  -vaf file...  Admin-File Report
  -vtf file...  Tivoli-File Report
  -vau user...  Admin-User Report by User Name
  -vah handle... Admin-User Report by User Handle
  -func         Function Timing Report
  -funcfile     Function Timing By File Report
  -userinfo     User Report with stats from USERINFO() function
  -ISAM         ISAM Activity Report
  -sql          SQL Activity Report
  -text         System Activity Report
  -file [csv]   File Activity Report
  -iotime on|off Turn disk I/O call timing on or off
  -wrktime on|off Turn function call timing on or off

options:
  -s svn        c-tree Server name
  -u uid        User name
  -p upw        User password
  -i int [cnt]  Pause int seconds for optional cnt times
  -h frq        Print a description header every frq outputs
  -d           Show cache stats as delta
  -m           Show memory file stats when using -vaf report
  -t           Output timestamp with header
```

For additional information on the output of ctstat, consult the c-tree Server Administrator's Guide.

ctstop

ctstop stops the c-tree Server with a single command, without user interaction like ctadmn. It's useful if you plan to stop the c-tree Server from a script file or in the Linux crontab.

Usage:

```
ctstop [[-auto] adminuser adminpass servername]
```

Where:

- *-auto*, if specified, doesn't ask the user what to do with connected clients, if any.
- *adminuser* is the name of the administrator user, usually "admin" (case insensitive).
- *adminpass* is the password for the administrator user, usually "ADMIN" (case sensitive).
- *servername* is the name of the c-tree Server to stop. By default the c-tree Server is named "FAIRCOMS".

Note - If *-auto* is omitted and there are connected clients or if you don't pass *adminuser*, *adminpass* and *servername* on the command line, the utility will prompt for information and therefore user interaction is required.

cttctx

cttctx is a multi-threaded c-tree client test for comprehensive testing of c-tree server operations. This utility was designed specifically for profiling c-tree performance. Use this utility to simulate high load conditions against the c-tree server for verifying application performance.

Usage

```
cttctx <uid> <upw> <svn> [create [-t<trnmod>] | <nThrds> [-c<concurrency>] [-d<dist>] [-e] [-h] [-j] [-o<op>] [-p] [-r<msec>] [-n<niter>] ]
```

Arguments

- <uid> User name
- <upw> User password
- <svn> c-tree Server name
- create Create new test data/index files
- -t<trnmod> Transaction mode to use when creating the files
- trnmod is one of the following
 - o t ctTRNLOG
 - o p ctPREIMG
 - o n no tran
- <nThrds> Number of c-tree threads to spawn
- -c<concurrency> Change thread concurrency to <concurrency>
- -e Use the embedded c-treeSQL interface (otherwise use the ISAM interface)
- -h Hard exit, abrupt termination without thread cleanup
- -j Add to vlength files
- -o<op> Operation to apply
- <op> is one of the following:
 - o a Add
 - o r Read
 - o d Delete
- -p track performance stats such as latency of operations
- -r<msec> Number of milliseconds between performi

Refer to c-tree Server Administrator's Guide for additional information on this utility.

cttrnmod

The cttrnmod changes the transaction mode.

Usage

```
cttrnmod (set <tranmode>|get) (-d <database>|-f <filelist>)
        [-u <userid>] [-p <password>] [-s <servername>] [-n <sect>]

options:
  set <tranmode>    Set the transaction mode to one of the following:
                    T   Full Transaction Control
                    P   Partial Transaction Control (No Recoverability)
                    N   No Transaction Control (No Recoverability)
  repl=on           Enable replication (requires full transaction control)
  repl=off          Disable replication

                    The following extended header attributes may also be set:
                    {+,-}R {Enable,Disable} Restorable deletes
                    {+,-}C {Enable,Disable} Transaction controlled deletes

  get               Display the current transaction mode.
  -d <database>     Operate on all files in the c-tree database <database>.
  -f <filename>     Operate on all files listed in the file <filelist>.
  -u <userid>       Specify c-tree user ID.
  -p <password>     Specify c-tree user password.
  -s <servername>  Specify c-tree Server name to connect to. Default: FAIRCOMS
  -n <sect>         Specify node sector size. Default: 64 (PAGE_SIZE=8192)
```

ctunf1

ctunf1 converts c-tree data files changing their numeric alignment. This utility is useful if you need to move your data files to an operating system that has a different numeric alignment.

Example:

c-tree files have been created on Microsoft Windows that has little endian alignment and must be moved to an AIX machine, that has big endian alignment. Before moving the files, ctunf1 must be run on the Windows machine in order to change their numeric alignment.

Usage:

```
ctunf1 <file name> <new alignment> <new flavor> [<confirm>] [<binflag>] [<files>]
```

where new alignment = 1 (byte), 2 (word), 4 (double word), or 8 (quad word).
new flavor = L (least significant byte first; e.g., 8086 family), or
 = M (most significant byte first; e.g., 68000 family).
<confirm> = Y (convert file without prompting to continue)
<binflag> = optional c-treeDB binary field attribute
 = B1 (binary field data does not contain length count)
 = B2 (binary field data contains length count)
files = optional number of logical files to accommodate
 (default = 32)

IT IS ASSUMED THAT NO DATA RECORDS WILL CHANGE IN LENGTH. IF A
CHANGE IN ALIGNMENT CAUSES A RECORD LENGTH CHANGE, CTUNF1 WILL
AUTOMATICALLY TERMINATE.

EXAMPLE:

```
ctunf1 mydatafile.dat 8 M Y B2 200
```

ctutil

ctutil allows you to examine files, extract data records, change the maximum record size, and rebuild corrupted indexes. The functions are designed to allow you to specify all possible task parameters up front, so that the utility can run unattended or with a minimum of user interactions.

Usage

```
ctutil [-c config_file] [-s] [-l]
Information options
  -info      file [-x]
  -param     file
  -profile   file
Maintenance options
  -make      file iss_file
  -copy      source_file dest_file
  -clone     source_file dest_file
  -filecopy  source_file dest_file [-overwrite]
  -rename    source_file dest_file [-overwrite]
  -remove    file
  -upgrade   source_file [dest_file]
Consistency options
  -check     file [-x] [-k=index]
  -rebuild   file [-purgedups] [-delidx] [-repairdata]
  -compact   file [-purgedups] [-delidx] [-repairdata]
  -fileid    file
Definition options
  -setpath   file
  -tron      file T|P|F|W
  -segment   file max_file_size max_segments
  -conv      file convention_ID
  -compress  file
  -uncompress file
String image options
  -getimg    file
  -makeimg   file image_string
  -checkimg  file image_string
  -rblimg    file image_string
  -addimg    file image_string
Import/export options
  -load      file seq_file [-b|t|p|r2] [-v[2|4|8][n|x]] [-n] [-r|s] [-rs=recsiz]
  -unload    file seq_file [-b|t|p] [-v[2|4|8][n|x]] [-k=index]
SQL options
  -sqlinfo   file [iss_file [convention_ID]]
  -sqllink   file database_name
              [-symb=table_name] [-prefix=table_prefix]
              [-owner=user_name] [-public[=ro]]
  -sqlunlink file database_name
              [-symb=table_name] [-prefix=table_prefix]
              [-owner=user_name]
  -sqlize    file iss_file database_name
              [-symb=table_name] [-prefix=table_prefix]
              [-owner=user_name] [-public[=ro]]
              [-conv=convention_ID] [-rule=rules_file]
  -sqlrefresh file iss_file database_name
              [-symb=table_name] [-prefix=table_prefix]
              [-owner=user_name] [-conv=convention_ID]
              [-rule=rules_file]
  -sqlcheck  file iss_file [-conv=convention_ID] [-show=show_type]
  -ddf2xdd   DDF_directory [-rule=rules_file]
Miscellaneous options
  -cryptconf config_file output_file
  -test      [config|connect|filerules]
  -run       command_list_file
```

General rules

- *config_file* specifies the configuration file to be used instead of the default. See [Configuring the client through CTREE_CONF](#) for details about the configuration file.
- When the *-s* option is specified, superuser mode is activated and files marked as corrupted can be opened.
- When the *-l* option is specified, the utility dumps the configuration in XML format before the command output.
- option names have been changed. Old names (that begin with *_ct* instead of *-*) are still supported for backward compatibility.

Commands

-check

Checks for the file integrity. If the file is corrupted, you can try to repair it using *-rebuild*.

Usage

```
ctutil -check file [-x] [-k=index]
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.
- If the *-x* option is specified, ctutil performs extended tests matching records with keys.
- If the *-k=index* option is specified, the check is performed only on the index specified. For very large files with many indexes, you may want to consider using this option to validate only a single index.

-checking

Checks file integrity using the physical file information.

Usage

```
ctutil -checking file image_string
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.
- *image_string* has the following format:

```
MaxRecSize, MinRecSize, NumKeys, [ NumSegs, Dups [ SegSize, SegOffset ] ... ] ...
```

Where:

- o *MaxRecSize* is a five-digit number representing the maximum record length.
- o *MinRecSize* is a five-digit number representing the minimum record length.
- o *NumKeys* is a three-digit number representing the number of keys in the file.
- o *NumSegs* is a two-digit number representing the number of segments in a key.
- o *Dups* is a one-digit number representing the duplicate flag of a key. 0 means that duplicates are not allowed, 1 means that duplicates are allowed.

- o *SegSize* is a three-digit number representing the size of a segment in a key.
- o *SegOffset* is a five-digit number representing the offset of a segment in a key.

SegSize and *SegOffset* are repeated for each segment of the key
NumSegs, *Dups* and segments description are repeated for each key

Note - spaces in are shown to improve readability, they are not part of the format. Fields in the first pair of brackets are repeated for each key, fields in the second pair of brackets are repeated for each segment of the key.

Example - the following string applies to a file with a fixed record length of 108 bytes, a primary key composed of one segment of three bytes and an alternate key with duplicates composed of two segments, the first of two bytes in size and the second of three bytes in size:

```
00108,00108,002,01,0,003,00000,02,1,002,00003,003,00005
```

-clone

Creates an empty copy of an existing file.

Usage

```
ctutil -clone source_file dest_file
```

- *source_file* is the name of the c-tree file to copy.
- *dest_file* is the name of the new empty file that will be created.

-compact

Compacts data file by removing deleted records.

Usage

```
ctutil -compact file [-purgedups] [-delidx] [-repairdata]
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.
- if the *-purgedups* options is specified, all records with a duplicated primary key are removed from the file.
- The *-delidx* option forces the deletion of the index file before rebuilding.
- if the *-repairdata* option is specified, ctutil attempts to recover a file, as done prior to V11.5. To avoid potential data loss, FairCom strongly recommends only using our latest c-treeACE V11.5 compact and rebuild logic that default to stopping if a corrupt data record header is encountered, and returns error DCPT_ERR(1107) rather than attempting to repair the data link in the record header. This allows an administrator to know with greater certainty that serious damage has occurred and they may prefer to recover the file from a backup.

-compress

Compress a c-tree file. Compression level and strategy are read from the [<datacompress>](#) configuration entry. See [Configuring the client through CTREE_CONF](#) for details.

Usage

```
ctutil -compress file
```

- o *file* is the name of the c-tree file.

-conv

Insert information about the sign convention into the c-tree file. This information is useful for the c-tree SQL Server.

Usage

```
ctutil -conv file sign_convention
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.
- *sign_convention* can be one of the following:
 - o A (ANSI convention, default)
 - o B (MBP convention)
 - o D (Data General convention)
 - o I (IBM convention)
 - o M (Micro Focus convention)
 - o N (NCR convention)
 - o R (Realia convention)
 - o V (VAX convention)

-copy

Copies a file record-by-record creating a new file based on the configuration settings.

Usage

```
ctutil -copy source_file dest_file
```

- *source_file* and *dest_file* are the name of the involved c-tree files. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.

-copy vs -filecopy

The **-copy** option creates a copy of a file by building it record-by-record, which creates a new file based on the configuration settings. The **-filecopy** option makes an exact copy of the source file, which can be faster than the **-copy** option.

The difference between these options is that the **-copy** option creates the destination file based on the configuration setting while the **-filecopy** option creates an exact copy of the source file.

Both approaches have limitations and advantages:

- **-filecopy** is faster than **-copy** and provides an exact copy of the file.

- `-copy` may be helpful when you need to change file characteristics.

-cryptconf

Encrypts the `ctree.conf` configuration file.

Usage

```
ctutil -cryptconf config_file output_file
```

- `config_file` is the name of the configuration file file.
- `output_file` is the encrypted file that will be created.

-filecopy

Perform a physical file copy.

Usage

```
ctutil -filecopy source_file dest_file [-overwrite]
```

- `source_file` is the name of the c-tree file to copy.
- `dest_file` is the name of the new file.
- `-overwrite` deletes existing `dest_file` before copying.

The command is not dependent on the configuration file.

-filecopy vs -copy

The `-copy` option creates a copy of a file by building it record-by-record, which creates a new file based on the configuration settings. The `-filecopy` option makes an exact copy of the source file, which can be faster than the `-copy` option.

The difference between these options is that the `-copy` option creates the destination file based on the configuration setting while the `-filecopy` option creates an exact copy of the source file.

Both approaches have limitations and advantages:

- `-filecopy` is faster than `-copy` and provides an exact copy of the file.
- `-copy` may be helpful when you need to change file characteristics.

-fileid

Reset the unique file ID of a file that has been copied at the system level.

Usage

```
ctutil -fileid file
```

- `file` is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.

Note - Copying files is against FairCom's recommended best practices. However, because copying files is fairly common in many industries, FairCom has the following provisions:

- Use the `-filecopy` option or the `-copy` option to copy files.
- Restamp file ID after copying using the operating system - The `ctutil -fileid` option is for restamping the file ID after a file is copied using the operating system facilities.

-getimg

Displays the physical file information.

Usage

```
ctutil -getimg file
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.
- Physical information has the following format:

```
MaxRecSize, MinRecSize, NumKeys, [ NumSegs, Dups [ SegSize, SegOffset ] ... ] ...
```

Where:

- o *MaxRecSize* is a five-digit number representing the maximum record length.
- o *MinRecSize* is a five-digit number representing the minimum record length.
- o *NumKeys* is a three-digit number representing the number of keys in the file.
- o *NumSegs* is a two-digit number representing the number of segments in a key.
- o *Dups* is a one-digit number representing the duplicate flag of a key. 0 means that duplicates are not allowed, 1 means that duplicates are allowed.
- o *SegSize* is a three-digit number representing the size of a segment in a key.
- o *SegOffset* is a five-digit number representing the offset of a segment in a key.

SegSize and *SegOffset* are repeated for each segment of the key
NumSegs, *Dups* and segments description are repeated for each key

Note - spaces in are shown to improve readability, they are not part of the format. Fields in the first pair of brackets are repeated for each key, fields in the second pair of brackets are repeated for each segment of the key.

Example - the following string applies to a file with a fixed record length of 108 bytes, a primary key composed of one segment of three bytes and an alternate key with duplicates composed of two segments, the first of two bytes in size and the second of three bytes in size:

```
00108,00108,002,01,0,003,00000,02,1,002,00003,003,00005
```

-info

Displays file information.

Usage

```
ctutil -info file [-x]
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.
- if the -x option is specified, ctutil prints extended information.

-load

Imports data from raw sequential file to c-tree file.

Usage

```
ctutil -load file_name seq_file [-b|t|p|r2] [-v[2|4|8] [n|x]] [-n] [-r|-s] [-rs=recsiz]
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.
- *seqfile* is the name of the input file. Relative paths are resolved according to the current working directory.
- -b - Indicates a binary sequential format is used.
- -t - Indicates a line sequential format (records separated by new-line character).
- -p - Indicates an ASCII file created by the BTRV BUTIL -save command.
- -r2 - Indicates an ASCII file created by the RM RECOVER2.
- -v - Indicates variable-length format (record data is preceded by record length) and is optionally followed by a 2 or 4 or 8 that indicates the size of the record length field that precedes the record data and by a n or x to indicate if the record length is represented in native (n) or big-endian (x) format.
- -n - Empty the c-tree file before loading records.
- -r - Replace any existing record that returns 'duplicate key' error.
- -s - Skip any existing record that returns 'duplicate key' error.
- -rs=recsiz - Record size of the sequential file (required only if the record size of the destination file differs from that of the sequential file).

The command assumes that the record size of the sequential file is the same as the c-tree file. If they differ, the -rs option is used to specify the record size of the sequential file.

-loadtext

Imports data from line sequential file to c-tree file

Note - This command has been replaced by the -t option of the [-load](#) command.

-make

Creates a file from scratch with the characteristics described in a given iss dictionary.

Usage

```
ctutil -make file iss_file
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory
- *iss_file* is the name of a iss dictionary. These dictionaries are generated by the isCOBOL Compiler when the -efc option is used.

-makeimg

Creates a new file using the physical file information.

Usage

```
ctutil -makeimg file image_string
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.
- *image_string* has the following format:

```
MaxRecSize, MinRecSize, NumKeys, [ NumSegs, Dups [ SegSize, SegOffset ] ... ] ...
```

Where:

- o *MaxRecSize* is a five-digit number representing the maximum record length.
- o *MinRecSize* is a five-digit number representing the minimum record length.
- o *NumKeys* is a three-digit number representing the number of keys in the file.
- o *NumSegs* is a two-digit number representing the number of segments in a key.
- o *Dups* is a one-digit number representing the duplicate flag of a key. 0 means that duplicates are not allowed, 1 means that duplicates are allowed.
- o *SegSize* is a three-digit number representing the size of a segment in a key.
- o *SegOffset* is a five-digit number representing the offset of a segment in a key.

SegSize and *SegOffset* are repeated for each segment of the key

NumSegs, *Dups* and segments description are repeated for each key

Note - spaces in are shown to improve readability, they are not part of the format. Fields in the first pair of brackets are repeated for each key, fields in the second pair of brackets are repeated for each segment of the key.

Example - the following string applies to a file with a fixed record length of 108 bytes, a primary key composed of one segment of three bytes and an alternate key with duplicates composed of two segments, the first of two bytes in size and the second of three bytes in size:

```
00108,00108,002,01,0,003,00000,02,1,002,00003,003,00005
```

-param

Displays file definitions in 'parameter file' format.

Usage

```
ctutil -param file
```

file is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.

-profile

Shows internal information of a c-tree file.

Usage

```
ctutil -profile file
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.

-rblimg

Rebuilds indexes using the physical file information.

Usage

```
ctutil -rblimg file image_string
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.
- *image_string* has the following format:

```
MaxRecSize, MinRecSize, NumKeys, [ NumSegs, Dups [ SegSize, SegOffset ] ... ] ...
```

Where:

- o *MaxRecSize* is a five-digit number representing the maximum record length.
- o *MinRecSize* is a five-digit number representing the minimum record length.
- o *NumKeys* is a three-digit number representing the number of keys in the file.
- o *NumSegs* is a two-digit number representing the number of segments in a key.
- o *Dups* is a one-digit number representing the duplicate flag of a key. 0 means that duplicates are not allowed, 1 means that duplicates are allowed.
- o *SegSize* is a three-digit number representing the size of a segment in a key.
- o *SegOffset* is a five-digit number representing the offset of a segment in a key.

SegSize and *SegOffset* are repeated for each segment of the key
NumSegs, *Dups* and segments description are repeated for each key

Note - spaces in are shown to improve readability, they are not part of the format. Fields in the first pair of brackets are repeated for each key, fields in the second pair of brackets are repeated for each segment of the key.

Example - the following string applies to a file with a fixed record length of 108 bytes, a primary key composed of one segment of three bytes and an alternate key with duplicates composed of two segments, the first of two bytes in size and the second of three bytes in size:

```
00108,00108,002,01,0,003,00000,02,1,002,00003,003,00005
```

-rebuild

Rebuilds indexes using internal file definitions.

Usage

```
ctutil -rebuild file [-purgedups] [-delidx] [-repairdata]
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.
- if the *purgedups* option is specified, all records with a duplicated primary key are removed from the file.
- if the *delidx* option is specified, the index file is deleted before the rebuild process starts.
- if the *-repairdata* option is specified, ctutil attempts to recover a file, as done prior to V11.5. To avoid potential data loss, FairCom strongly recommends only using our latest c-treeACE V11.5 compact and rebuild logic that default to stopping if a corrupt data record header is encountered, and returns error DCPT_ERR(1107) rather than attempting to repair the data link in the record header. This allows an administrator to know with greater certainty that serious damage has occurred and they may prefer to recover the file from a backup.

-remove

Deletes a c-tree file.

Usage

```
ctutil -remove file
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.

-rename

Renames a c-tree file.

Usage

```
ctutil -rename source_file dest_file
```

- *source_file* and *dest_file* are the name of the involved c-tree files. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.

-segment

Allows file segmentation.

Usage

```
ctutil -segment file max_file_size max_segments
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.
- *max_file_size* is the maximum file size for a single file segment
- *max_segments* is the maximum number of file segment. Keep this parameter as low as possible to optimize performance.

-setpath

Resets file path information.

The c-tree files contain information about the file including the path location of the file itself. After copying a file to a location other than the creation directory, it is necessary to reset the file path information according to the new file location.

Usage

```
ctutil -setpath file
```

file is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.

-sign

This option is synonymous with the [-conv](#) option and it has the same usage.

-sqlcheck

Verifies that data in a file is compatible with SQL definitions. The command scans all records of the given file using the primary key index and stops at the first conversion error encountered showing an error message, the record number, schema and field name of the incorrect data.

Usage

```
ctutil -sqlcheck file iss_file [-conv=convention_ID]
```

- *file* - File name without extension
- *iss_file* - Path to a data definition file in XML format
- *convention_ID* - Optional numeric storage convention ID:
 - o A - ACUCOBOL-GT (default)
 - o B - MBP COBOL
 - o D - Data General
 - o I - IBM
 - o M - Micro Focus
 - o N - NCR COBOL
 - o R - Realia COBOL
 - o V - VAX COBOL

-sqlinfo

Includes extended file information from iss file into the physical file.

iss files are generated by the isCOBOL Compiler when -efc option is used.

Usage

```
ctutil -sqlinfo file iss_file [sign_convention]
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.
- *iss_file* is the dictionary file. Relative paths are resolved according to the ctutil working directory.
- *sign_convention* identifies the numeric data convention.
 - o use A for programs compiled with -dca option or without data compatibility options.
 - o use B for programs compiled with -dcb option.
 - o use D for programs compiled with -dcd option.
 - o use I for programs compiled with -dci option.
 - o use M for programs compiled with -dcm option.
 - o use N for programs compiled with -dcn option.
 - o use R for programs compiled with -dcr option.
 - o use V for programs compiled with -dcv option.

If omitted, A is assumed

This operation is necessary before linking a file into c-tree SQL Server with **-sqllink** option.

-sqllink

Links a physical file into c-tree SQL Server database. It requires the administrator password, the database name and optionally a logical name to be used for the table. If the last parameter is not provided, the table on the database will be called the same way of the disk file.

After the link operation, the physical file can be interfaced as standard ISAM file by isCOBOL Framework and as database table from SQL clients.

Usage

```
ctutil -sqllink file database_name [-symb=symbolic_name] [-owner=owner_name] [-  
prefix=table_prefix] [-public[=ro]]
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.
- *database_name* is the name of the destination SQL Server database.
- the *-symb* option allows to specify the name for the resulting table. By default, the table has the same name as the disc file.
- the *-owner* option allows to specify a specific table ownership.
- the *-prefix* option allows to specify a prefix that will be put before the table name in the database. For example, if you link a file named "file1" with -prefix=myprefix, you obtain a table named "myprefixfile1" in the database.

- the *-public* option grants public access permissions on the resulting table. Use *-public=ro* to indicate read-only permissions to all SQL users. Admin users are always able to modify a c-tree file.

Administrator password

The administrator password is included in the configuration file.* This file can be:

- in the working directory in a file named *ctree.conf*
- in the directory specified in the environment variable CTREE_CONF
- passed to ctutil using the -c option in the command line

This configuration file should contain this statement:

```
<config>
  <instance server="FAIRCOMS" user="admin" password="ADMIN">
  </instance>
</config>
```

Other statements can be added to the configuration file as needed.

*To be compatible with older c-tree versions that allowed you to pass the administrator password in the command line, you can add the new configuration variable COMPATIBILITY SQLIMPORT_ADMIN_PASSWORD to the c-tree Server configuration file (ctsrvr.cfg). However, adding the admin and password to the configuration file as shown above is considered the best practice.

Note - After the archive has been registered as a SQL table, it can be removed by a DROP TABLE statement on the database. If you want to keep the archive on disc for ISAM access after a DROP TABLE, add [SQL_OPTION DROP_TABLE_DICTIONARY_ONLY](#) to the c-tree Server's configuration file (ctsrvr.cfg).

-sqlunlink

Removes the reference to the physical file from the c-tree SQL Server database. It requires the administrator password, the database name and the logical name if specified during the link. After the unlink operation, the physical file will not be available anymore as database table for SQL clients, but it can be interfaced again by the isCOBOL Framework.

Usage

```
ctutil -sqlunlink file database_name [-symb=symbolic_name] [-owner=owner_name]
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.
- *database_name* is the name of the destination SQL database.
- the *-symb* option allows to specify the name for the resulting table. By default, the table has the same name as the disc file.
- the *-owner* option allows to specify a specific table ownership.

Administrator password

The administrator password is included in the configuration file.* This file can be:

- in the working directory in a file named *ctree.conf*
- in the directory specified in the environment variable CTREE_CONF
- passed to ctutil using the -c option in the command line

This configuration file should contain this statement:

```
<config>
  <instance server="FAIRCOMS" user="admin" password="ADMIN">
  </instance>
</config>
```

Other statements can be added to the configuration file as needed.

*To be compatible with older c-tree versions that allowed you to pass the administrator password in the command line, you can add the new configuration variable COMPATIBILITY SQLIMPORT_ADMIN_PASSWORD to the c-tree Server configuration file (ctsrvr.cfg). However, adding the admin and password to the configuration file as shown above is considered the best practice.

-sqlize

Registers a file into the c-tree SQL Server in one single step. Using -sqlize is like using -sqlinfo followed by -sqlunlink and -sqllink. It requires the administrator password, the database name and an optional logical name to be used for the table as well as the iss file and the numeric data convention.

Usage

```
ctutil -sqlize file iss_file database_name [-syb=symbolic_name] [-owner=owner_name] [-
prefix=table_prefix] [-public[=ro]] [-conv=sign_convention] [-rule=rules_file]
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.
- *iss_file* is the dictionary file. Relative paths are resolved according to the ctutil working directory.
- *database_name* is the name of the destination SQL Server database.
- the *-syb* option allows to specify the name for the resulting table. By default, the table has the same name as the disc file.
- the *-owner* option allows to specify a specific table ownership.
- the *-prefix* option allows to specify a prefix that will be put before the table name in the database. For example, if you link a file named "file1" with -prefix=myprefix, you obtain a table named "myprefixfile1" in the database.
- *sign_convention* identifies the numeric data convention.
 - o use A for programs compiled with -dca option or without data compatibility options.
 - o use B for programs compiled with -dcb option.
 - o use D for programs compiled with -dcd option.
 - o use I for programs compiled with -dci option.
 - o use M for programs compiled with -dcm option.
 - o use N for programs compiled with -dcn option.
 - o use R for programs compiled with -dcr option.
 - o use V for programs compiled with -dcv option.

If omitted, A is assumed
- the *-public* option grants public access permissions on the resulting table. Use *-public=ro* to indicate read-only permissions to all SQL users. Admin users are always able to modify a c-tree file.

- *rules_file* identifies a file with external rules. See [Defining External Rules](#) in Faircom's documentation for more information.

Administrator password

The administrator password is included in the configuration file.* This file can be:

- in the working directory in a file named *ctree.conf*
- in the directory specified in the environment variable CTREE_CONF
- passed to ctutil using the -c option in the command line

This configuration file should contain this statement:

```
<config>
  <instance server="FAIRCOMS" user="admin" password="ADMIN">
  </instance>
</config>
```

Other statements can be added to the configuration file as needed.

*To be compatible with older c-tree versions that allowed you to pass the administrator password in the command line, you can add the new configuration variable COMPATIBILITY SQLIMPORT_ADMIN_PASSWORD to the c-tree Server configuration file (ctsrvr.cfg). However, adding the admin and password to the configuration file as shown above is considered the best practice.

Note - After the archive has been registered as a SQL table, it can be removed by a DROP TABLE statement on the database. If you want to keep the archive on disc for ISAM access after a DROP TABLE, add [SQL_OPTION DROP_TABLE_DICTIONARY_ONLY](#) to the c-tree Server's configuration file (ctsrvr.cfg).

-sqlcheck

Verifies that the data is correct in sqlized tables.

Usage

```
ctutil -sqlcheck file iss_file [-conv=sign_convention] [-show=show_type]
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.
- *iss_file* is the dictionary file. Relative paths are resolved according to the ctutil working directory.
- *sign_convention* identifies the numeric data convention.
 - o use A for programs compiled with -dca option or without data compatibility options.
 - o use B for programs compiled with -dcb option.
 - o use D for programs compiled with -dcd option.
 - o use I for programs compiled with -dci option.
 - o use M for programs compiled with -dcm option.
 - o use N for programs compiled with -dcn option.
 - o use R for programs compiled with -dcr option.
 - o use V for programs compiled with -dcv option.

If omitted, A is assumed

- *show_type* is the level of error to show:

Type	Meaning
all	show all errors in the table
first	show the first error in the table
firstrec	show all errors in the first problematic record in the table

-sqlrefresh

Preserves existing information about the table (synonyms, grants, etc.) when relinking into SQL.

Usage

```
ctutil -sqlrefresh file iss_file database_name [-symb=symbolic_name] [-owner=owner_name] [-prefix=table_prefix] [-conv=sign_convention] [-rule=rules_file]
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.
- *iss_file* is the dictionary file. Relative paths are resolved according to the ctutil working directory.
- *database_name* is the name of the destination SQL Server database.
- the *-symb* option allows to specify the name for the resulting table. By default, the table has the same name as the disc file.
- the *-owner* option allows to specify a specific table ownership.
- the *-prefix* option allows to specify a prefix that will be put before the table name in the database. For example, if you link a file named "file1" with *-prefix=myprefix*, you obtain a table named "myprefixfile1" in the database.
- *sign_convention* identifies the numeric data convention.
 - o use A for programs compiled with *-dca* option or without data compatibility options.
 - o use B for programs compiled with *-dcb* option.
 - o use D for programs compiled with *-dcd* option.
 - o use I for programs compiled with *-dci* option.
 - o use M for programs compiled with *-dcm* option.
 - o use N for programs compiled with *-dcn* option.
 - o use R for programs compiled with *-dcr* option.
 - o use V for programs compiled with *-dcv* option.

If omitted, A is assumed
- *rules_file* identifies a file with external rules. See [Defining External Rules](#) in Faircom's documentation for more information.

Administrator password

The administrator password is included in the configuration file.* This file can be:

- in the working directory in a file named *ctree.conf*

- in the directory specified in the environment variable CTREE_CONF
- passed to ctutil using the -c option in the command line

This configuration file should contain this statement:

```
<config>
  <instance server="FAIRCOMS" user="admin" password="ADMIN">
  </instance>
</config>
```

Other statements can be added to the configuration file as needed.

*To be compatible with older c-tree versions that allowed you to pass the administrator password in the command line, you can add the new configuration variable COMPATIBILITY SQLIMPORT_ADMIN_PASSWORD to the c-tree Server configuration file (ctsrvr.cfg). However, adding the admin and password to the configuration file as shown above is considered the best practice.

-test

Checks the configuration and connections to servers.

Usage

```
ctutil -test [config | connect]
```

- Running *ctutil -test* with no option, or with the *config* option, checks the configuration.
- Running *ctutil -test connect* checks that all servers defined in the configuration (with the <instance server> attribute) are reachable.

-tron

Defines the type of data safety.

Usage

```
ctutil -tron file option
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory
- *option* can be one of the following:
 - o T - Enable full transaction processing control (ctTRNLOG c-tree file mode). This mode is recommended for most applications and it is required to take full advantage of c-tree Server's advanced features. Avoid it when performance is more important than data safety.
 - o P - Enable logical transaction processing control without logging (ctPREIMG c-tree file mode). This is the default mode and is indicated where both transaction atomicity and high performance are required. By suppressing transaction logging it is possible to achieve high data throughput at the expense of data recoverability. This mode is recommended in environments protected by system crash or in case data integrity/recoverability is not an issue.
 - o W - Synchronous writes (ctWRITETHRU c-tree file mode). This mode is indicated where transaction processing is not suitable yet data safety is still critical.
 - o F - Asynchronous writes. This mode should only be used on files in which data integrity is not critical as data recovery is not guaranteed in case of a system crash.

-upgrade

This option upgrades the file to the current configured format. With this switch it is possible to take an existing data file and upgrade it to the latest format.

Usage

```
ctutil -upgrade source_file [dest_file]
```

- *source_file* - Source file name without extension.
- *dest_file* - Destination file name without extension.

This capability gives the customer a tool to upgrade an existing file to match the current settings in `ctree.conf` and/or the current c-treeRTG file format. This switch makes it possible to upgrade an existing data file to the latest format. For example, it would be used if a revision changed the file's physical layout (e.g., altering the header).

-uncompress

Uncompress a compressed c-tree file.

Usage

```
ctutil -uncompress file
```

- o *file* is the name of the c-tree file.

-unload

Exports data from c-tree file to raw sequential file.

Usage

```
ctutil -unload file_name seq_file [-b|t|p] [-v[2|4|8] [n|x]] [-k=index]
```

- *file* is the name of the c-tree file. The default file extension (that is ".dat" if not configured differently) must be omitted. Relative paths are resolved according to the c-tree server working directory.
- *seqfile* is the name of the output file. Relative paths are resolved according to the current working directory.
- **-b** - Indicates a binary sequential format is used.
- **-t** - Indicates a line sequential format (records separated by new-line character).
- **-p** - Indicates an ASCII file to be used by the BTRV BUTIL -load command.
- **-v** - Indicates variable-length format (record data is preceded by record length) and is optionally followed by a 2 or 4 or 8 that indicates the size of the record length field that precedes the record data and by a n or x to indicate if the record length is stored in native (n) or big-endian (x) format.
- **-k** - Reads and writes records in specified index order (-k=1 for the primary key) .

-unloadtext

Exports data from c-tree file to line sequential file.

Note - This command has been replaced by the -t option of the **-unload** command.

-ddf2xdd

Transforms Btrieve Data Dictionary Files (DDF) file into c-tree XDD files. The XDD format is equivalent to the isCOBOL's ISS format.

Usage

```
ctutil -ddf2xdd dirname
```

- *dirname* is the name of a directory that contains file.sav, field.sav, and index.sav files.

The utility will create one XDD file for each file contained in file.sav. These files will be saved in the same directory.

-run

Executes a script file allowing to run multiple ctutil commands.

Usage

```
ctutil -run command_list_file
```

- *command_list_file* is a text file that contains one ctutil command on each line followed by all its required parameters.

The value returned by the ctutil -run if an error occurs executing one of the listed commands is the line number of the command list file containing the failed command.

The return value is 0 if all operations listed in the command list file are completed successfully.

dbdump

The dbdump utility writes data from a c-tree SQL Server file to an output file.

Usage:

```
dbdump -f commands_file [-u user_name] [-a password] [-n] database_name
```

Where:

- *command_file* is the script file containing the dump commands. Consult <http://www.faircom.com/doc/isql/> (Data Unload Utility: dbdump -> The Commands File) for details on this kind of files. The following snippet shows a sample command_file:

```
DEFINE RECORD ord_rec AS
  ( ord_no, item_name, date, item_qty ) FIELD DELIMITER ' ' ;
FOR RECORD ord_rec dump into ord_dat
USING SELECT order_no, product, order_date, qty
FROM items;
```

- *database_name* is the name of the destination database
 - *options* can be one or more of the following:
 - o *-u user_name*: user name to connect to the database.
 - o *-a password*: password to connect to the database.
- n*: parse the commands file and display errors, if any, without doing the database load.

dbload

The dbload utility loads records from an input data file into tables of a c-tree SQL database.

Usage:

```
dbload -f commands_file [options] database_name
```

Where:

- *command_file* is the script file containing the load commands. Consult <http://www.faircom.com/doc/isql/> (Data Load Utility: dbload -> The Commands File) for details on this kind of files. The following snippet shows a sample command_file:

```
DEFINE RECORD ord_rec AS
  ( ord_no, item_name, date, item_qty ) FIELD DELIMITER ' ' ;
FOR EACH RECORD ord_rec FROM ord_in
  INSERT INTO ADMIN.orders (order_no, product, order_date, qty)
  VALUES (ord_no, item_name, date, item_qty) ;
NEXT RECORD
```

- *database_name* is the name of the destination database
- *options* can be one or more of the following:
 - o *-u user_name*: user name to connect to the database.
 - o *-a password*: password to connect to the database.
 - o *-z maximum multiple inserts*: maximum number of records to be inserted at one time in each bulk insert (used to improve performance)
 - o *-l logfile*: the file into which the error logging is done. stderr is the default.
 - o *-b badfile*: the file into which the bad rows that were not loaded, are written. By default badfile is put in the current directory.
 - o *-c commit_frequency*: the specified number of records before committing the transaction. The default frequency is 100 records.
 - o *-e maxerrs*: the maximum number of tolerable errors. The default number is 50 errors.
 - o *-s skipcount*: skip the specified number of rows in the first data file. If multiple files are specified, the rows are skipped only in the first file. The default is 0.
 - o *-m maxrows*: stop storing rows at the specified number.

-n: parse the commands file and display errors, if any, without doing the database load.

dbschema

The dbschema utility generates statements to recreate the specified database elements and data. This can be a great tool for backing up database schema designs, and building scripts to recreate databases.

Usage:

```
dbschema [ -d ] [-u user_name ] [-a password ] [ -o outfile ]  
          [ -p [ user_name.]procedure_name [ , ... ] ]  
          [ -t [ user_name.]table_name [ , ... ] ]  
          [ -T [ user_name.]trigger_name [ , ... ] ]  
          [ database_name ]
```

Where:

- **-d**: in conjunction with the **-t** option, specifies that dbschema generates SQL INSERT statements for data in the tables, in addition to CREATE statements.
- **-u user_name**: user name to connect to the database.
- **-a password**: password to connect to the database.
- **-o outfile**: redirects the output to the specified file. The default is stdout.
- **-t [user_name.]table_name [, ...]**: a comma-separated list of tables and views for which definitions should be generated. Specify a list of specific tables, or use the % to generate definitions for all tables.
- **-p [user_name.]procedure_name [, ...]**: a comma-separated list of stored procedures for which definitions should be generated. The table names in the list can include the % and underscore (_) characters, which provide pattern-matching semantics: the % matches zero or more characters in the procedure name while the underscore (_) matches a single character in the procedure name.
- **-T [user_name.]trigger_name [, ...]**: a comma-separated list of triggers for which definitions should be generated. The table names in the list can include the % and underscore (_) characters, which provide pattern-matching semantics: the % matches zero or more characters in the trigger name while the underscore (_) character matches a single character in the trigger name.

By default, dbschema generates definitions for resources owned by the current user. Use the optional *user_name* qualifier to specify a trigger owned by a different user.

isql

The isql utility is a command-line SQL client for c-tree SQL. Using the proper queries it's possible to manage all database items, like tables, users and triggers.

Usage:

```
isql [-s<script>] [-u username] [-a passwd] port@ipaddress:databasename
```


Where:

- *script* is a text file encoded with UTF-8 that contains SQL commands to execute. This feature is useful if you need to launch one or more SQL commands without user interaction.
- *username* is the login user, e.g. 'admin'
- *passwd* is the login password, e.g. 'ADMIN'
- *port* is the port on which the c-tree SQL Server is listening, e.g. '6597'
- *ipaddress* is the address on which the c-tree SQL Server is listening, e.g. 'localhost'
- *databasename* is the name of the c-tree SQL database, e.g. 'ctreeSQL'

If the login is successful the following prompt will appear and you will be able to input SQL commands:

```
FairCom/isql.exe Version 11.2.0.10214 (Build-160621)

FairCom Corporation (C) 1992-2016.
Dharma Systems Pvt Ltd-Dharma Systems Inc (C) 1992-2016.

ISQL>
```

Consult the SQL Reference Manual (sqlref.pdf) installed with the product for information about the supported SQL syntax.

sa_admin

The command-line system administrator program, *sa_admin*, can be used to perform many user operations directly from shell scripts.

Usage:

```
sa_admin [-a username] [-p passwd] [-f filepasswd] [-s servername] <option>
```

Where:

- *username* is the login user, e.g. 'admin'
- *passwd* is the login password, e.g. 'ADMIN'
- *filepasswd* is the file password, if applicable
- *servername* is the c-tree server name, e.g. 'FAIRCOMS'
- *option* is one of the following:

-oua	Add a user account
-oud	Change user account description
-oue	Change user account extended settings
-oug	Add a user to a group

-oul	List user accounts
-oum	Change user account memory limit
-oup	Change user account password
-our	Delete a user account
-ous	Show user account information
-oux	Remove a user from a group
-oga	Add a group
-ogd	Change group description
-ogl	List groups
-ogm	Change group memory limit
-ogr	Delete a group
-ogs	Show group information
-ofg	Change file group
-ofl	List files matching filename
-ofo	Change file owner
-ofp	Change file password
-ofs	Change file permissions

Graphical utilities

Most graphical utilities are written in .NET and they're provided as executable files for the Windows platform.

See [.NET utilities](#) for details.

Some utilities are available also as Java programs so they can run on both Windows and Unix platforms.

See [Java utilities](#) for details.

.NET utilities

The following .NET graphical utilities are available for the Windows platform:

- [c-treeACEMonitor](#)
- [c-treeConfigManager](#)
- [c-treeGauges](#)
- [c-treeISAMExplorer](#)
- [c-treeLoadTest](#)
- [c-treeLogAnalyzer](#)
- [c-treePerfMon](#)
- [c-treeQueryBuilder](#)
- [c-treeSecurityAdmin](#)
- [c-treeSqlExplorer](#)
- [c-treeTPCATEST](#)
- [DrCtree](#)
- [ErrorViewer](#)

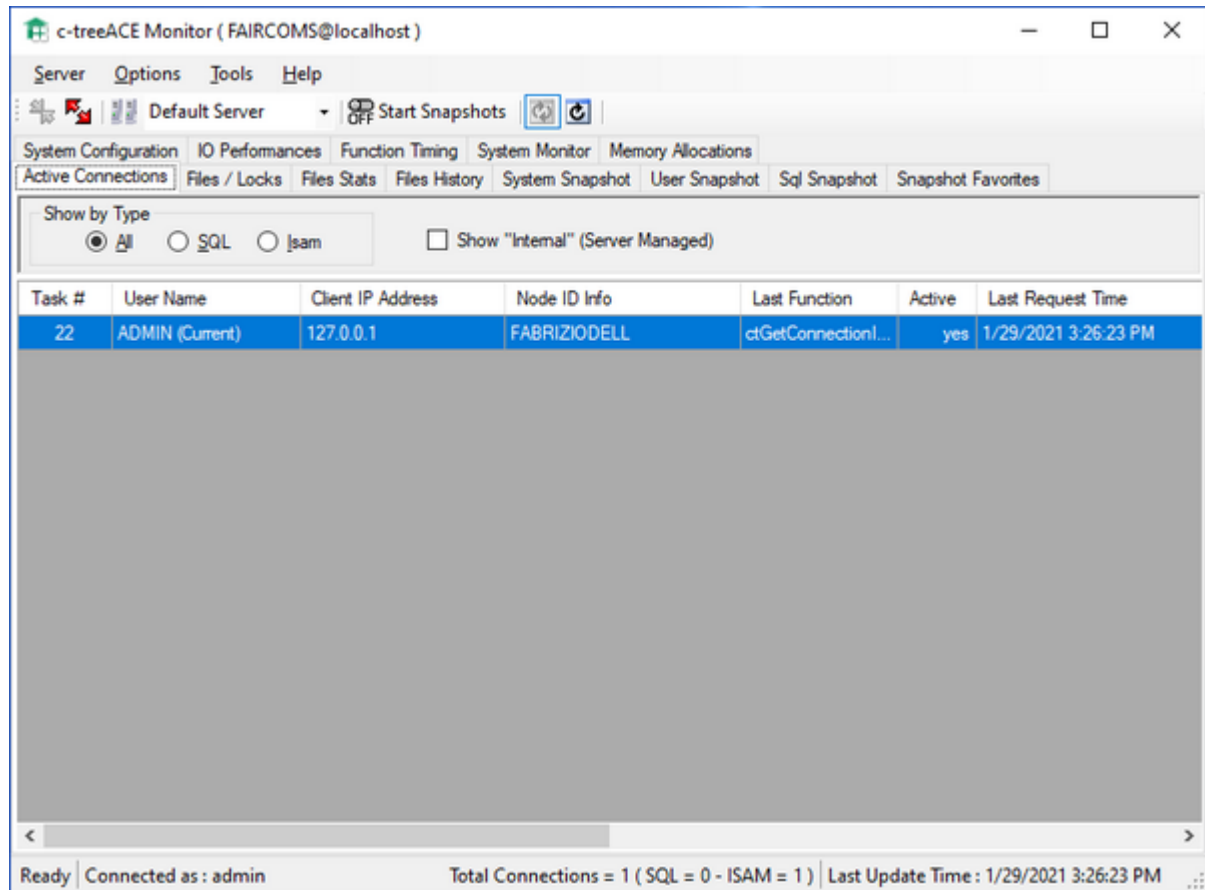
The executable files are installed under the “tools\gui” folder.

All these utilities can run on the same machine as the c-tree Server as well as on a remote machine. The utilities that require to connect to the c-tree Server prompt you for connection details at startup:

The screenshot shows a 'Connection Options' dialog box. It has two main sections: 'Server Connection' and 'User Options'. In the 'Server Connection' section, there is a 'Server Name' dropdown menu with 'FAIRCOMS' selected, a 'Machine Name / IP Address' text box with 'localhost', and an '@' symbol between them. Below this is an 'SSL Mode' dropdown menu with 'None' selected, and a 'Certificate' text box with a browse button ('...'). In the 'User Options' section, there is a 'User Name' text box with 'admin' and a 'User Password' text box with masked characters ('*****'). At the bottom are 'OK' and 'Cancel' buttons.

c-treeACEMonitor

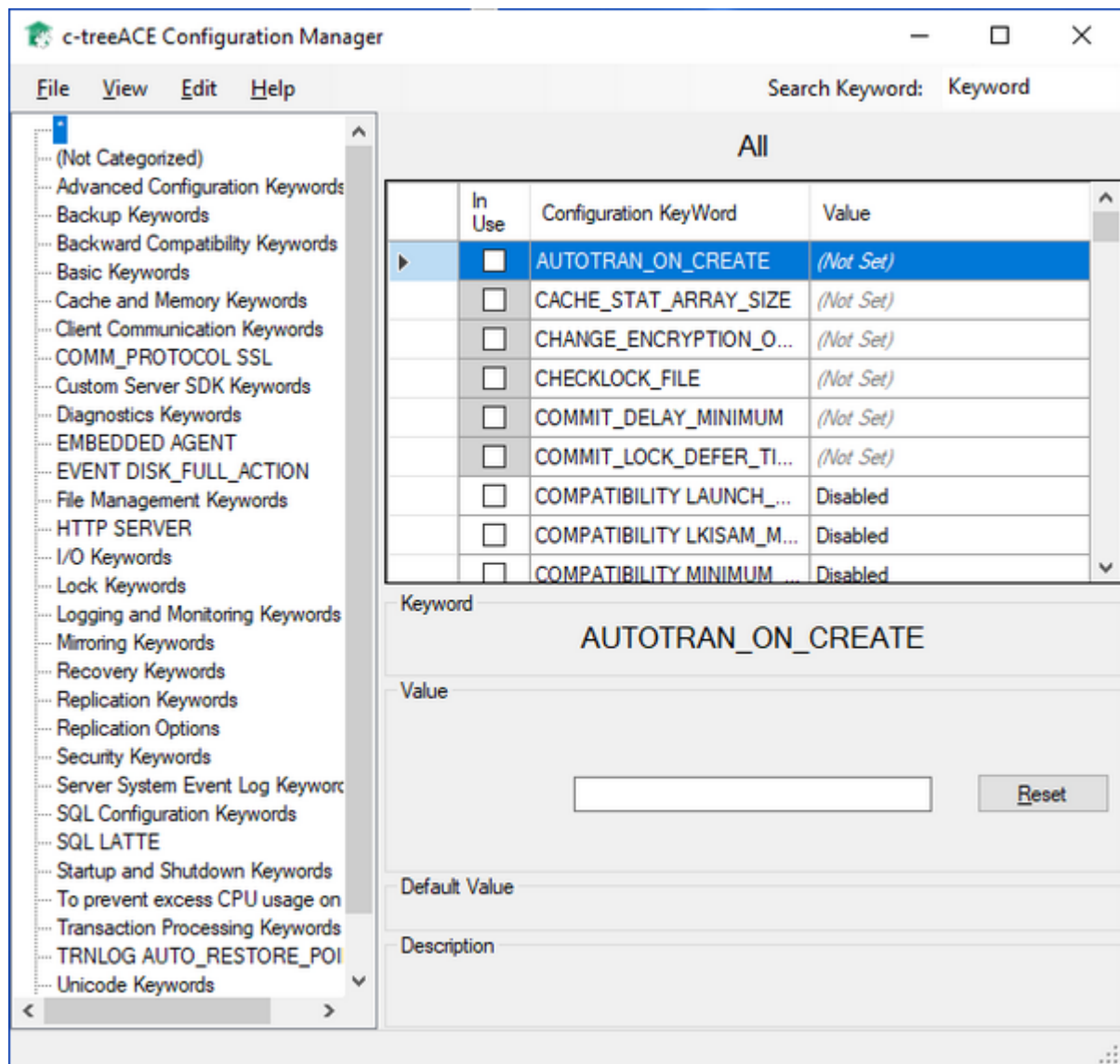
c-treeMonitor is a graphical utility provided for the Windows platform. It monitors all the activities of the c-tree Server through a graphical window made of multiple panels.



Refer to c-tree Server Administrator's Guide for additional information on this utility.

c-treeConfigManager

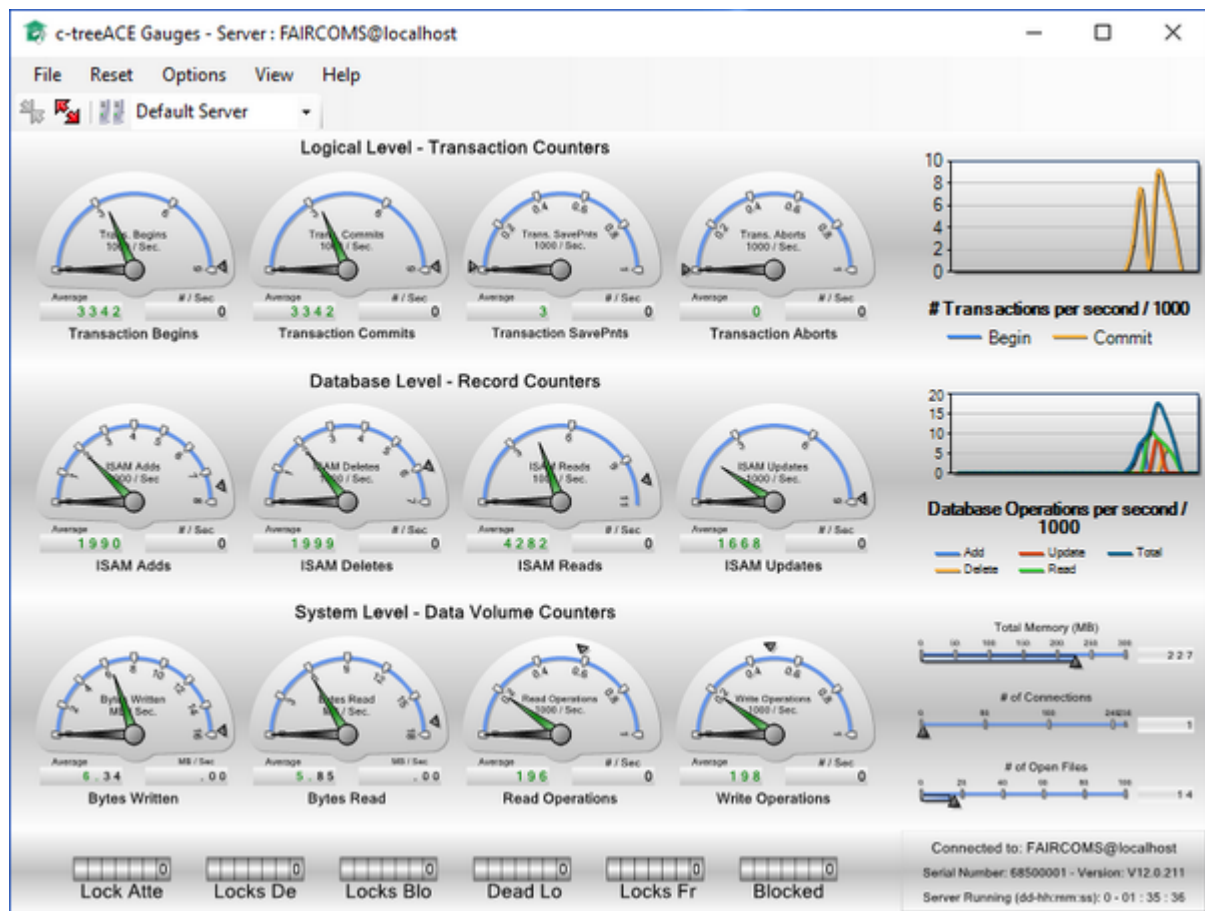
c-treeConfigManager is a graphical utility provided for the Windows platform. It allows to edit the c-tree server configuration file (ctsrvr.cfg) using a graphical interface.



Refer to c-tree Server Administrator's Guide for additional information on this utility.

c-treeGauges

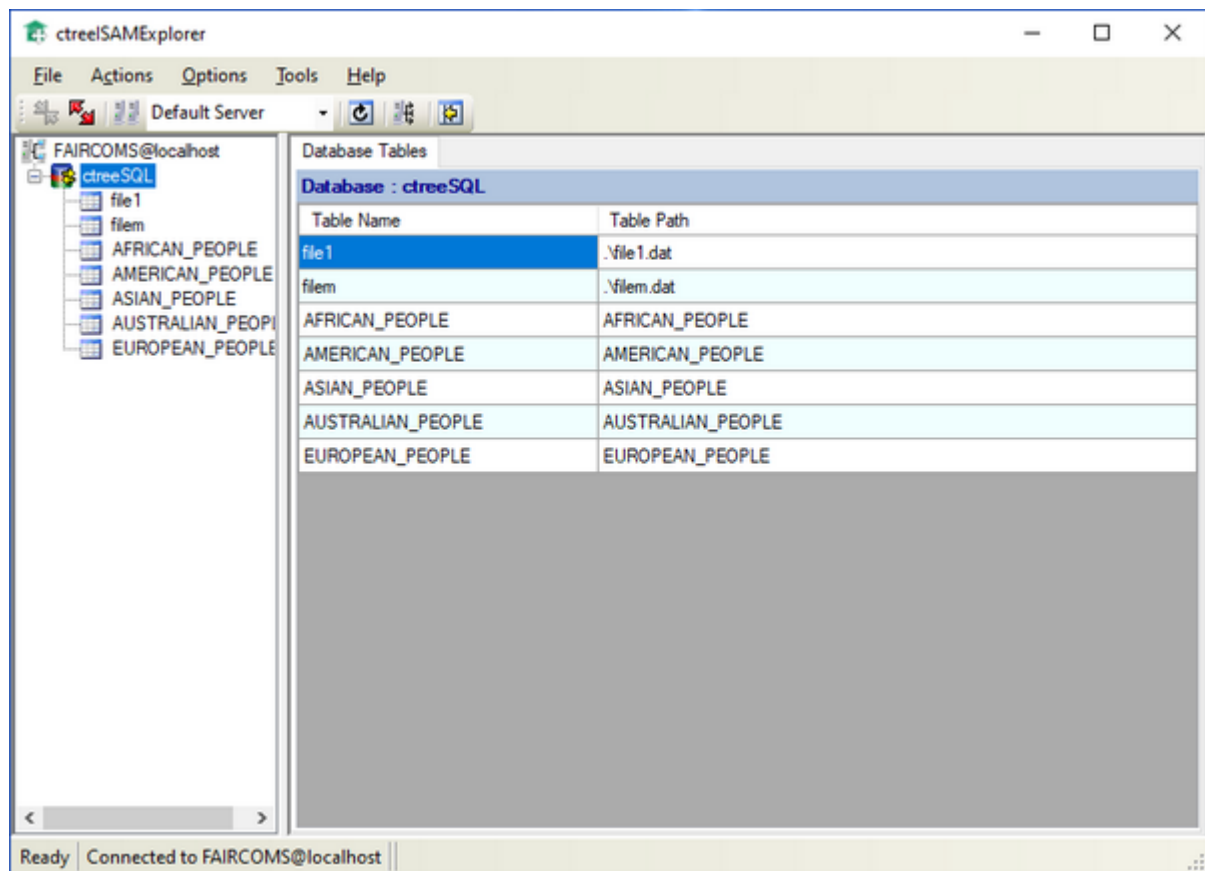
c-treeGauges is a graphical display of performance metrics provided for the Windows platform. While not intended as a full monitoring solution, c-treeGauges makes it easy to begin understanding the internal workings of the c-tree server.



Refer to c-tree Server Administrator's Guide for additional information on this utility.

c-treeISAMExplorer

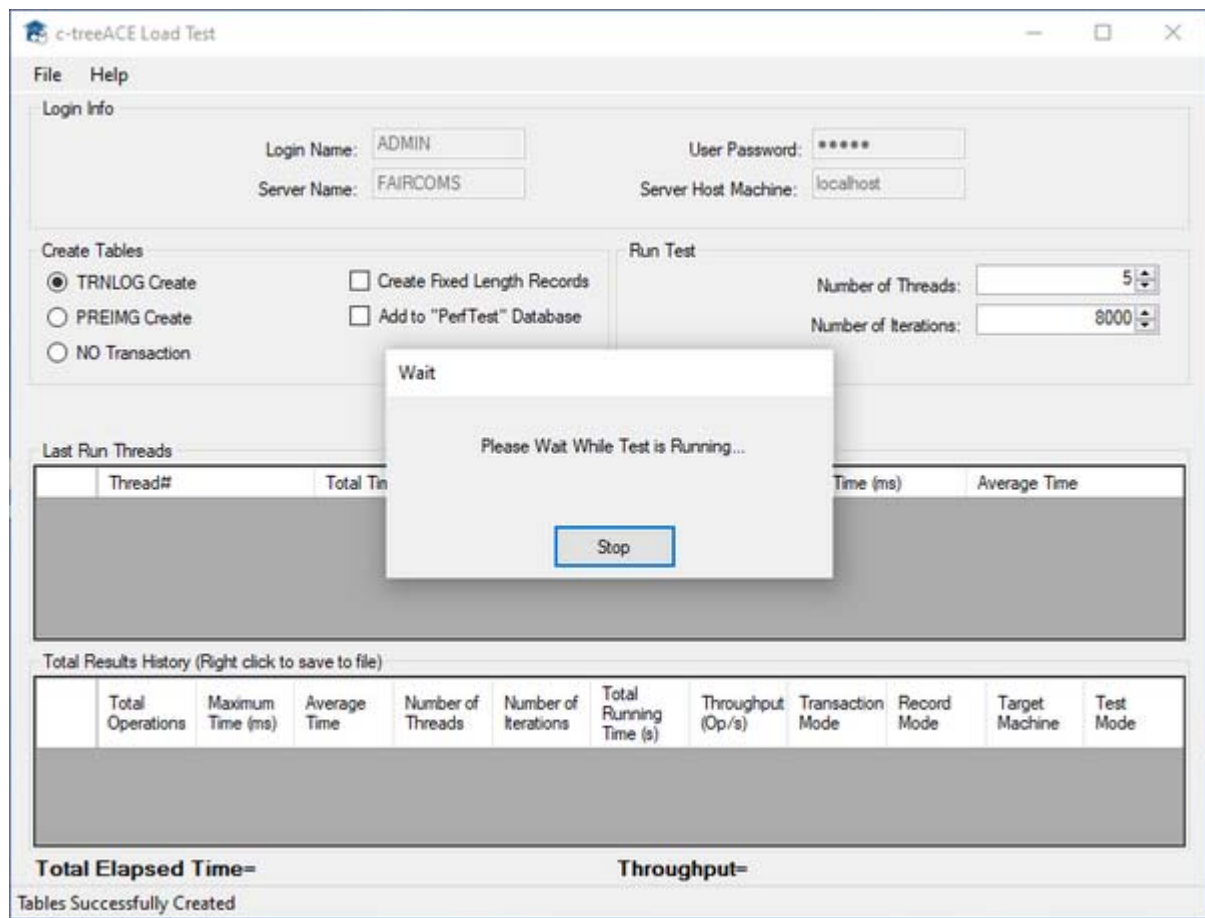
c-treeISAMExplorer is a graphical tool to create, view, and modify your c-tree databases without requiring SQL.



Refer to c-tree Server Administrator's Guide for additional information on this utility.

c-treeLoadTest

c-treeLoadTest is a graphical utility provided for the Windows platform that quickly test and analyze throughput on specific machines and configurations. With multiple options to simulate various data streams, you can easily pinpoint performance bottlenecks.



Refer to c-tree Server Administrator's Guide for additional information on this utility.

c-treeLogAnalyzer

c-treeLogAnalyzer is a graphical utility provided for the Windows platform. It allows to easily review the CTSTATUS.FCS log file that is created and constantly updated the c-tree server.

As the utility starts, click on the *File* menu and choose *Open*. Browse for the CTSTATUS.FCS file stored in the c-tree server directories and open it.

The screenshot shows the 'c-treeACE Status Log Analyzer' window. The title bar indicates the file path: C:\Programmi\veryant\c-treeRTG v3.0.0\data\CTSTATUS.FCS. The menu bar includes File, Options, Tools, and Help. Below the menu is a toolbar with icons for opening files, saving, and printing. The main window has tabs for 'Filtered Event View', 'Filtered Text View', 'Filtered Graphic View', 'Raw View', and 'Messages'. The 'Raw View' tab is selected, displaying a table of log entries.

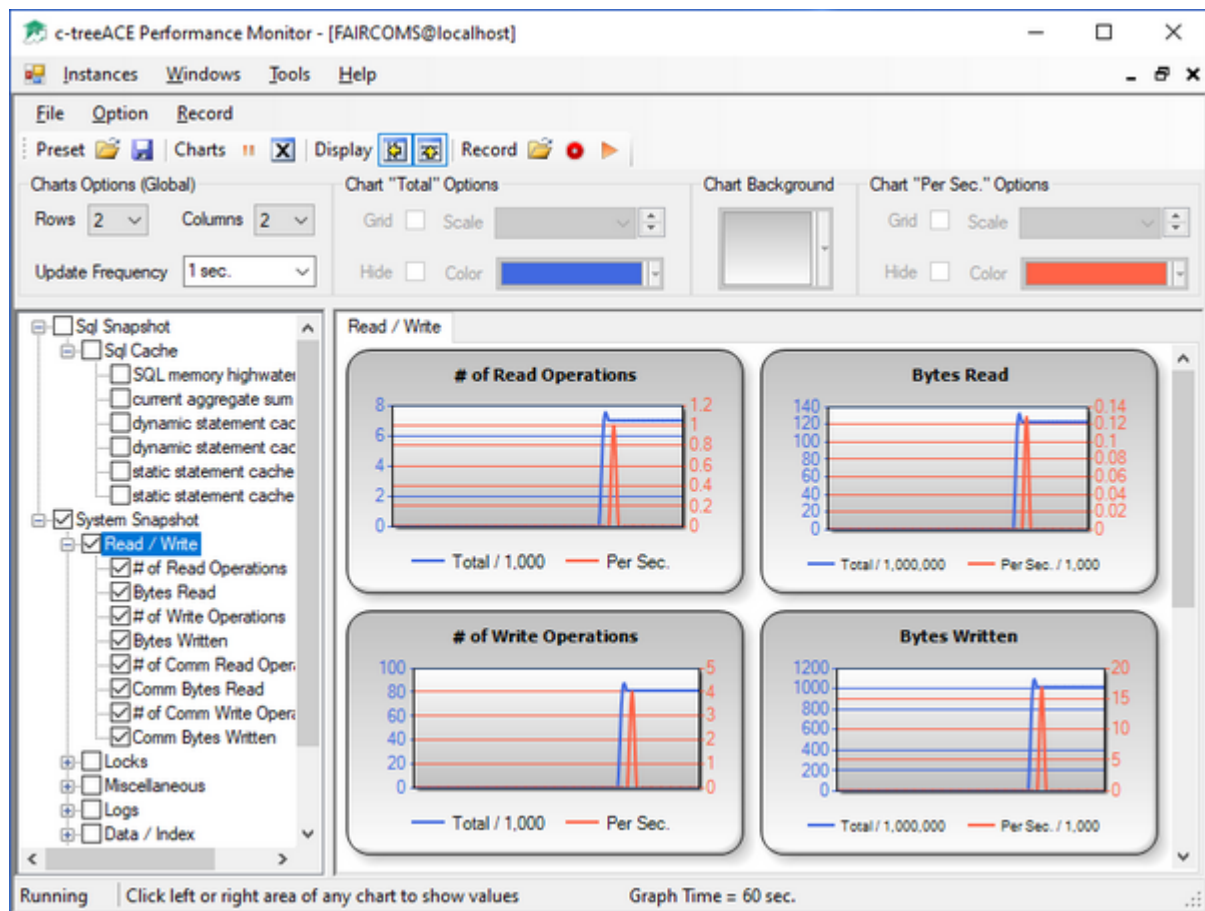
TimeStamp	User #	Error #	Description
Fri Jan 29 12:59:51 2021	00001		LICENSE: License file in use: \ctsrvr68500001.lic
Fri Jan 29 12:59:51 2021	00001		Using configuration file <../config/ctsrvr.cfg>
Fri Jan 29 12:59:51 2021	00001		DH_JVM_OPTION_STRINGS=-Xms100m;-Xmx300m;-XX:+UseG1GC
Fri Jan 29 12:59:51 2021	00001		Windows native reader/writer lock support enabled
Fri Jan 29 12:59:51 2021	00001	I2707	Process ID: 45780
Fri Jan 29 12:59:51 2021	00001	I0455	Alternative server name...
Fri Jan 29 12:59:51 2021	00001	I0455	FAIRCOMS
Fri Jan 29 12:59:51 2021	00001	I0492	Compatibility bit maps: (1)00002000x (2)00020000x (3)000000000x (4)000000000x (5)000000000x
Fri Jan 29 12:59:51 2021	00001	I0491	Diagnostic bit maps: (1)400000000x (2)000000000x (3)000000000x
Fri Jan 29 12:59:51 2021	00001	I0490	64-bit File Address Support
Fri Jan 29 12:59:51 2021	00001	I0489	6 Byte Transaction Numbers
Fri Jan 29 12:59:51 2021	00001	I0498	Commit Read Lock Support
Fri Jan 29 12:59:51 2021	00001	I0488	NOWAIT ursema enabled
Fri Jan 29 12:59:51 2021	00001	I2702	Smart File Sync Support
Fri Jan 29 12:59:51 2021	00001	I2703	ctIOSEMA I/O semaphore optimization is enabled
Fri Jan 29 12:59:51 2021	00001		ctIOSEMAmemfile I/O semaphore optimization is enabled
Fri Jan 29 12:59:51 2021	00001		ctFeatGNSEMAhsh index cache mutex optimization is enabled
Fri Jan 29 12:59:51 2021	00001		ctFeatDBSEMAhsh data cache mutex optimization is enabled
Fri Jan 29 12:59:51 2021	00001		Memory suballocator allocation unit: 16
Fri Jan 29 12:59:51 2021	00001		Startup c-treeRTG Server - V3.0.0.211(Build-210123)

Ready

Refer to c-tree Server Administrator's Guide for additional information on this utility.

c-treePerfMon

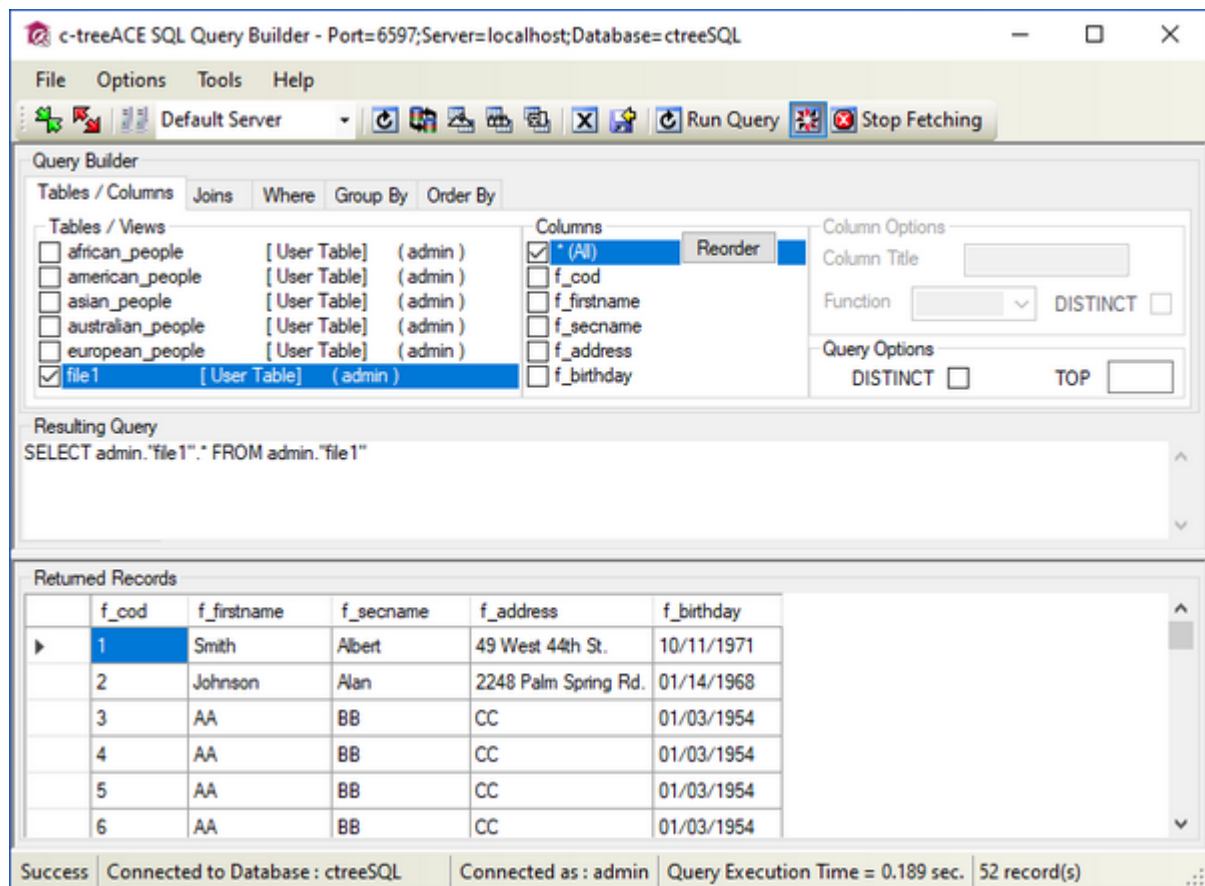
c-treePerfMon is a graphical utility provided for the Windows platform. It graphically displays critical c-tree operational parameters in real time enabling you to quickly spot performance bottlenecks. Sorted by functional categories, c-treePerfMon allows you to hone in on the exact metrics you wish to monitor. Everything from memory usage to the number of specific ISAM operations executed is available at a glance.



Refer to c-tree Server Administrator's Guide for additional information on this utility.

c-treeQueryBuilder

c-treeQueryBuilder is a graphical utility provided for the Windows platform. It's a QBE (Query By Example) that allows you to create SQL queries from the GUI instead of writing SQL code.



1. Select the table you want to query from the *Tables/Views* list.
2. Select the column(s) you want to show from the *Columns* list.
3. Optionally switch to the pages *Joins*, *Where*, *Group By* and *Order By* to specify additional options and criteria.

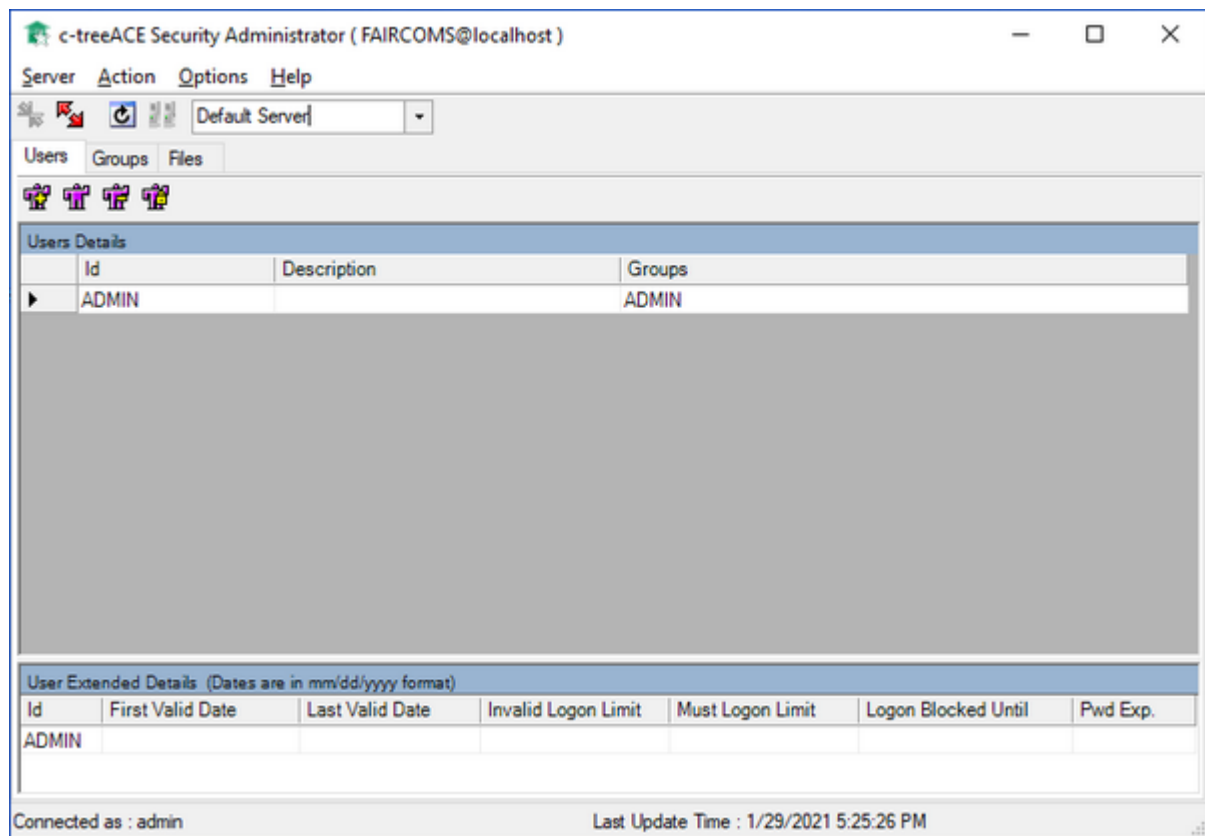
The SQL query is updated in the *Resulting Query* field in real time.

The results of the query are updated in the *Resulting Records* field in real time.

Refer to c-tree Server Administrator's Guide for additional information on this utility.

c-treeSecurityAdmin

c-treeSecurityAdmin is a graphical utility provided for the Windows platform. It manages the users and permissions through a graphical window composed of multiple panels.



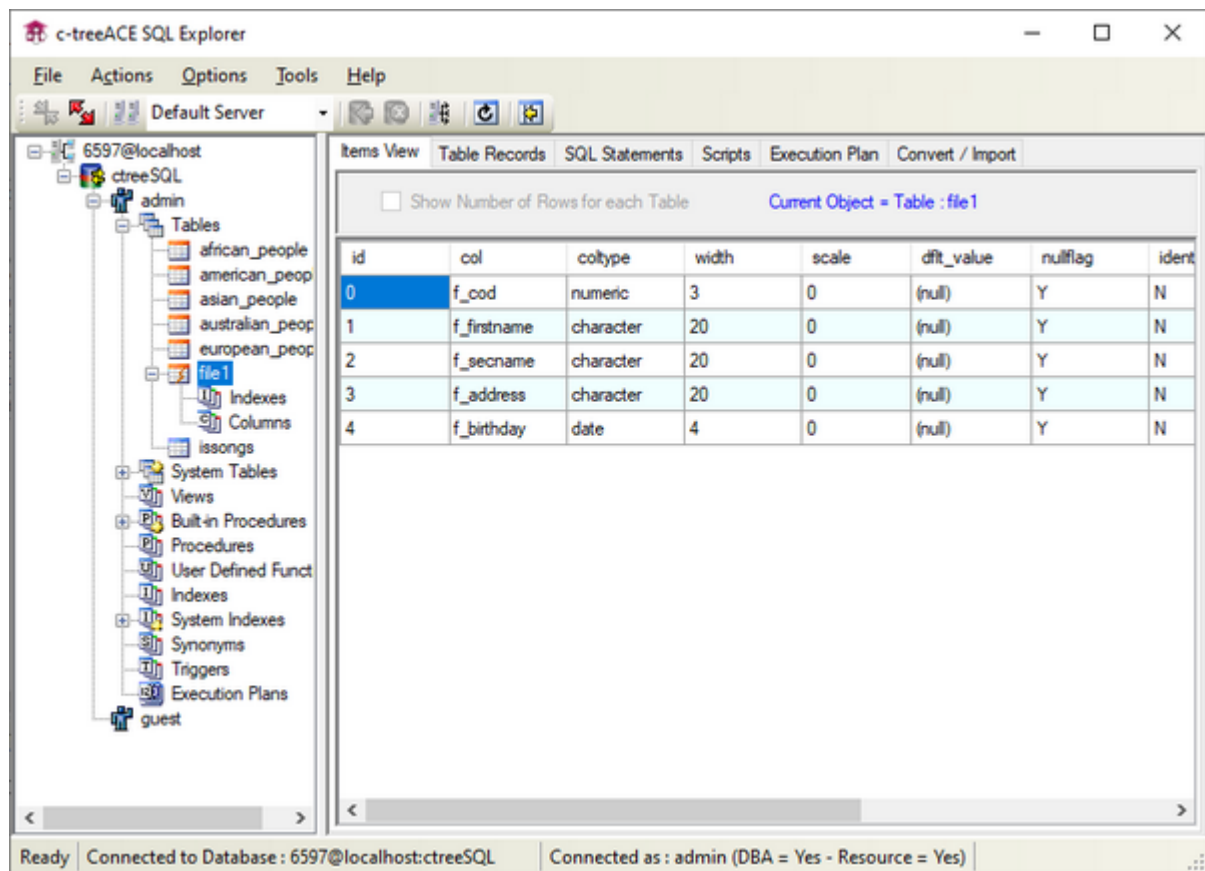
ctreeSecurityAdmin allows to:

- Add, Delete and Modify Users
- Change User Passwords
- Create, Modify and Delete Groups
- Modify File Security Attributes
- Change File Passwords

Refer to c-tree Server Administrator's Guide for additional information on this utility.

c-treeSqlExplorer

ctreeSqlExplorer is a graphical DBA (DataBase Administration) utility provided for the Windows platform.



From the tree menu on the left it's possible to select the database items you wish to monitor or manage. When an item is selected, the panels on the right allow to perform the possible operations on it.

The icon before table names allows you to distinguish pure SQL tables from ISAM files that have been sqlized using the [ctutil](#) utility. The icon with a blue bar identifies a pure SQL table, while the icon with an orange bar identifies a sqlized archive.

The *Clear Database* function drops all the tables in the database. This operation also deletes sqlized archives unless you activated [SQL_OPTION DROP_TABLE_DICTIONARY_ONLY](#) in the c-tree Server's configuration file (ctsrvr.cfg).

c-tree SQL Explorer also allows you to manage users if you start it with the following command:

```
c-treeSqlExplorer -adv
```

You can create a new database or add an existing database using the options in the *Actions* menu. Before adding an existing database, be sure that the database dbs folder has been copied to the server's *data* directory.

Refer to c-tree Server Administrator's Guide for additional information on this utility.

c-treeTPCAtest

c-treeTPCAtest is a graphical performance benchmark provided for the Windows platform.

c-treeACE TPCA Test

ISAM CTDB ODBC

Transaction Mode

☒ TRNLOG

☐ PREIMG

☐ NO Transactions

Run Details

☒ Seconds To Run: 10

☐ Transaction To Run: 1 Commits: 1

Number Of Threads: 1

Login Details

User Name: admin Password:

Server Name: FAIRCOMS Host: localhost

Run

Last Run Threads

	# Thread	Transactions	Seconds	Max	Average	Transactions Per Seconds Average
▶	01	14136	10.02s	411ms	1ms	1411.48

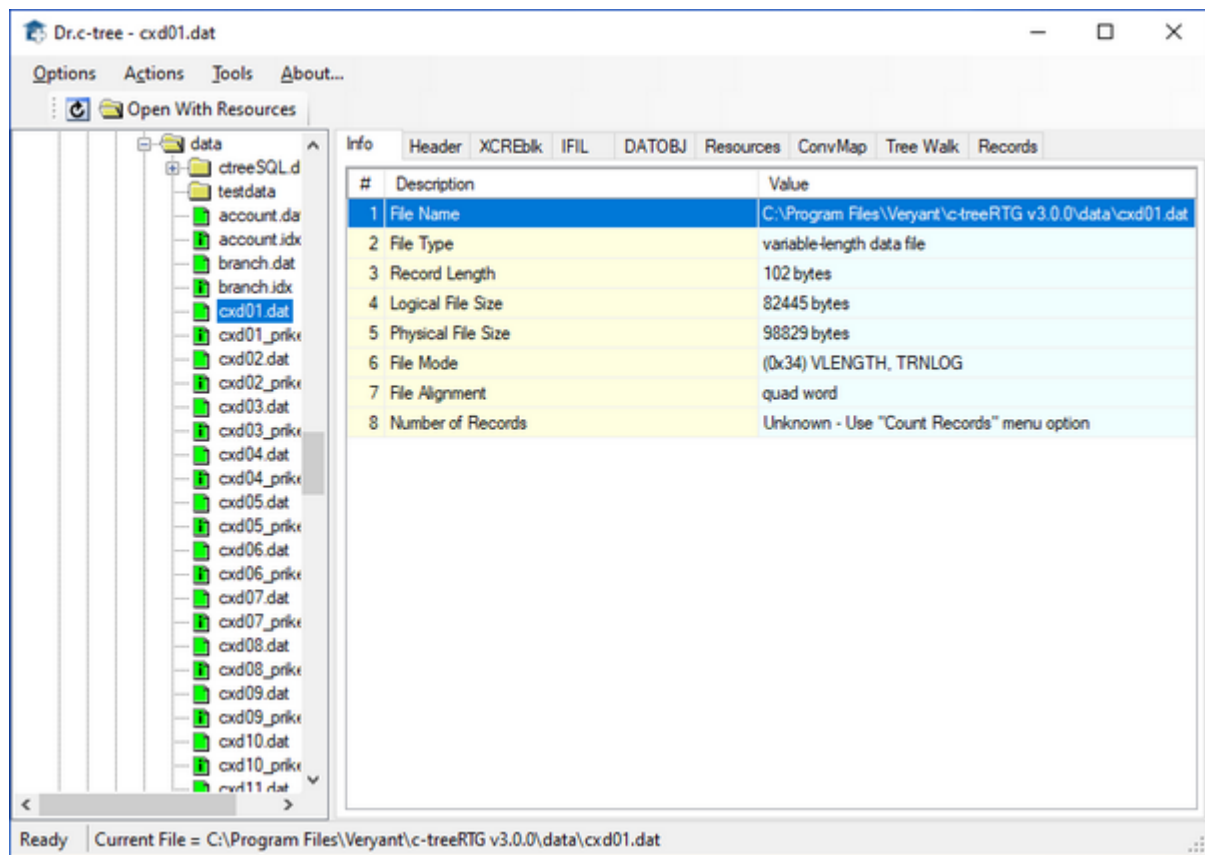
Results History (Right click to export results to file)

	Total Transactions	Max Time	Average Time	# Of Threads	Total Run Time	Transactions Per Second Average	Run Mode	Run Time
▶	14136	411ms	1ms	1	10.02s	1411.48	ISAM Serv...	1/29/2021 ...

Refer to c-tree Server Administrator's Guide for additional information on this utility.

DrCtree

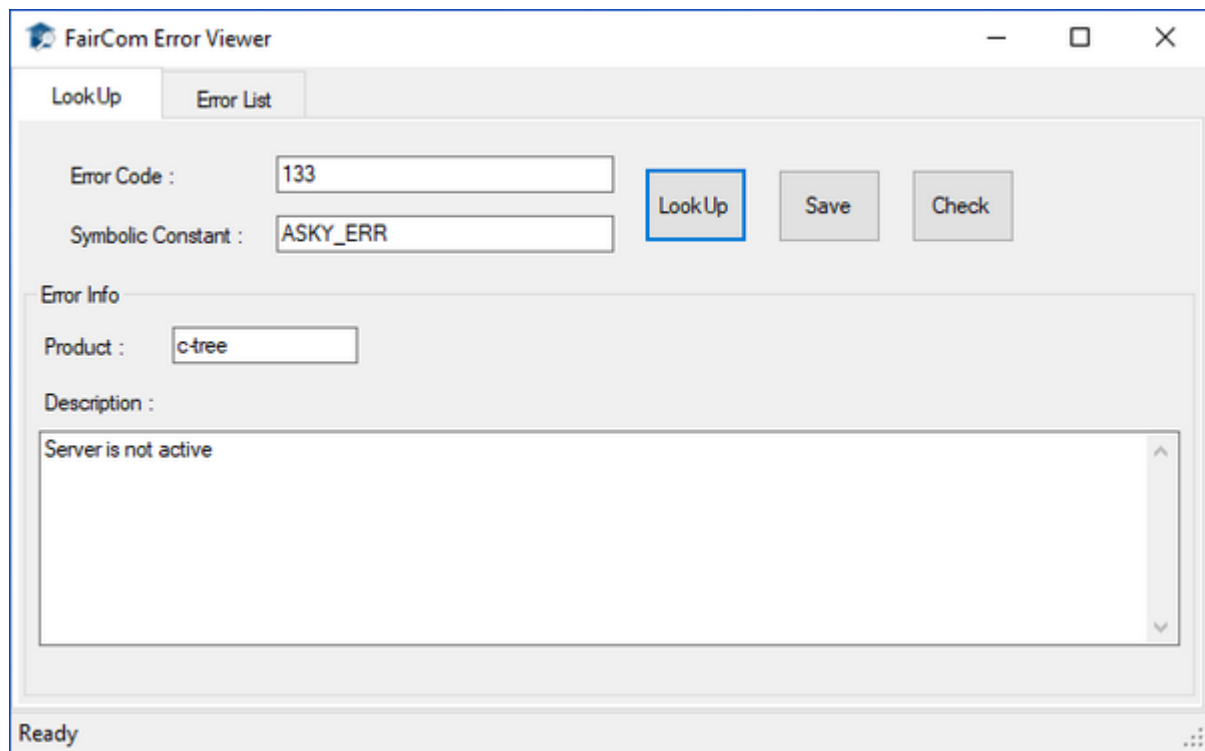
DrCtree is a graphical file management utility provided for the Windows platform. It allows you to browse for c-tree files on the local drive by using the file explorer on the left. When you select a c-tree file, information is displayed on the right. Right clicking on the file name or using the Actions menu, allows to perform file maintenance actions, like rebuild the file, for example.



Refer to c-tree Server Administrator's Guide for additional information on this utility.

ErrorViewer

ErrorViewer is a graphical utility provided for the Windows platform. It provides error codes description. Type the error code in the field and press Enter (or click the "Look Up" button) to obtain the related description. Switch to the "Error List" page to see a list of known c-tree errors.



Error codes description can be found also in this documentation at [Error Codes](#).

Java utilities

The following Java graphical utilities are available:

- [c-treeACEExplorer.jar](#)
- [c-treeACEMonitor.jar](#)
- [DrCtree.jar](#)
- [ErrorViewer.jar](#)
- [RTGConfig.jar](#)

The jar files are installed under the “tools/java” folder.

You can run these utilities by changing to the “tools/java” and running a command like:

```
java -jar <utility_name>.jar
```

Note - On Windows, having the software installed under “C:\Program Files”, be sure to have administrator privileges before running the java command, otherwise some features may not work properly.

All these utilities can run on the same machine as the c-tree Server as well as on a remote machine. The utilities that require to connect to the c-tree Server prompt you for connection details at startup:

Connection Options

Server Connection

Server Name @ Host Name / IP Address

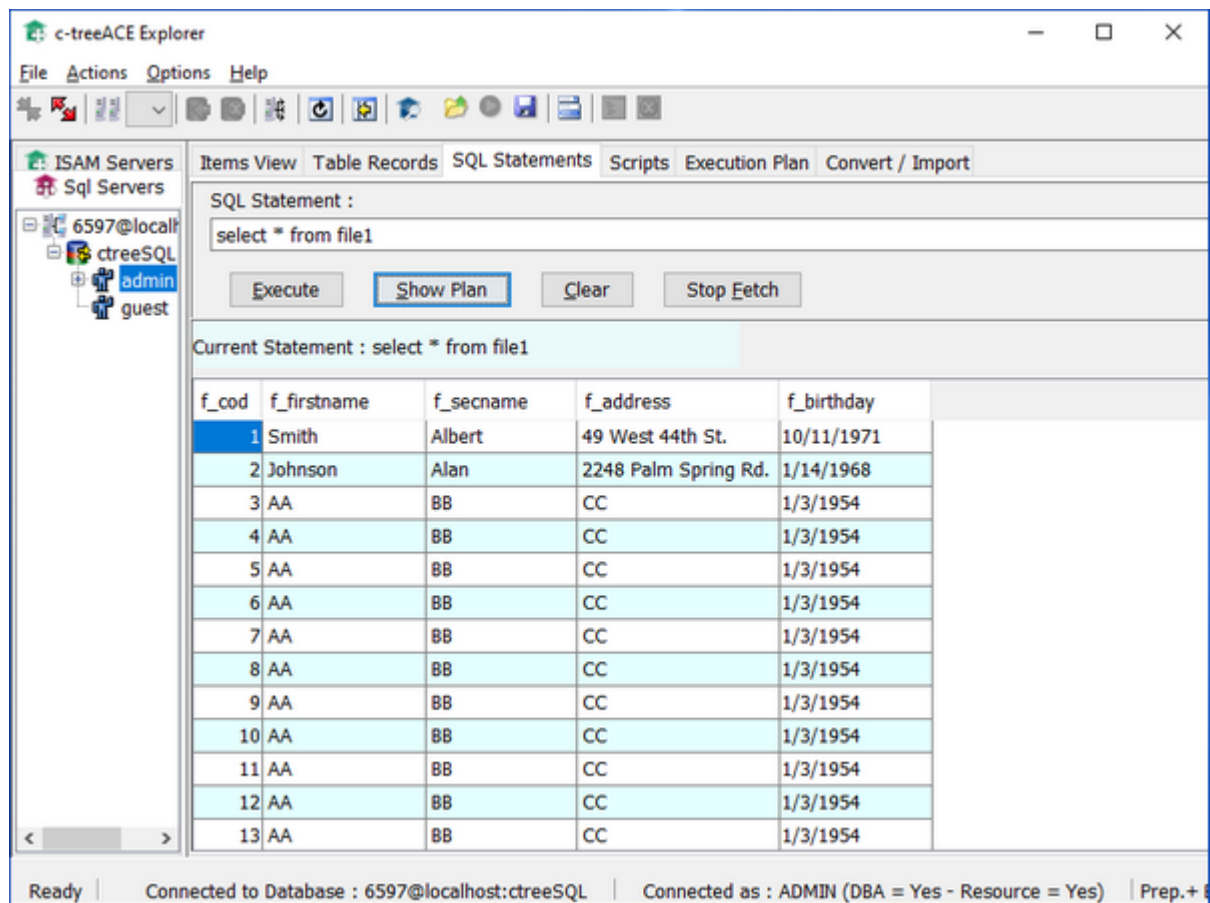
User Options

User Name : User Password :

OK Cancel

c-treeACEExplorer.jar

c-treeACEExplorer allows you to manage your SQL database with or without SQL code. It offers an ISAM view and a SQL view on the database.



The *Clear Database* function drops all the tables in the database. This operation also deletes sqlized archives unless you activated `SQL_OPTION DROP_TABLE_DICTIONARY_ONLY` in the c-tree Server's configuration file (ctsrvr.cfg).

Refer to Faircom's documentation for more information about this utility.

c-treeACEMonitor.jar

c-treeACEMonitor is a collection of various features that .NET utilities provide in separate tools. It allows you to monitor active connection as well as analyze log files, manage users and monitor performance.

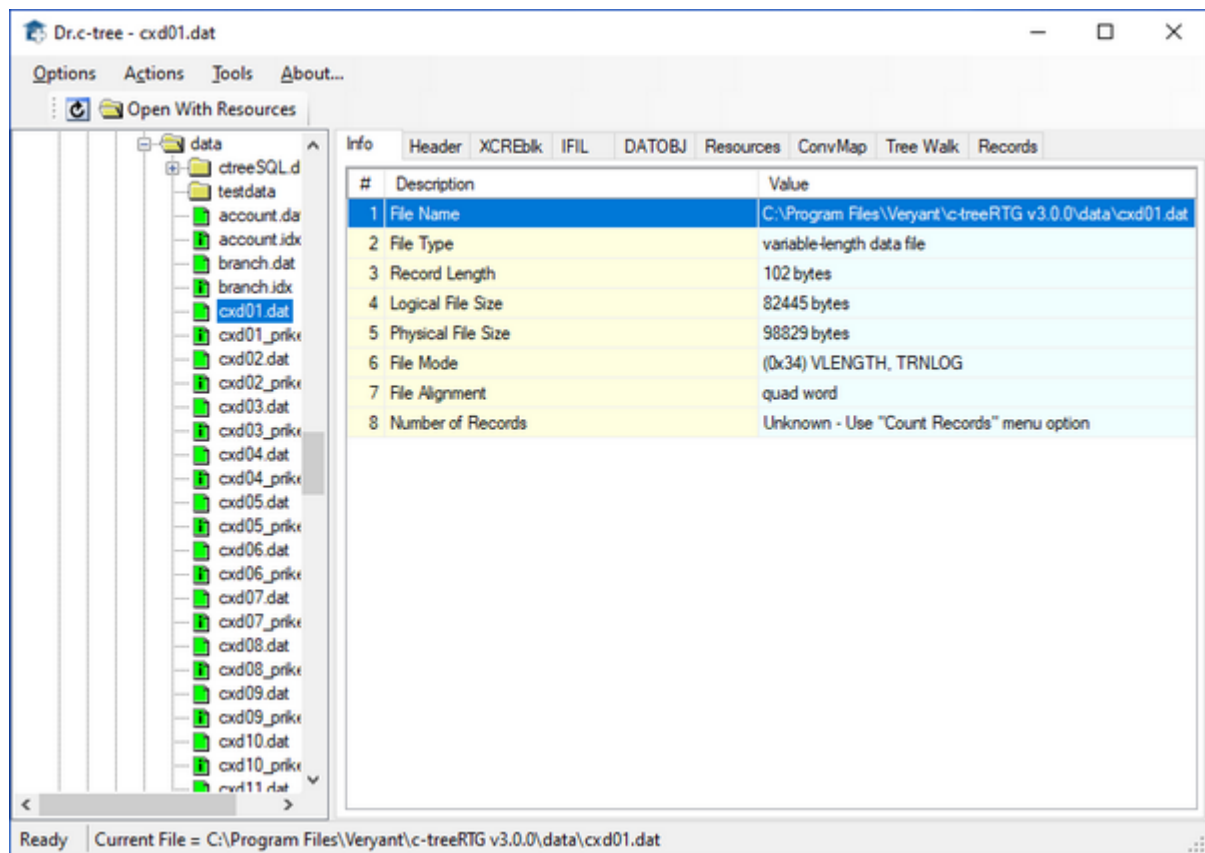
Task #	Client IP	Node ID	Last Function	Active	Last Request Time	Last TRANSEG Time	Logon Time	Files	Memory	Comm Info
22	127.0.0.1	Fabrizio...	USERINFO	yes	1/29/21, 6:04:29 PM	n.a.	1/29/21, 5:59:41 PM	0	59128	FSHAREMM
25	127.0.0.1	RTG_NEXTPREV	no	no	1/29/21, 6:04:29 PM	n.a.	1/29/21, 6:04:27 PM	2	133936	FSHAREMM

Ready Connected as : ADMIN Total Connections = 2 (SQL = 0 - ISAM = 2) Last Update Time : 1/29/21, 6:04:29 PM

Refer to Faircom's documentation for more information about this utility.

DrCtree.jar

DrCtree allows you to browse for c-tree files on the local drive by using the file explorer on the left. When you select a c-tree file, information is displayed on the right. Right clicking on the file name or using the Actions menu, allows to perform file maintenance actions, like rebuild the file, for example.



Refer to Faircom's documentation for more information about this utility.

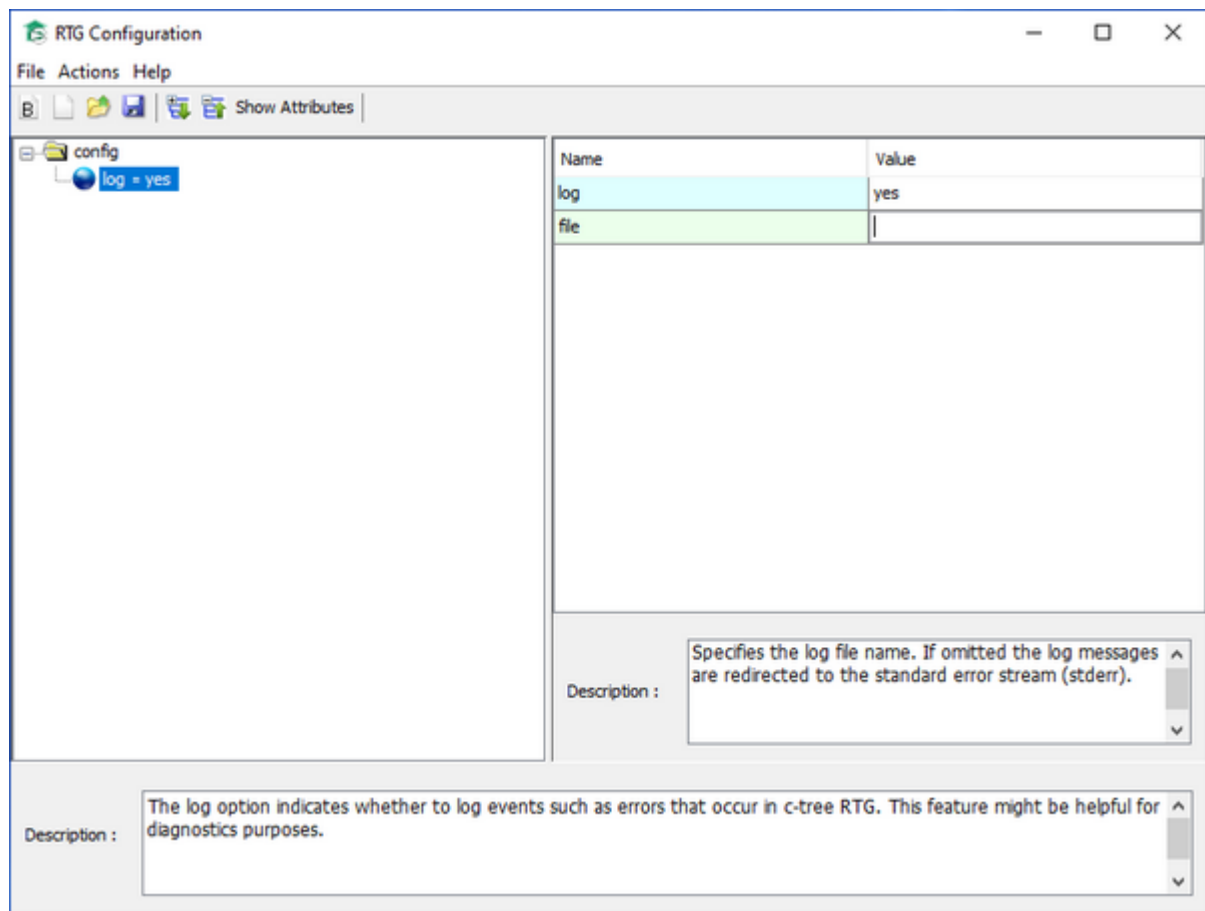
ErrorViewer.jar

ErrorViewer provides error codes description. Type the error code in the field and press Enter (or click the "Look Up" button) to obtain the related description. Switch to the "Error List" page to see a list of known c-tree errors.

The screenshot shows a window titled "Error Viewer" with standard Windows window controls (minimize, maximize, close). Inside the window, there are two tabs: "Look Up" and "Error List". The "Look Up" tab is active. It contains two input fields: "Error Code :" with the value "133" and "Symbolic Constant :" with the value "ASKY_ERR". To the right of these fields are three buttons: "Look Up" (highlighted with a blue dashed border), "Save", and "Update". Below these fields is a section titled "Error Info". It contains a "Product :" field with the value "c-tree" and a "Description :" label above a large text area. The text area contains the message "Server is not active". At the bottom of the window, there is a status bar that says "Ready".

RTGConfig.jar

RTGConfig allows you to create and maintain client side configuration files (CTREE_CONF). It represents the XML structure of the configuration file in a tree view and provides a description for each item. You can add, edit o remove items from the tree and the changes will be reflected in the configuration file on disc.



Refer to Faircom’s documentation for more information about this utility.

Web utilities

Web utilities can be reached via HTTP or HTTPS and used within your favourite web browser.

To use these utilities, the Faircom’s Replication Manager (Memphis) must be started and listening.

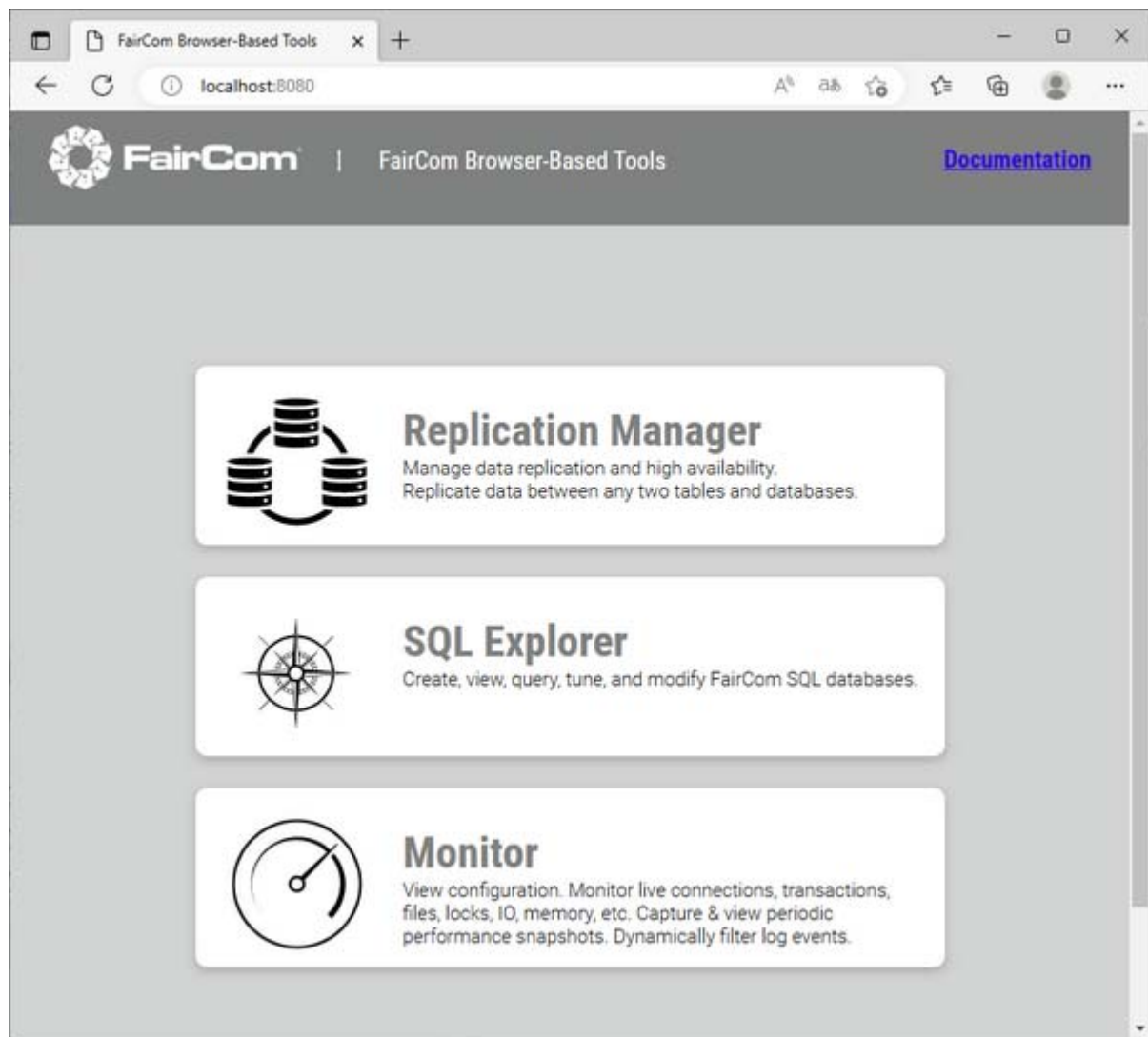
To start the Faircom’s Replication Manager, change to the folder “replication.manager/replicationmanager” and run the command:

```
memphis
```

After a successful startup, the Faicom’s Replication Manager listens for standard HTTP connections on the port 8080 and for secure HTTP connections on the port 8443.

Note - The default ports can be changed by editing the *cthttpd.json* file under the “replication.manager/replicationmanager/config” folder.

Navigate either to “http://localhost:8080” or to “https://localhost:8443” with your favourite web browser to obtain the list of available utilities:



Note - replace “localhost” with the server IP if you’re connecting from a remote PC.

The Faircom’s Replication Manager can be started on the same machine as the c-tree Server as well as on a different machine. The utilities prompt you for connection details at startup:

Connection Options

Server Connection

Server Name / Port Number : FAIRCOMS @ Host Name / IP Address: localhost
 (Prepend Port Number with a # sign)

User Options

User Name: admin User Password:

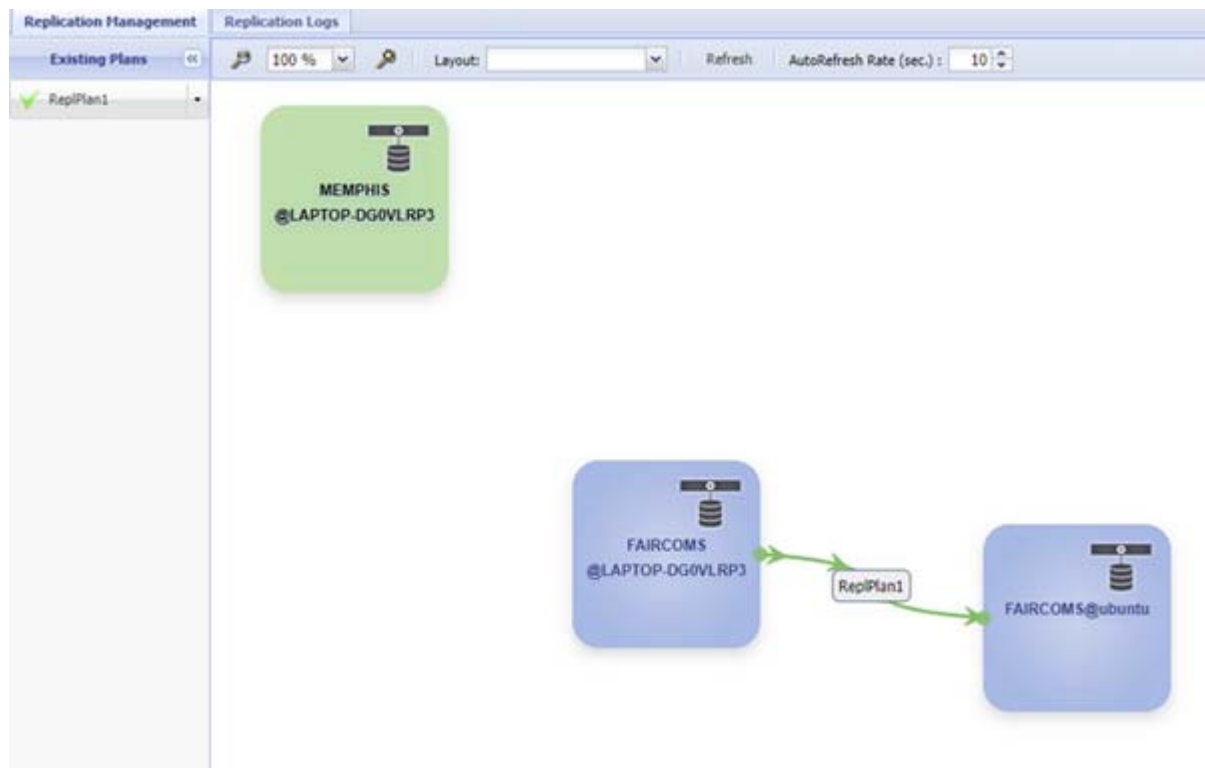
Connect Cancel

The following utilities are available:

Utility	URL	Secure URL
Replication Manager	http://localhost:8080/ReplicationManager	https://localhost:8443/ReplicationManager
SQL Explorer	http://localhost:8080/SQLExplorer	https://localhost:8443/SQLExplorer
Monitor	http://localhost:8080/AceMonitor	https://localhost:8443/AceMonitor
ISAM Explorer	http://localhost:8080/ISAMExplorer	https://localhost:8443/ISAMExplorer

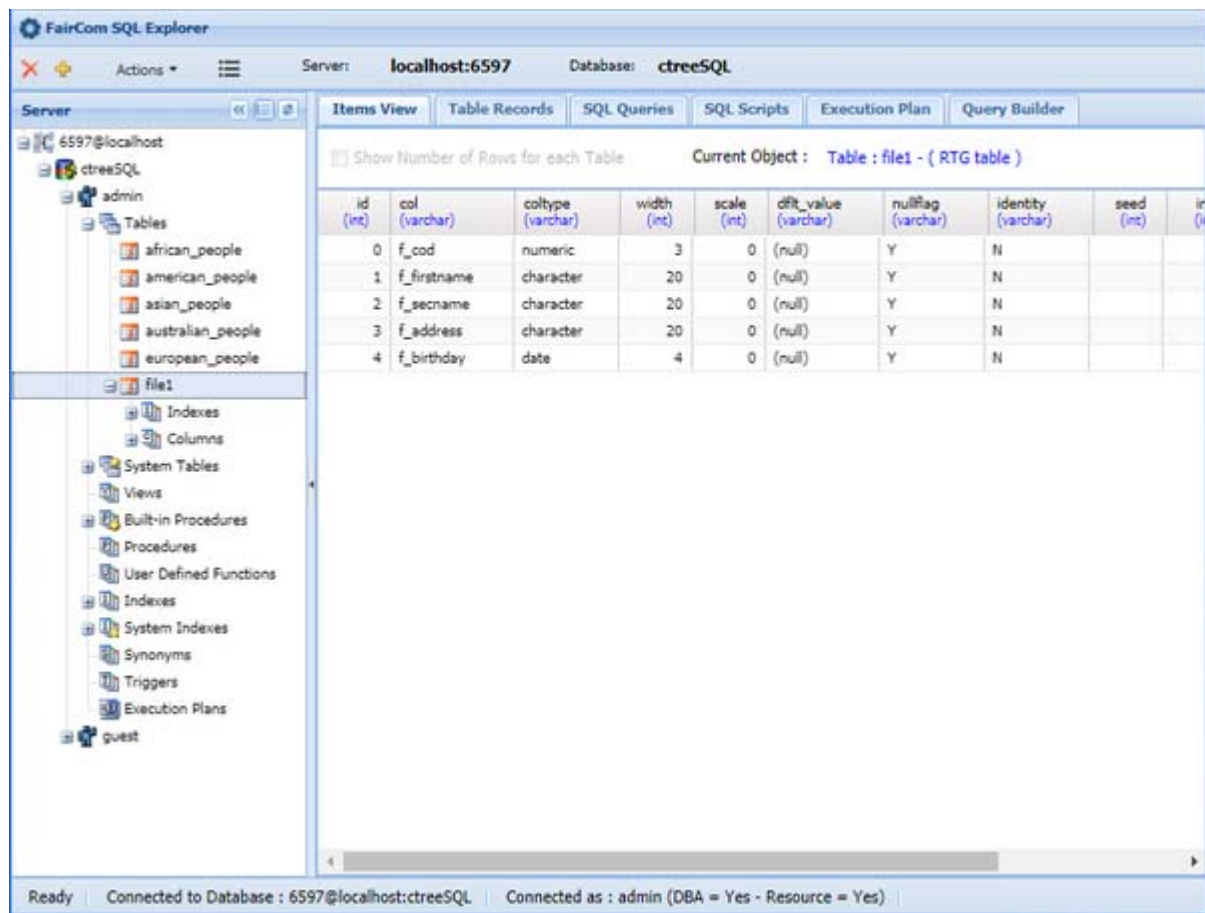
Replication Manager

Replication Manager provides a graphical overview of data replication. It allows you to configure a new replication and to monitor active replications. See [Usage and configuration](#) for more information.



SQL Explorer

SQL Explorer allows you to manage the c-treeSQL database as a relational database. You can take advantage of the features available in its context menu as well as issue SQL queries.



The *Clear Database* function drops all the tables in the database. This operation also deletes sqlized archives unless you activated `SQL_OPTION DROP_TABLE_DICTIONARY_ONLY` in the c-tree Server's configuration file (ctsrvr.cfg).

Refer to Faircom's documentation for more information about this utility.

Monitor

Monitor is a collection of various features that .NET utilities provide in separate tools. It allows you to monitor active connection as well as analyze log files and monitor performance.

The screenshot shows the FairCom Monitor web interface in a browser window. The address bar shows `localhost:8080/AceMonitor/i...`. The interface has a top navigation bar with tabs: Dashboard, Gauges, Charts, Active Connections (selected), Files / Locks, Snapshots, IO Performances, Function Timing, and System Configurat. Below the tabs, there's a 'Users' section with filters: Show By Type (All, SQL, Isam), Show "Internal" (Server Managed), and Total Connections = 1 (SQL = 0 - ISAM = 1). A table lists active connections with columns: #, T..., User Name, Client IP..., Node ID Info, Last Function, Act..., Last Reque..., Last TRAN..., Logon Time, Fi..., Me..., and Comm I... The first row shows a connection for user ADMIN (C...). Below this, a 'User Snapshot for Task # : 24' is displayed as a table with columns: #, Member, Category, Description, Value, and Value / Sec. The snapshot table lists various system metrics like client_ver, server_ver, foken, varlen, contents, unused, snapshottm, strtsum, strtmax, and scthrimbis. At the bottom, a status bar shows 'Ready', 'Connected to Server: FAIRCOMS@localhost', and 'Last Update Time : 2/1/2021, 2:48:28 PM'.

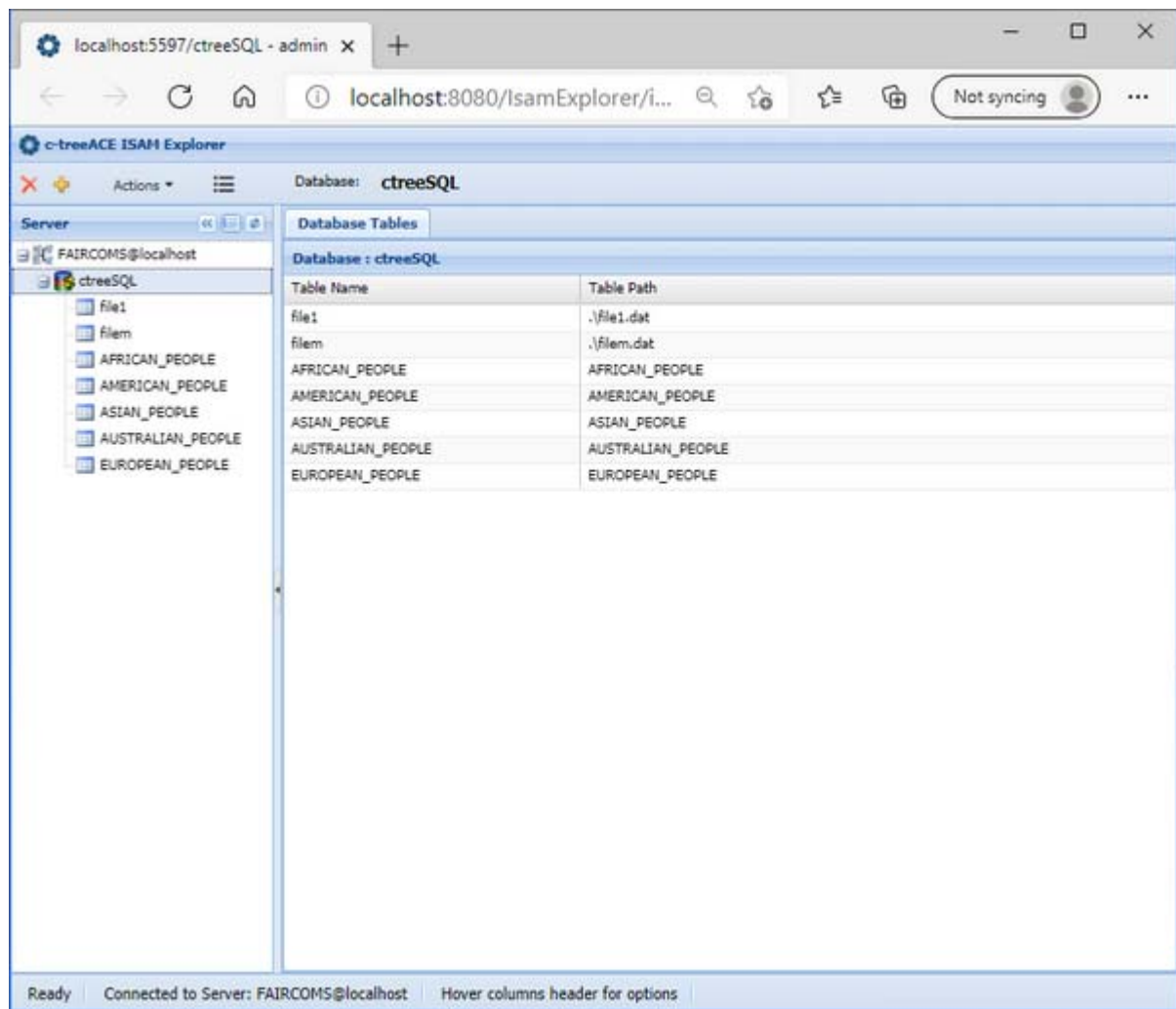
#	T...	User Name	Client IP ...	Node ID Info	Last Function	Act...	Last Reque...	Last TRAN...	Logon Time	Fi...	Me...	Comm I...
1	24	ADMIN (C...	127.0.0.1	Web Server - ACEM...	USERINFOX	yes	Feb 01, 20...	n.a.	Feb 01, 20...	0	59,168	FSHAR...

#	Member	Category	Description	Value	Value / Sec.
1	client_ver	Internal	client version of structure	2	
2	server_ver	Internal	server version of structure	2	
3	foken	Internal	length of fixed portion of snapshot	576	
4	varlen	Internal	length of variable region (if any)	0	
5	contents	Internal	bit map of var len contents	0	
6	unused	Internal	available for use	0	
7	snapshottm	Internal	snapshot time stamp: seconds since 70 (M d,Y H:m:s)	Feb 01, 2021 14:48:29	
8	strtsum	Internal	user trntime sum^ (d - H : m : s)	0 - 00 : 00 : 00	
9	strtmax	Internal	user trntime max^ (d - H : m : s)	0 - 00 : 00 : 00	
10	scthrimbis	Internal	high res timer ticks per sec^	10,000,000	

Refer to Faircom's documentation for more information about this utility.

ISAM Explorer

ISAMExplorer is a web tool to create, view, and modify your c-tree databases without requiring SQL.



Refer to Faircom's documentation for more information about this utility.

Chapter 8

Backup options

There are two solutions to backup data with c-treeRTG.

The [Offline Backup](#), performed by stopping or quiescing the c-tree server and then copying the indexed files with external commands.

The [Online Backup](#), performed by configuring the Dynamic Dump feature.

Offline Backup

c-tree files can be copied using external tools and commands without side effects only when the c-tree server is either down or quiesced. Copying the files while the c-tree server is operational might lead to a corrupted backup copy.

To perform an offline backup with the c-tree server down, follow these steps:

1. launch the [ctadmn](#) command line utility and log in with administrator credentials
2. choose option 7 (Stop Server) and confirm when prompted. This message will appear

```
Server Termination Initiated.  
  
Session Completed.
```

The shutdown can be activated also with a less interactive command line utility, [ctstop](#).

3. copy the files using system commands or third party utilities
4. restart the c-tree server

To perform an offline backup with the c-tree server quiesced, follow these steps:

1. launch the [ctadmn](#) command line utility and log in with administrator credentials
2. choose option 8 (Quiesce Server) and wait for the following message to appear:

```
Successful Quiesce.  
  
It is now safe to perform a system backup of c-tree Server's controlled files.  
Press RETURN once the backup is completed to resume the c-tree Server.  
  
Press RETURN to continue...
```

Note - while the server is quiesced, all client processes (including COBOL programs) block at the next i-o operation and wait for the ctsrvr to finish the backup.

The quiesce mode can be activated also with a less interactive command line utility, [ctquiet](#).

3. copy the files using system commands or third party utilities
4. return to the ctadmn screen and press ENTER

Online Backup

The Dynamic Dump feature of c-treeRTG provides a safe method of backing up data while the server is operational. An 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 c-tree server dynamic dump feature relies on a simple plain text file script. This script directs the time, location, and frequency that the backup should occur. Set it once, and the server can continue to make unattended data backups as long as it remains operational.

Here is a simple sample script:

```
!TIME    23:00:00
!DUMP    /mnt/backup/CTREE.BAK
!DELAY   60
!FREQ    24
!PROTECT
!FILES
*.dat
*.idx
FAIRCOM.FCS
SEQUENCEDT.FCS
!END
```

This script schedules a daily dump, starting at 11:00 PM. This backs up all data and index files along with two critical files in the server working directory. The system will wait until 11:01 PM (that is, 60 seconds delay, after the starting time of 11:00 PM) for all active transactions to complete. Then it will abort any transactions still active and begin the actual backup. Remember, the server continues normal operation once the backup process begins.

The dump data will be saved in the file named CTREE.BAK under /mnt/backup. The dynamic dump backup feature defaults to breaking-up the backup file (stream file) into multiple physical files (segments) of 1GB size.

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 and normal operation is automatically resumed once the file has been backed up. The keyword !FREQ specifies the interval (in hours) between one backup and another; in the above example we want the backup to be performed once a day.

For more information about dump script file entries, see <https://docs.faircom.com/doc/faircom-database-backup-guide/database-backup-script-format.htm>.

There are two ways to schedule dynamic dumps:

- Through server configuration: the c-tree server configuration file (ctsrvr.cfg) 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. The path to this script is relative to the server's working directory. E.g.

```
DUMP /home/user1/myscripts/dump.txt
```

or

- Through the dynamic dump utility: [ctdump](#), is a separate utility for the administrator to use at any time while the server is active.
To schedule an ad hoc dynamic dump with ctdump, while the c-tree server is running, run the ctdump utility passing admin credentials, script file name and (optionally) the c-tree server name as command line parameters; e.g.

```
ctdump admin ADMIN /home/user1/myscripts/dump.txt FAIRCOMS
```

If the command is successful, the following message is shown:

```
Dynamic Dump (home/user1/myscripts/dump.txt) has been scheduled.
```

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

Restoring From a Dynamic Dump

In the event of a catastrophic system failure (for example, a hard drive failure) that renders the transaction logs or the actual data files unreadable, it will be necessary to use a dynamic dump file to restore data to a consistent, well defined state. This is known as a dynamic dump recovery. Should you need to restore files from a c-tree server backup, you use the dynamic dump recovery utility, [ctrdump](#).

The ctrdump utility provides dynamic dump recovery. This utility is itself a c-tree server so there are important points to observe when running it:

- Be sure that no other c-tree server is running when ctrdump starts because two c-tree servers operating simultaneously interfere with each other.
- Because it is a c-tree server, ctrdump 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 ctrdump 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.

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

1. Run ctrdump by passing the dynamic dump script file to it, e.g.:

```
ctrdump /home/user1/myscripts/dump.txt
```

Note - The same script file used to perform the dump should be used to restore the dump.

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

- Each recovered/recreated file will be listed as it is completed.

- After all specified files have been recovered, a message is output indicating 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, that is, the time the dynamic dump started.
- A message indicating the dump recovery process finished successfully.

If everything is successful, you should see a similar output:

```
DR: recreating file: customer.dat
DR: recreating file: customer.idx
DR: recreating file: products.dat
DR: non-transaction file not updated during Dynamic Dump.
    Marking file clean...products.dat
DR: recreating file: products.idx
DR: non-transaction file not updated during Dynamic Dump.
    Marking file clean...products.idx
DR: recreating file: FAIRCOM.FCS
DR: recreating file: SEQUENCEDT.FCS
DR: recreating system file: S0000000.FCS
DR: recreating system file: S0000001.FCS
DR: recreating system file: L0000001.FCS
DR: restore files to dump time...Wed Sep 10 23:01:00 2014

DR: Absolute path names -
DR:   deleting system file: S0000000.FCS
DR:   deleting system file: S0000001.FCS
DR:   deleting system file: L0000001.FCS
DR: Successful Dump Restore Termination
```

Note - The following warning is written to the log file for non-transaction files: "non-transaction file not updated during Dynamic Dump".

For more information about dynamic dump, please refer to the c-tree Administration Guide.

Chapter 9

Bound Server mode

The Bound Server feature causes isCOBOL to load the c-tree engine and communicate with it in memory instead of connecting to a c-tree server through TCP/IP.

The main advantages are:

- simpler usage
- better performances while working on the local machine

The only disadvantage is that only one process can work in Bound Server mode, and it cannot create more than one instances of the c-tree server.

Because of the above limitation the Bound Server feature can be used only:

- server side in Thin Client environment where only one instance of isCOBOL Server is running
- for single user installations

To activate the feature, the following setting must appear in the isCOBOL configuration:

```
iscobol.ctree.bound_server=true
```

With the above setting, the isCOBOL engine will load ctdbsapp library instead of c-tree library.

The ctdbsapp library and its dependent libraries must be available in the library path.

Note - In version 3.0.2.454 and previous versions the c-tree server library was named ctreedbs, not ctdbsapp. If you need to start one of these c-tree versions in bound server mode, the following configuration setting must be used as well: *iscobol.ctree.bound_library=ctreedbs*.

The c-tree server configuration file (ctsrvr.cfg) and the license file should be pointed by these environment variables:

```
FCSRVR_CFG=/path/to/ctsrvr.cfg  
FCSRVR_LIC=/path/to/ctsrvr#####.lic
```

Alternatively, they can be copied to the working directory.

Note - the ctsrvr.cfg configuration file includes some relative paths that should be reviewed, as the working directory is usually different in bound server mode (it's the working directory of the process that binds the c-tree server).

The OPEN of the first file in the runtime session may take longer since the c-tree engine must be initialized. The same slowdown could be experienced during the STOP RUN, when the c-tree engine is unloaded.

Chapter 10

Troubleshooting

Problems starting the c-tree Server

The faircom command terminates during startup

When the faircom command doesn't complete and terminates suddenly, check the log file CSTATUS.FCS generated in the c-tree Server's local directory for more information. By default CTSTATUS.FCS is generated in the *data* subfolder.

The most common causes for this issue are an invalid or missing license or an invalid entry in the ctsrvr.cfg configuration file. The content of CSTATUS.FCS will explain the cause of the problem.

Failed startup due to the number of CPU cores

The following error message could be shown at startup, causing the startup to fail:

```
LICENSE NOTICE:

c-treeRTG is licensed for # CPUs, but # CPUs have
been detected in the host machine. Either upgrade
the Server license to support a greater number
of CPUs or bind the Server to specific CPUs using
the CPU_AFFINITY configuration keyword.
```

The c-tree server is limited to use up to a specific number of CPU cores. This limit is controlled by the license. The default license allows c-tree to use up to 8 CPU cores.

If the server machine has more cores, the above error message is shown and c-tree doesn't start. There are two actions that can be taken to resolve this issue:

- purchase a different license, that allows more CPU cores, or
- limit the number of cores used by the faircom command.

To limit the number of cores used by c-tree:

1. edit the *ctsrvr.cfg* configuration file
2. add the following entry

```
CPU_AFFINITY 0,1,2,3,4,5,6,7
```

Note - The above information is for limiting c-tree to 8 CPU cores. Use more or less numbers if you need to limit it to a different number of cores.

3. restart c-tree

On Linux it's also possible to skip the editing of the *ctsrvr.cfg* configuration file and use the *taskset* system command as follows:

```
taskset --cpu-list 1-8 ./faircom
```

If you want to start c-tree with *nohup* in the background, add the syntax after the *taskset* command. For example:

```
taskset --cpu-list 1-8 nohup ./faircom &
```

Note - The above information is for limiting c-tree to 8 CPU cores. Use different numbers if you need to limit it to a different number of cores.

On Solaris systems, *CPU_AFFINITY* accepts a single numeric value, which is interpreted as a processor set number. For example, *CPU_AFFINITY 2* configures the c-tree server to run on processor set 2.

To create a processor set on Solaris, use the Solaris command *psrset*. For example, to create a set comprising processors 1 through 8, use:

```
psrset -c 1-8
```

The ID of the newly created processor set is returned.

To bind a process to this processor set, use:

```
psrset -b <ID> <pid>
```

Note - you must have root permissions to run the above commands.

Problems working with the c-tree Server Windows Service

Problems starting the c-tree Server Service

If the c-tree Server Service fails to start, check the log file *CTSTATUS.FCS* generated in the c-tree Server's local directory. By default *CTSTATUS.FCS* is generated in the *data* subfolder. A possible startup failure condition is the lack of a valid license.

If *CTSTATUS.FCS* doesn't include any useful error message, then check the Windows Event Log.

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. If the c-tree Server Service is running, check its status log file (*CTSTATUS.FCS*) 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](#) utility is also a useful tool for verifying whether clients can connect to the c-tree Server Service

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).

Faircom's [ctadmn](#) utility can also be used to stop the c-tree Server Service.

Problems connecting from a COBOL program

If the COBOL program cannot connect to the c-tree Server correctly, the following error is returned:

```
Internal error: java.lang.IllegalArgumentException: ct_init ERROR 19:133:0
```

There are three possible causes for this error:

- the c-tree Server is not active
- the c-tree Server cannot be reached due to network problems or firewalls
- the version of the client library doesn't match with the version of the server

Additional information may be available in the c-tree client side log file.

To activate this log using isCOBOL configuration properties, set:

```
iscobol.file.index.log.file=/path/to/ctree.log  
iscobol.file.index.log.debug=true  
iscobol.file.index.log.error=true  
iscobol.file.index.log.info=true  
iscobol.file.index.log.profile=true  
iscobol.file.index.log.warning=true
```

To activate this log in the ctree.conf configuration file, add the `<log>` entry as follows:

```
<config>  
  <log file="/path/to/ctree.log">  
    <debug>yes</debug>  
    <error>yes</error>  
    <info>yes</info>  
    <profile>yes</profile>  
    <warning>yes</warning>  
  </log>  
  ...
```

Error Codes

The following table lists the most common error codes that are applicable to c-tree and provides a brief description and some advice.

These codes can appear:

- as secondary code of [9D](#) error in the COBOL program
- in the c-tree client log file (see [<log>](#) and [iscobol.file.index.log.file](#) for details)
- as output of [ctutil](#) commands

Logical Error	Meaning
-3	The client library (ctree.dll on Windows, libctree.so on Linux) is not compatible with the c-tree Server. This error is usually returned when there is a mismatch in the major versions of these components, e.g. a ctree.dll v10.4 is trying to connect to a c-tree server v3.0.0.
36	The most common cause of this error is that you're opening a file that is not a c-tree file. Jlsam files, for example, have the same default extensions for the data portion (.dat) and the index portion (.idx), so a c-tree file and a Jlsam file could be easily confused for one another.
133	The c-tree server is down or unreachable. Double check network connectivity (e.g. try to ping the server where c-tree is running) and ensure that no firewall software is blocking the communication on the ports used by c-tree (by default: 5597 and 6597) on the server.
401	The file is corrupt. Try to repair it using the -rebuild option. If the error persists, contact the Veryant's support team.
456	A common cause of this error is that the ADMIN credentials were passed on the ctutil command line instead of being passed through a configuration file. See Registering existing c-tree files in c-tree SQL Server for more details.
978	A shared memory connection could not be established because the system denied access to the client process. For example, if a client is run as a Windows service and c-tree Server is not run as a Windows service (or vice-versa),

If you encounter an error that is not in the above list, you can use the [ErrorViewer](#) utility to obtain a brief description of the error. If the information returned by the utility doesn't help in understanding the error cause and resolution, contact the Veryant's support team.

Core Dumps

In the rare case of a crash of the c-tree Server executable, you might look for core dumps in the c-tree Server's directory. These files could be useful for support technicians to understand the cause of the crash, so collect them before contacting the technical support.

Windows

On Windows, look for files with mdmp extension, i.e.

```
stack1036_01.mdmp
```

Linux / Unix

On Linux / Unix systems, by default a core dump file is named "core", but the system can be configured to define a template that is used to name core dump files. The creation of core dumps might also be disabled in the system in order to save disc space.

Refer to your Linux distro documentation for information about how to enable and configure the creation of core dumps.

Chapter 11

Drivers

The c-tree installation includes several drivers to interact with external software:

- an [ODBC Driver](#)
- a [JDBC Driver](#)
- [Other Drivers](#)

These drivers interface the SQL engine of c-tree and allows you to work on database tables using SQL statements. Database tables include:

- tables created directly with a CREATE TABLE statement
- ISAM archives sqlized with the [ctutil -sqlize](#) command
- ISAM archives created with the [iscobol.sqlserver.iss \(boolean\)](#) property set to true

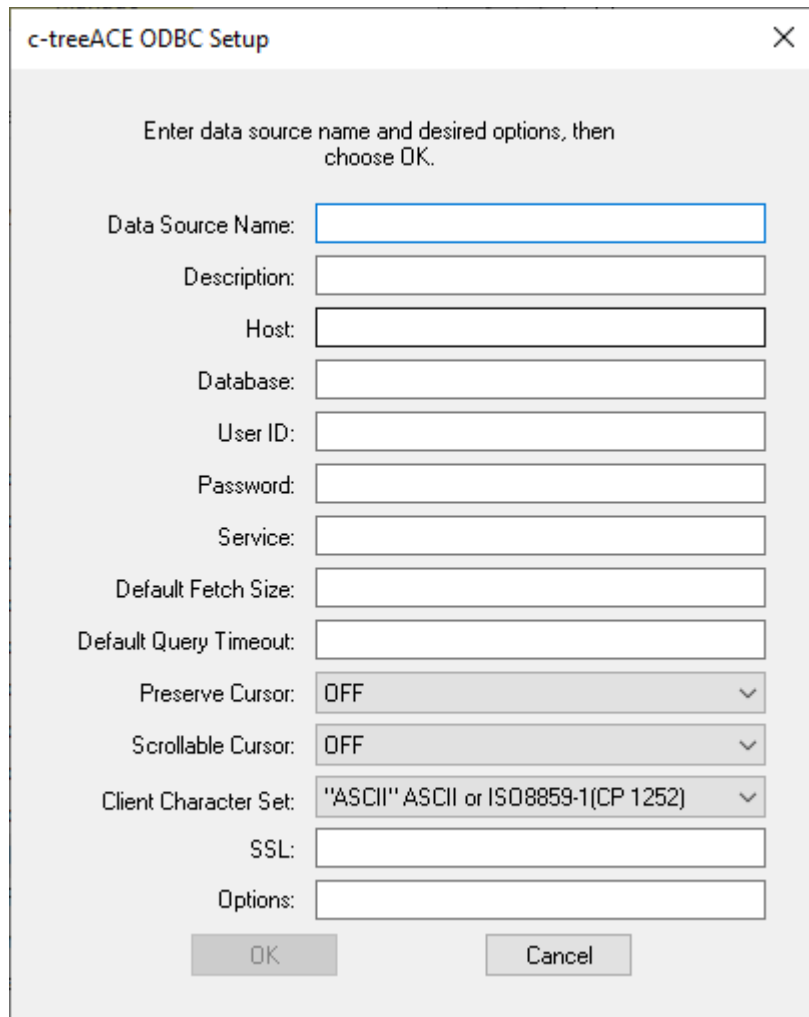
ODBC Driver

The ODBC Driver is installed on Windows machines along with c-tree if the proper option is checked. See the [Installing on Windows](#) chapter for details.

After the installation, you can create a DSN by following these steps:

1. Open Control Panel, Administrative Tools, ODBC Data Sources
2. Switch between User and System pages depending on the type of DSN you wish to create.
3. Click on ADD button
4. Choose "FairCom ODBC Driver" from the list of drivers

The following panel will appear:



The image shows a Windows-style dialog box titled "c-treeACE ODBC Setup". It contains a close button (X) in the top right corner. Below the title bar, there is a text instruction: "Enter data source name and desired options, then choose OK." The dialog features several input fields and dropdown menus arranged vertically: "Data Source Name:" (a text box), "Description:" (a text box), "Host:" (a text box), "Database:" (a text box), "User ID:" (a text box), "Password:" (a text box), "Service:" (a text box), "Default Fetch Size:" (a text box), "Default Query Timeout:" (a text box), "Preserve Cursor:" (a dropdown menu currently showing "OFF"), "Scrollable Cursor:" (a dropdown menu currently showing "OFF"), "Client Character Set:" (a dropdown menu currently showing "ASCII" ASCII or ISO8859-1(CP 1252)"), "SSL:" (a text box), and "Options:" (a text box). At the bottom of the dialog are two buttons: "OK" and "Cancel".

Fill it in as follows:

- *Data Source Name*: name of the DSN. It's the name that will be used by ODBC clients to reach the datasource.
- *Description*: optional commentary info for the DSN.
- *Host*: IP address or server name of the pc in which c-tree Server is started.
- *Database*: name of the database (i.e. "ctreeSQL").
- *User ID*: name of the user (i.e. "admin").
- *Password*: password of the user (i.e. "ADMIN")

Remaining fields should not be modified.

On other platforms, the ODBC Driver is provided as a shared library that you can link to your ODBC client program. The library is stored under *drivers/sql.odbc* in the c-tree directory.

Consult Faircom's documentation for additional information.

JDBC Driver

The JDBC Driver is installed on Windows machines along with c-tree if the proper option is checked. See the [Installing on Windows](#) chapter for details.

On other platforms the driver library is installed under *drivers/sql.jdbc* in the c-tree directory.

The driver library is named "ctreeJDBC.jar" and must be available in the Java Classpath.

The following code snippet shows how to connect to c-tree SQL from a Java program:

```
Class.forName ("ctree.jdbc.ctreeDriver");  
Connection conn = DriverManager.getConnection ("jdbc:ctree:6597@localhost:ctreeSQL",  
"ADMIN", "ADMIN");
```

Consult Faircom's documentation for additional information.

Other Drivers

c-tree provides the following additional drivers:

- ADO.NET
- C#
- C/C++
- COBOL drivers
- IOT
- ISQL
- Javascript
- Java
- Node-RED
- PHP
- Python
- SQL Stored Procedures
- VB

These drivers are installed on Windows machines along with c-tree if the proper option is checked. See the [Installing on Windows](#) chapter for details.

Drivers are installed in the *drivers* sub directory of the c-tree installation. Each driver is installed in a separate folder. The folder includes also a couple of examples and a README.

Consult Faircom's documentation for additional information.

Chapter 12

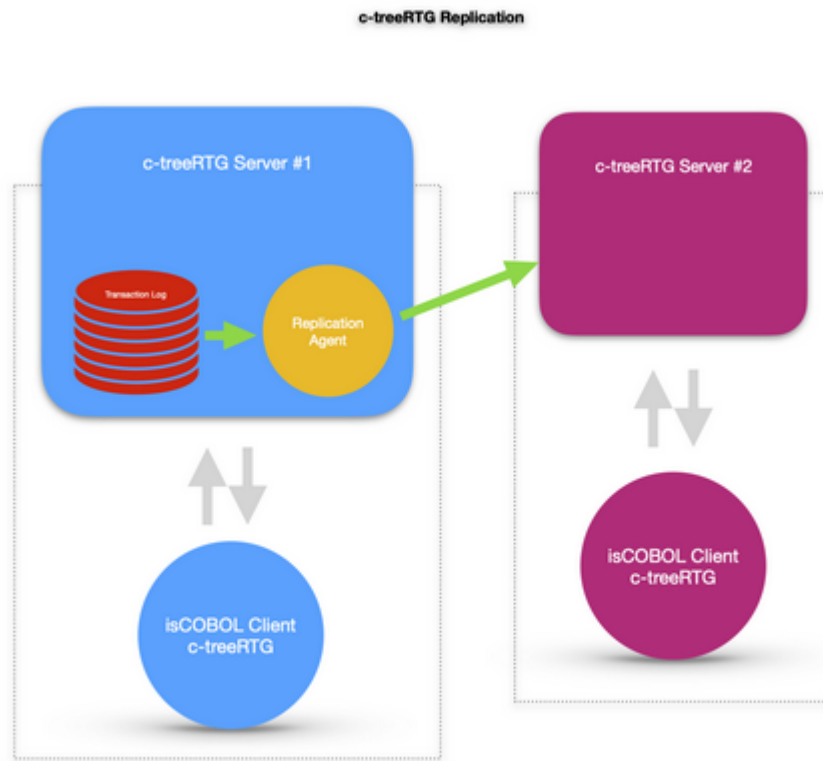
Data Replication

Overview

The c-tree server includes a Replication Agent, a component acting as a conduit to pass data from transaction log entries on a master server to a local server. The Replication Agent is responsible for establishing connections to both c-tree database engines, maintaining a current position in case of connection failures, and logging exceptional transactions that cannot be reliably replicated.

The Replication Agent provides the ability to replicate data from one c-tree database engine instance to another.

The Replication Agent can be activated on the same machine as the production database engine or on the backup copy of the database engine. Best practice is generally to run the agent on the same machine as the backup database engine for optimal throughput.



Installation and licensing

The Replication Agent is part of the c-tree Server engine, but it requires a specific license in order to be activated.

For a successful data replication, the involved c-tree servers must have data replication features enabled in their license. The default license installed with c-treeRTG doesn't enable data replication features.

Contact your Veryant representative to request a proper license.

Usage and configuration

In this chapter we're going to setup an asynchronous data replication between two c-tree servers.

Transaction logging required

The c-tree replication feature relies on a transaction log based facility. Your files must be under transaction control to take advantage of c-tree replication.

You can enable transaction logging in the configuration by setting `<transaction>` in `ctree.conf` or `iscobol.file.index.transaction.logging (boolean)` among isCOBOL properties. This will affect new files.

Transaction logging can be applied to existing files using the `-tron` option of the `ctutil` utility.

Shutdown the c-tree servers

It's good practice to stop the involved c-tree servers before proceeding with the steps discussed below.

Start the Replication Manager (Memphis)

To start the Faircom's Replication Manager, change to the folder "replication.manager/replicationmanager" and run the command:

```
memphis
```

After a successful startup, the Faircom's Replication Manager listens for standard HTTP connections on the port 8080 and for secure HTTP connections on the port 8443.

Note - The default ports can be changed by editing the *cthhttpd.json* file under the "replication.manager/replicationmanager/config" folder.

Configure the c-tree servers

The involved c-tree servers must have different names and have the necessary plugins enabled.

The following actions should be performed on the involved c-tree servers:

1. Edit *config/ctsrvr.cfg*:
 - a. be sure that the SERVER_NAME entry in this configuration file uses a different value than the SERVER_NAME entry in the other configuration files. For example, you could name the first server FAIRCOMS1 and the second server FAIRCOMS2;
 - b. activate the ctagent plugin and the httpd plugin by removing the leading semicolon that comments the lines out:

Windows

```
PLUGIN cthttpd;./web/cthhttpd.dll  
PLUGIN ctagent;./agent/ctagent.dll
```

Linux/Unix

```
PLUGIN cthttpd;./web/libcthhttpd.so  
PLUGIN ctagent;./agent/libctagent.so
```

2. replace the content of *replication.manager/replicationmanager/config/ctagent.json* with this code:

```
{  
  "embedded": true,  
  "log_file": "agentLog.txt",  
  "memphis_server_name": "MEMPHIS",  
  "memphis_sql_port": 7000,  
  "memphis_host": "localhost",  
  "memphis_database": "MEMPHIS",  
  "ctree_check_mask": "*.dat;*.idx;*.fdd;*.fsd",  
  "inactive_timeout": 600  
}
```

Note - "memphis_host" has to be set with the appropriate IP address of the machine where the Memphis is running.

3. replace the content of *replication.manager/replicationmanager/config/cthttpd.json* with this code:

Windows

```
{
  "listening_http_port": 8080,
  "listening_https_port": 8443,
  "ssl_certificate": "./web/fccert.pem",
  "document_root": "./web/apps",
  "applications": [
    "ctree;ctREST.dll",
    "AceMonitor;ctMonitor.dll",
    "SQLExplorer;ctSQLExplorer.dll",
    "ISAMExplorer;ctISAMExplorer.dll",
    "ReplicationManager;ctReplicationManager.dll"
  ]
}
```

Linux/Unix

```
{
  "listening_http_port": 8080,
  "listening_https_port": 8443,
  "ssl_certificate": "./web/fccert.pem",
  "document_root": "./web/apps",
  "applications": [
    "ctree;libctrest.so",
    "AceMonitor;libctmonitor.so",
    "SQLExplorer;libctsqlexplorer.so",
    "ISAMExplorer;libctisamexplorer.so",
    "ReplicationManager;libctReplicationManager.so"
  ]
}
```

Start the c-tree servers

After being configured, the c-tree servers can be started.

Use the Replication Manager web utility to configure and monitor the replication

Using a web-browser, browse to <http://localhost:8080/ReplicationManager>.

Use the IP address of the server where Memphis is running instead of "localhost" if you're connecting from a remote machine.

An overview of available c-tree servers is presented:



To create a replication between two servers, hover over the edge of the source server: a chain-link appears. Drag from the source server and drop on the target server:



The following dialog appears:

Replication Plan : **ReplPlan1** - **ReplPlan1 Description**

Basic Options Extension Library Advanced Options Replication Latencies

Name: Direction:

Description: Synchronous: ☐

Source Server (FAIRCOMS@LAPTOP-DG0VLRP3) -> Target Server (FAIRCOMS@ubuntu)

Available Publications on Source Server				Subscriptions on Target Server								
Name	Description	Edit	Subscription	Pub. Name	Pub. Description	Pub.	Subscription	Sub.				
<div>Create New Publication</div>												

Target Server (FAIRCOMS@ubuntu) -> Source Server (FAIRCOMS@LAPTOP-DG0VLRP3)

Publication Details : (click a publication row for details)

Filename	Database	Volume	Path

Close

Click *Create New Publication* and use the Replication Wizard to create a Publication.
The Publication includes the folders and files on the source server to be replicated.

Publications for FairCom Server : FAIRCOMS@LAPTOP-DG0VLRP3

Step 1 - Enter Basic Info and Choose Publication Type

Publication Name

Name:

Description:

Publication Type

Mode: ☐ By Database Folders ☒ By Folder ☐ By Files

Tips:

- A **File Publication** replicates specific data files.
- A **Folder Publication** replicates data files in a folder. When files are later added to the folder, they are automatically replicated. You can also create filters to include or exclude files from replication.
- A **Database Folders Publication** replicates a database's data dictionary and the data files in its folders. When files are later added to database folders, they are automatically replicated – if not excluded by your filters. You can exclude all folders from a Database Folder Publication to replicate only its data dictionary and you can use other File and Folder Publications to replicate a specific files.
- A single **Replication Plan** can include many subscriptions to all types of publications. Thus, you can make replication scenarios as simple or as sophisticated as you desire.

« Previous **Next** »

Publications for FairCom Server : FAIRCOMS@LAPTOP-DG0VLRP3

Step 2 of 5: Select a Folder

Basic Info

Name: Publication1
Description: Publication1
Publication Type: Folder

Select Files From: C: ☒ Include everything in selected folder ☐ Exclude everything in selected folder
Tip: the next panels allow you to define file, wildcard and regular expression exceptions.

Select a Single Folder Replicable ☐

- [-] spellings
- [-] tenorshare
- [-] tmp
 - [-] VisualStudio
 - [-] customers
 - [-] data**
 - [-] exthf_example.tar

Published Folders

Name	Vol.	Path
data	C:	tmp

Total Selected Folders:

< Previous Next > Finish

You can optionally exclude specific files:

Publications for FairCom Server : FAIRCOMS@LAPTOP-DG0VLRP3

Step 3 of 5: Exclude Specific Files from Folder Replication (optional)

Tips: All files in the selected folders are replicated unless you exclude them here.
The next pane allows you to use wildcards and regular expressions to exclude files.

Basic Info

Name: Publication1
Description: Publication1
Publication Type: Folder

Select Files From: data Scan Options : ☐ c-tree Files Only File Mask: *.*

Select Files Replicable

tmp	
data	
Customer.dat	✓
Customer.idx	

Excluded Files

Name	Path	Excluded

Total Excluded Files: 0

< Previous Next > Finish

In the end, a summary is shown:

Publications for FairCom Server : FAIRCOMS@LAPTOP-DG0VLRP3

Step 5 of 5: Folder Publication Summary

Basic Info

Name: Publication1
Description: Publication1
Publication Type: Folder

Published Folders

Folder
tmp/data

Excluded Files

Folder	File Names
--------	------------

Wildcard and Regex File Exclusions

Folder	Wildcard or Regex
--------	-------------------

< Previous Next >

Finish

Click *Finish* to confirm.

The Publication is now available:

Replication Plan : **ReplPlan1** - ReplPlan1 Description

Basic Options Extension Library Advanced Options Replication Latencies

Name: Direction:

Description: Synchronous: ☐

Source Server (FAIRCOMS@LAPTOP-DG0VLRP3) -> Target Server (FAIRCOMS@ubuntu)

Available Publications on Source Server				Subscriptions on Target Server				
Name	Description	Edit	Subscription	Pub. Name	Pub. Description	Pub.	Subscription	Sub.
Publication1	Publication1	Edit	Subscribe					

Create New Publication

Target Server (FAIRCOMS@ubuntu) -> Source Server (FAIRCOMS@LAPTOP-DG0VLRP3)

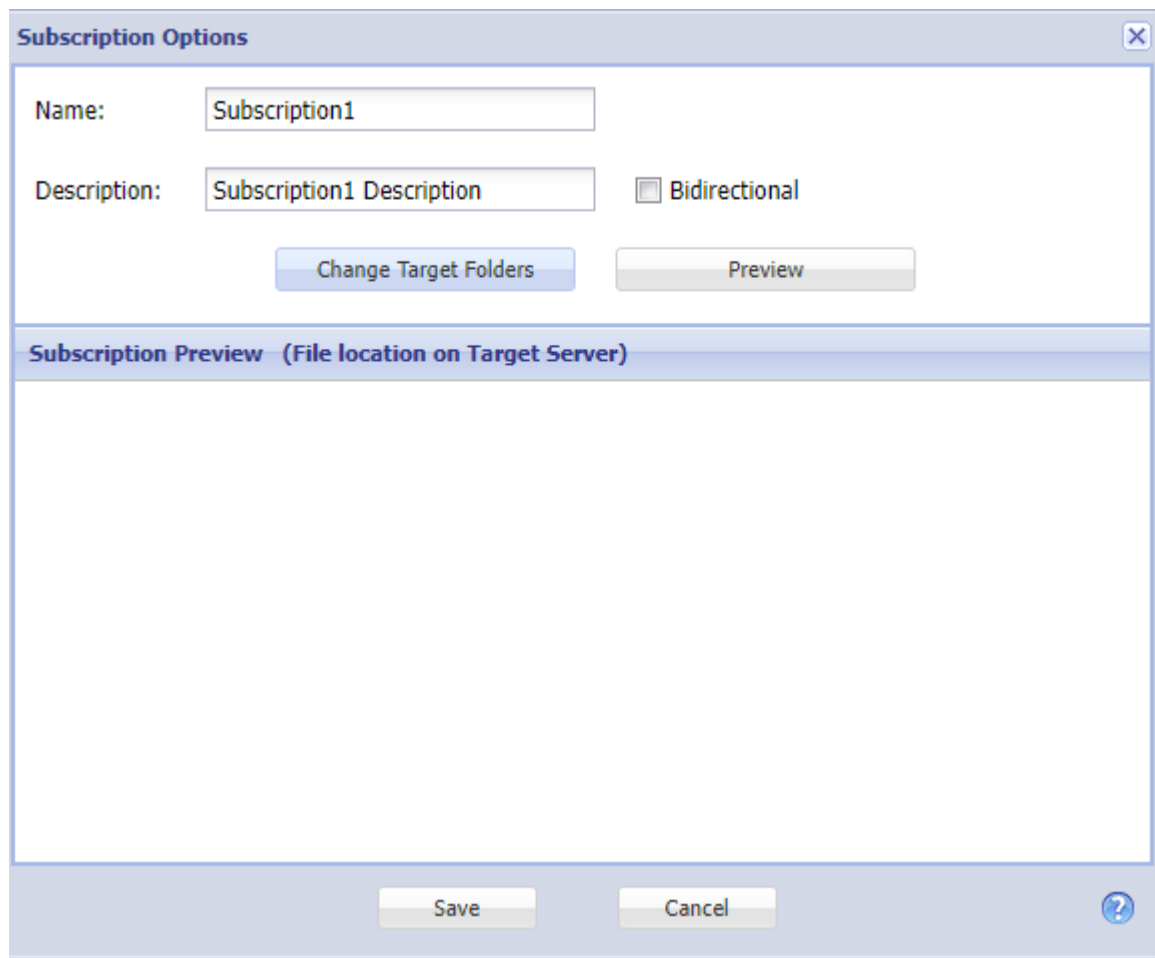
Publication Details : (click a publication row for details)

Filename	Database	Volume	Path

Close

Click *Subscribe* to make a Subscription.

The Subscription is the replication destination on the target server.



The image shows a Windows-style dialog box titled "Subscription Options". It has a standard title bar with a close button (X) in the top right corner. The main area contains two text input fields: "Name:" with the value "Subscription1" and "Description:" with the value "Subscription1 Description". To the right of the description field is a checkbox labeled "Bidirectional", which is currently unchecked. Below these fields are two buttons: "Change Target Folders" (highlighted in blue) and "Preview" (disabled, shown in grey). A horizontal separator line divides the dialog into two sections. The bottom section is titled "Subscription Preview (File location on Target Server)" and contains a large, empty rectangular area for a preview. At the bottom of the dialog are two buttons: "Save" (disabled, grey) and "Cancel" (disabled, grey). A help icon (question mark in a circle) is located in the bottom right corner.

Subscription Options

Name: Subscription1

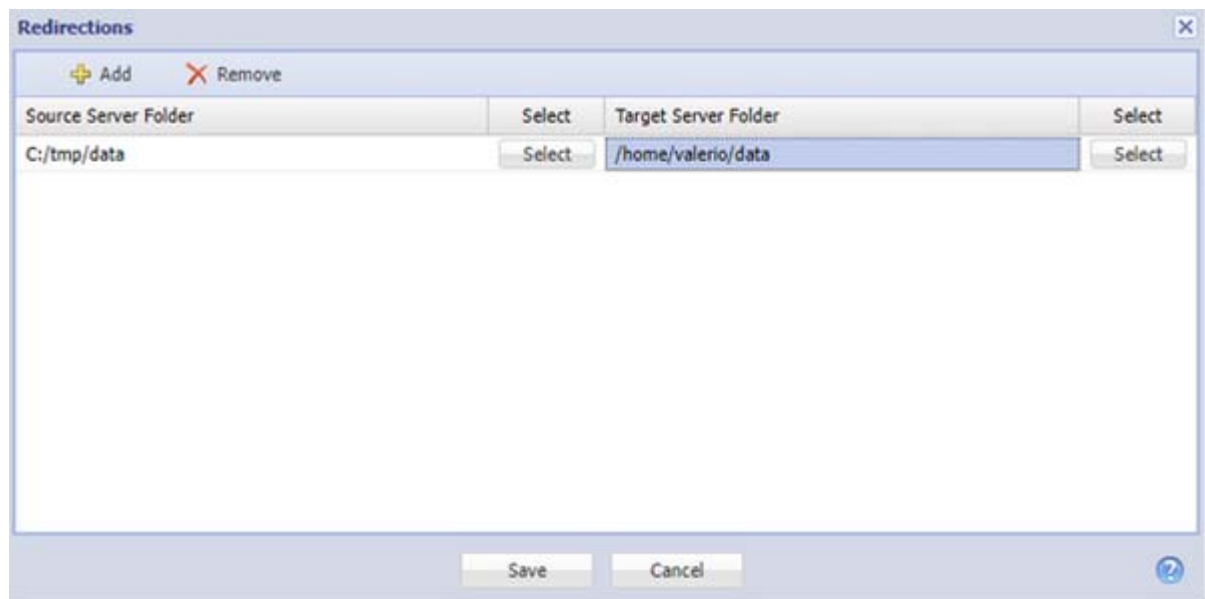
Description: Subscription1 Description ☐ Bidirectional

Change Target Folders Preview

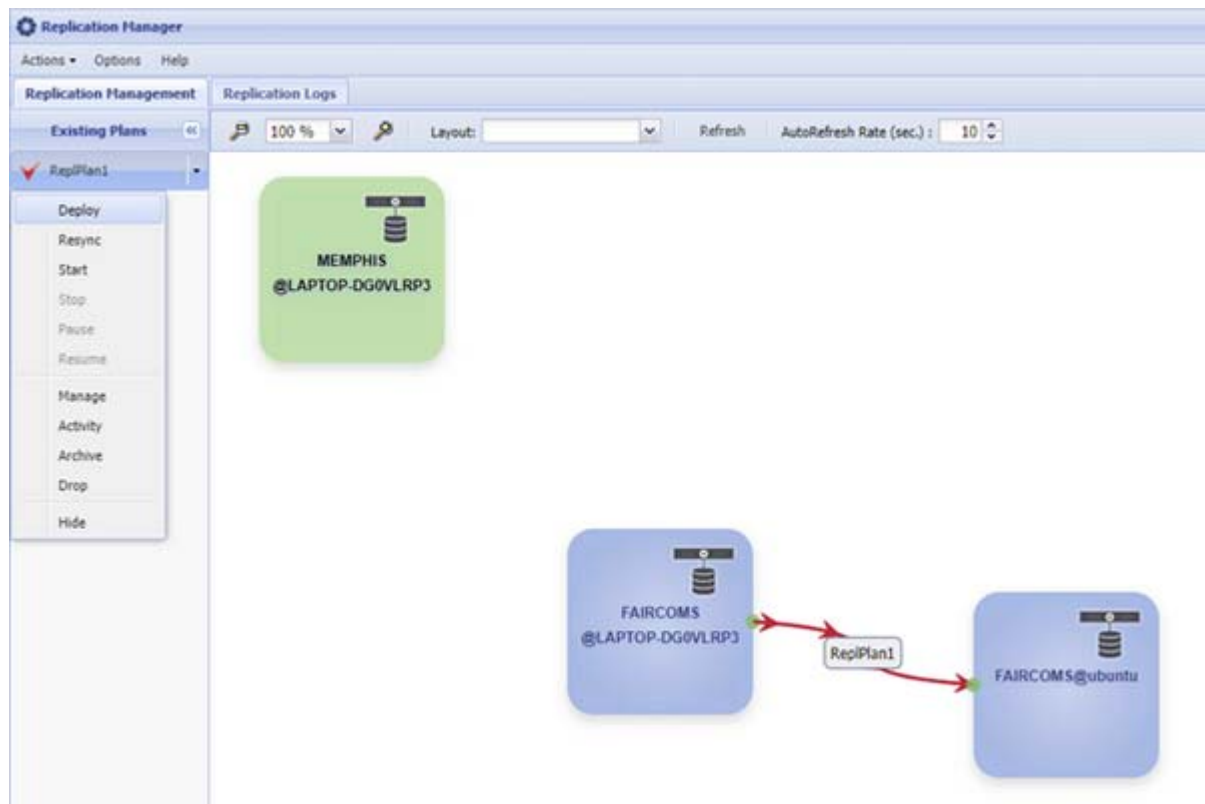
Subscription Preview (File location on Target Server)

Save Cancel ?

The Subscription Options dialog allows you to change the destination folder of the Target server and setup a Bidirectional replication.

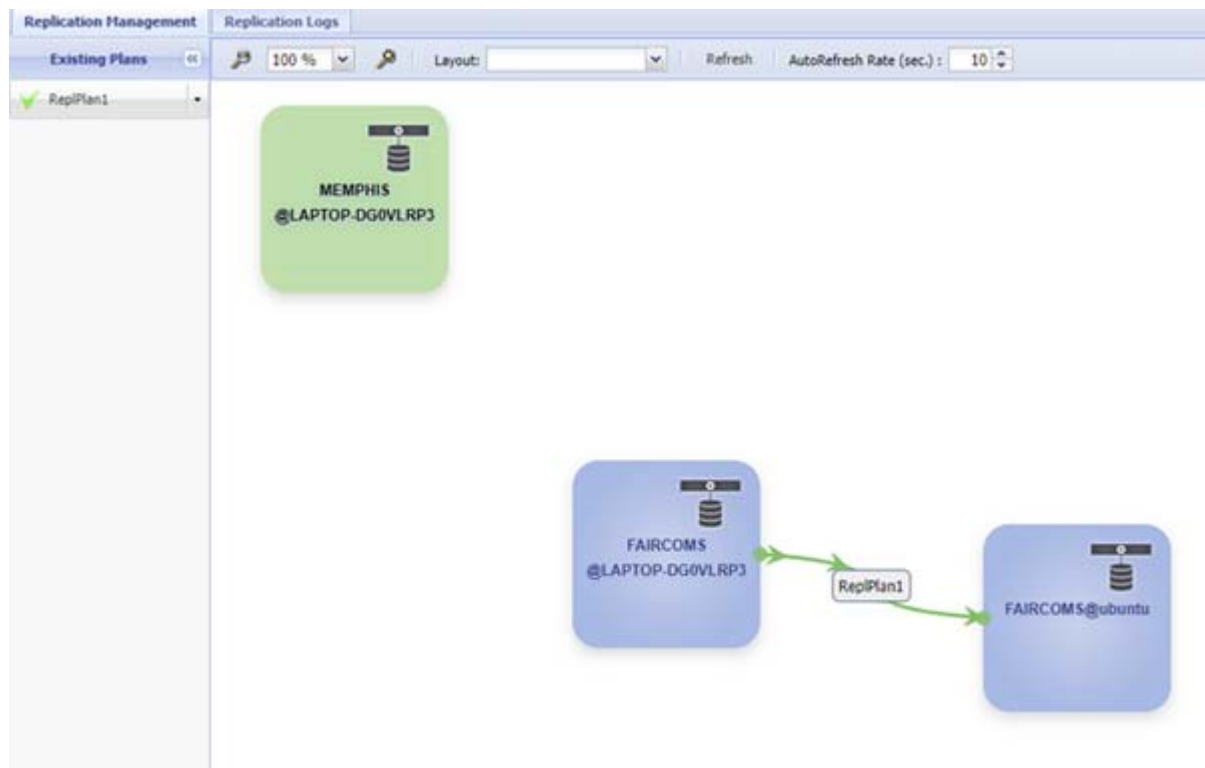


The replication plan is now available and can be managed in the "Existing Plans" section of the Replication Manager:



Click the arrow button next to a plan and click *Deploy* to deploy it, then click *Start* to begin replication.

The connection between the c-treeRTG Servers becomes green when the deploy task completes successfully:



Click the arrow button next to a plan and click *Activity* to have a summary of the work that was performed. The summary looks like this:

Plan activity for plan : ReplPlan1

Options

Time Frame: 1 Hour

Start Date: 12/16/2022

End Date: 12/16/2022

Start Time: 11:17 AM

End Time: 12:17 PM

Refresh

Auto Display Refresh Rate: 0 sec

Dump

Details

Statistics

Exceptions

Function Tuning

Page 1 of 1

Snapshot Status: Running

Stop

Collection Rate: 5

Displaying 1 - 28 of 28

	Identification		Replication				Thread Counts			Transaction Throughput			
	Type	Time	Paused	Aborted	Failures	Latency	Analyze	Apply	Commits	Add	Update	Delete	
20	sourceToTarget	12/16/2022 12:15:20 PM				0	1	8	3	2	0	1	
21	sourceToTarget	12/16/2022 12:15:30 PM				0	1	8	3	2	0	1	
22	sourceToTarget	12/16/2022 12:15:40 PM				0	1	8	3	2	0	1	
23	sourceToTarget	12/16/2022 12:15:51 PM				0	1	8	3	2	0	1	
24	sourceToTarget	12/16/2022 12:16:01 PM				0	1	8	3	2	0	1	
25	sourceToTarget	12/16/2022 12:16:11 PM				0	1	8	3	2	0	1	
26	sourceToTarget	12/16/2022 12:16:21 PM				0	1	8	3	2	0	1	
27	sourceToTarget	12/16/2022 12:16:31 PM				0	1	8	3	2	0	1	
28	sourceToTarget	12/16/2022 12:16:41 PM				0	1	8	3	2	0	1	

Close

Usage and configuration - Legacy method

In this chapter we're going to setup an asynchronous data replication between two c-tree servers without using the Replication Manager utility.

Transaction logging required

The c-tree replication feature relies on a transaction log based facility. Your files must be under transaction control to take advantage of c-tree replication.

You can enable transaction logging in the configuration by setting `<transaction>` in `ctree.conf` or `iscobol.file.index.transaction.logging (boolean)` among isCOBOL properties. This will affect new files.

Transaction logging can be applied to existing files using the `-tron` option of the `ctutil` utility.

Shutdown the c-tree servers

It's good practice to stop the involved c-tree servers before proceeding with the steps discussed below.

Prepare the Source

No particular operation is required on the Source c-tree server. Just be sure that the installed license allows replication features.

Prepare the Target

Be sure this second c-tree server resides in its own directory and has a `SERVER_NAME` different than the name of the Source c-tree server. For example, you can leave the default name "FAIRCOMS" in the Source c-tree server and use the name "FAIRCOMT" for the Target c-tree server.

1. Edit `config/ctsrvr.cfg`
 - a. provide a different `SERVER_NAME`, e.g. replace

```
SERVER_NAME FAIRCOMS
```

with

```
SERVER_NAME FAIRCOMT
```

- b. activate the `ctagent` plugin by removing the leading semicolon that comments the line out:

Windows

```
PLUGIN ctagent;./agent/ctagent.dll
```

Linux/Unix

```
PLUGIN ctagent;./agent/libctagent.so
```

2. replace the content of *config/ctagent.json* with this code:

```
{  
  "detached": true,  
  "configurationFileList": [  
    "../config/replication/ctreplagent1.cfg"  
  ]  
}
```

3. edit *config/replication/ctreplagent1.cfg*

- a. replace "SourceIP" with the source machine host name in the following line:

```
source_server FAIRCOMS@SourceIP
```

- b. replace "FAIRCOMS" with "FAIRCOMT" in the following line:

```
target_server FAIRCOMS@localhost
```

4. edit *config\replication\filter1.xml* and

- a. replace ".\ctreeSQL.dbs\admin_test.dat" with "*.dat" as follows:

```
<file status="include">*.dat</file>
```

- b. add a purpose for replicating modifications on existing files as follows:

```
<purpose>create_file</purpose>  
<purpose>open_file</purpose>  
<purpose>read_log</purpose>
```

5. edit *config\replication\source_auth.cfg* changing the PASSWD value to "ADMIN":

```
USERID ADMIN  
PASSWD ADMIN
```

6. edit *config\replication\target_auth.cfg* changing the PASSWD value to "ADMIN":

```
USERID ADMIN  
PASSWD ADMIN
```

7. open a command prompt

- a. change to the *config\replication* directory

- b. run these commands to generate the new *source_auth.set* and the new *target_auth.set*:

```
../../tools/ctcmdset source_auth.cfg  
../../tools/ctcmdset target_auth.cfg
```

8. copy existing c-tree files from the Source server to the Target server

(Optional) Switching from Asynchronous Replication to Synchronous Replication

Synchronous replication is mainly used for high-end transactional applications that need instant failover if the primary node fails and cannot tolerate data loss. As drawback, synchronous replication introduces latency that slows the primary application.

To make the replication synchronous

1. edit *config/replication/ctreplagent1.cfg* and activate the *syncagent* option by removing the leading semicolon that comments the line out:

```
syncagent yes
```

2. edit *config\replication\filter1.xml* and add

```
<purpose>sync_commit</purpose>
```

Testing the Replication Agent

After the above steps, start (or restart) the c-tree servers.

When both Source and Target c-tree servers are up and running, you can start editing the c-tree files on the Source server. Every modification will be replicated to the corresponding files on the Target server.

Troubleshooting

The most common errors that can happen during data replication are:

Error condition	Description
Error 133 at startup	The remote c-tree server is down or unreachable. Double check IP and port provided and ensure that a firewall is not blocking the connection
Error 979 at startup	The c-tree server license doesn't allow replication features. Contact your Veryant representative to purchase a proper license.

Error condition	Description
Errors 96 and 76 restarting after improper shutdown	<p>Error 96 indicates that a log could not be opened. If the Replication Agent had been using that log before it terminated, it should have recorded its next position offset.</p> <p>Error 76 indicates that the Replication Agent did not record a valid offset in the log. The log position did not correspond to the start of a transaction log entry: it was in the middle of an entry.</p> <p>An action taken to avoid the 96 error could cause error 76. Because the saved offset is specific to the log on the original source server, replacing the file could cause error 76. An example would be copying another file to replace the original log. Another possible cause is if you changed the agent to connect to a different source server than it was using before, if that server happens to have a log with the same name, you could get this error.</p>
Replication Agent Error 809	<p>It is possible to receive Replication Agent error 809 under certain conditions. This situation can occur when applications that use c-tree's socket timeout feature generate a lot of transaction log data for files that are not replicated. This activity can force the server to scan the logs up to a checkpoint, which can exceed the connection timeout. That triggers a disconnect, causing the Replication Agent to reconnect and continue.</p> <p>In this case, the timeout can be avoided by increasing the socket_timeout parameter (e.g., setting to 30 seconds or more). A setting of 300 seconds (5 minutes) should be a reasonable timeout value.</p>
Error 537 - TCOL_ERR	<p>The Replication Agent maintains its own transaction logs and housekeeping files. Removal of these can cause problems. Be sure the FairCom Replication is run in its own unique directory independent of any existing servers. You can use the LOCAL_DIRECTORY keyword in the agent's ctsrvr.cfg file to position these files, just as you would with a regular server configuration.</p>

More information

In this book you've been provided with basic information to start and test the data replication features of c-tree.

Refer to Faircom's documentation at <https://docs.faircom.com/doc/ctrepl/> for more information about the Replication Agent.

Chapter 13

External References

To retrieve further information about c-tree administration and troubleshooting, the following links can be visited:

c-tree Server Administrator's Guide	https://docs.faircom.com/doc/ctserver/
c-treeRTG Error Codes	https://docs.faircom.com/doc/error_codes/