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

Table of Contents

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.

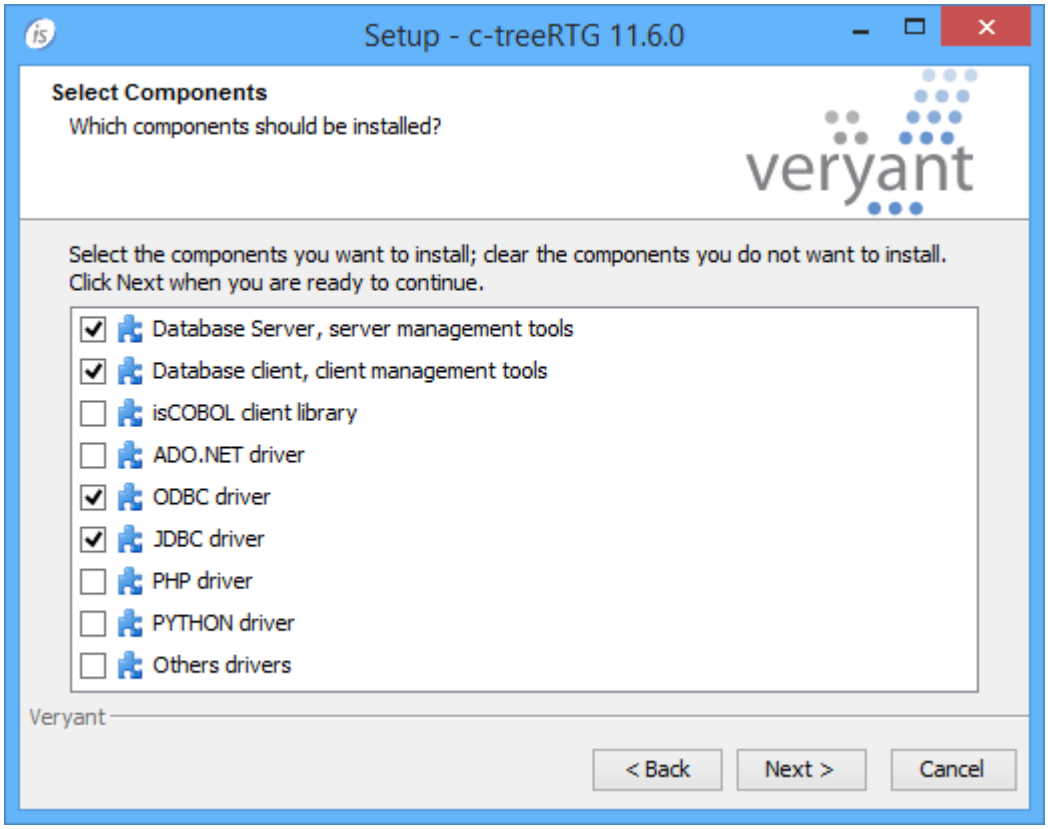
This kind of approach provides a fast and secure access to remote files and it's a strongly suggested alternative to shared directories.

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.

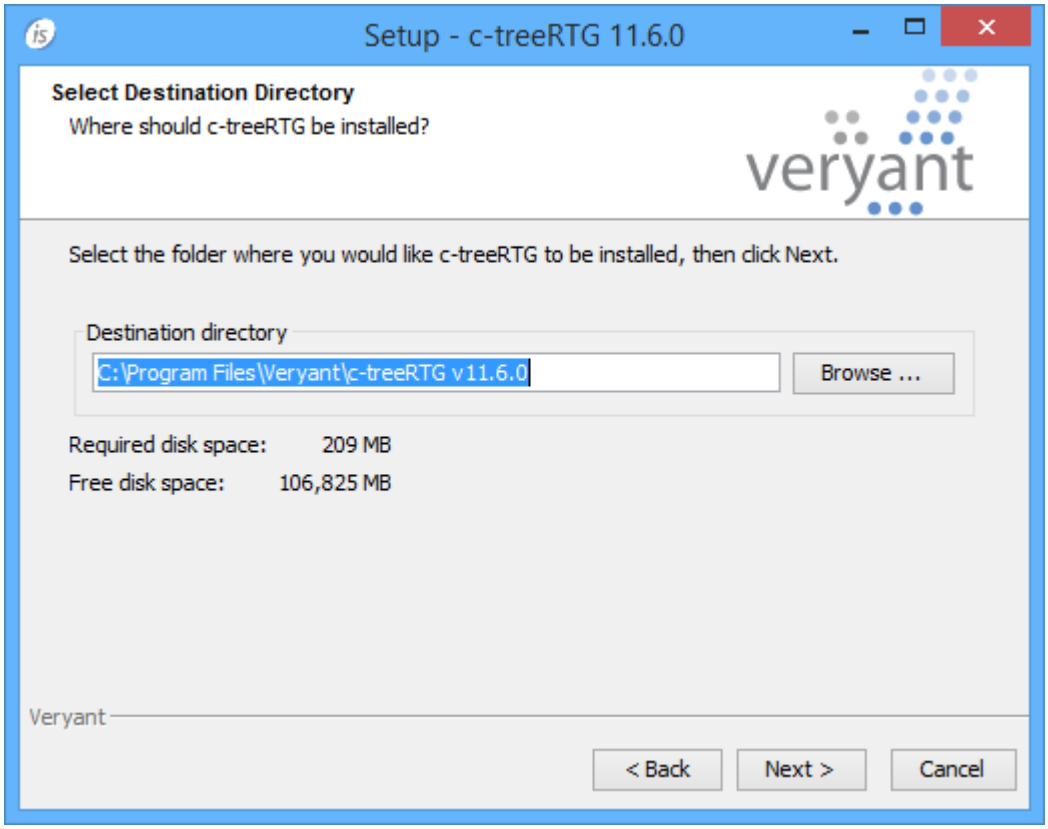
First of all, you have to choose which kind of installation to perform.



- *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.
- *isCOBOL client library* is the c-tree client installation. If this option is checked, only the client library (ctree.dll) will be installed.
- *ADO.NET driver*, if checked, installs the ADO driver for c-tree
- *ODBC driver*, if checked, installs the ODBC driver for c-tree
- *JDBC driver*, if checked, installs the JDBC driver for c-tree
- *PHP driver*, if checked, installs the PHP driver for c-tree
- *PYTHON driver*, if checked, installs the PYTHON driver for c-tree
- *Other drivers*, if checked, installs drivers to interface c-tree with other COBOLs

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 last step sets the path in which the c-tree and its utilities will be installed.



Installing on Unix

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

Current version of c-tree is:

- 11.6.0

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_v*Major.Minor* will be created.

Chapter 2

SQL Engine Licensing

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

- unlimited ISAM server features
- limited SQL server 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.

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

c-tree looks for a file called *ctsrvr.cfg* located in the same directory or the server executable. 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:

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

Example for Unix/Linux:

```
./ctreesql 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 the most commonly used configuration keywords. Refer to the c-tree Server Administrator's Guide for a complete list of keywords.

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	Specifies a communications module loaded by the Server. Possible values: F_TCPIP FSHAREMM DISABLE If no protocol is specified, then the default system communication protocol is used.

COMPATIBILITY TCPIP_CHECK_DEAD_CLIENTS	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>.
DAT_MEMORY	The memory allocated to the data cache, in bytes. 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: • 200 MB • 204800 KB • 209715200 The maximum value is 1 GB. The default value is 100 MB.
DELAYED_DURABILITY	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.
DEAD_CLIENT_INTERVAL <*seconds*>	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). COMPATIBILITY TCPIP_CHECK_DEAD_CLIENTS is required to enable the checking. Otherwise, no check is made.
FILES	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. 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. The default value is 1000.

IDX_MEMORY	The memory allocated to the index cache, bytes. 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: • 200 MB • 204800 KB • 209715200 The maximum value is 1 GB. The default value is 100 MB.
MAX_DAT_KEY	Maximum number of indices per data file. The default value is 64.
MAX_KEY_SEG	Maximum number of key segments allowed per index. The default value is 16.
MAX_FILES_PER_USER	The maximum number of data files and indices (where each index counts toward this total) per client connection. The default value is 2048.
SERVER_NAME	A name assigned to c-tree Server, instead of the default name. The default value is FAIRCOMS.
SERVER_PORT	A port assigned to c-tree Server, instead of the default port. The default value is 5597.
SERVER_DIRECTORY	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.
LOCAL_DIRECTORY	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.
CONSOLE NO_SHUTDOWN_PROMPT	If present, ctsrvr will not prompt for administrator password when closed (Windows only).
CONSOLE TOOL_TRAY	If present, ctsrvr will start minimized in the system tray (Windows only).
SQL_DATABASE	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.

Chapter 4

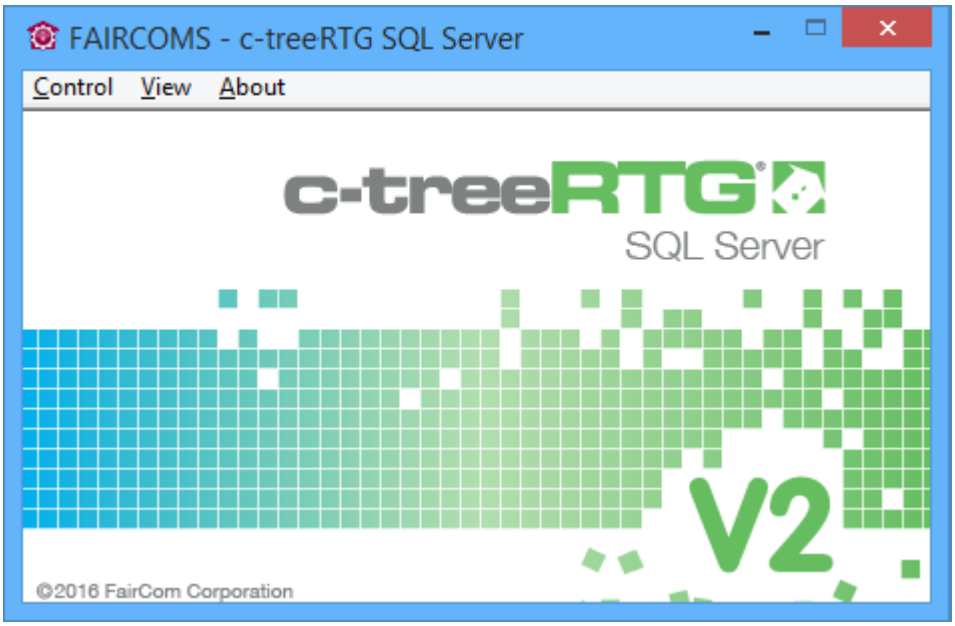
Server Startup

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-treeRTGv11.6.0* > *c-treeACE Server*

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



Otherwise, 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 *ctreesql.exe* file installed in the c-treeRTG "Server\sql" folder. It is the same executable file pointed by the link mentioned above. The ctreesql executable file supports command-line parameters allowing to specify configuration options or a custom configuration file directly on the command-line; see ctreesql command-line for details.

Windows Service

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

To install the service, start an MS-DOS prompt with Administrator privileges and run:

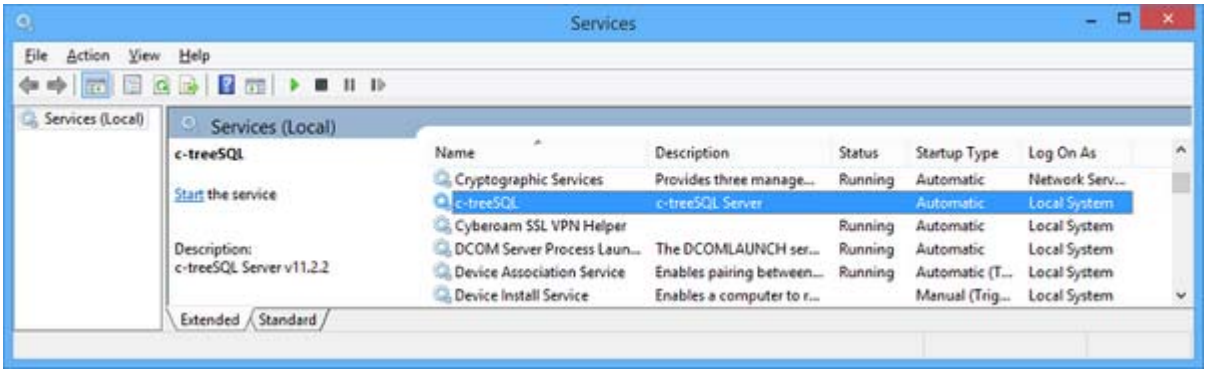
```
sc create c-treeSQL start= auto binPath= "C:\Program Files\Veryant\c-treeRTG
v11.6.0\Server\sql\ctreesql.exe"
```

Note - "c-treeSQL" is just a suggested name, but any name can be used. The ctreesql.exe path 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:\Program Files\Veryant\c-treeRTG
v11.6.0\Server\sql\ctreesql.exe  CTSRVR_CFG \path\to\ctsrvr1.cfg"

sc create c-treeSQL-2 start= auto binPath= "C:\Program Files\Veryant\c-treeRTG
v11.6.0\Server\sql\ctreesql.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:

```
./ctreesql
```

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

```
./ctreesql &
```

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 ctreesql &
```

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

See ctreesql command-line for information about the available command-line options.

ctreesql command-line

ctreesql 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:

```
ctreesql 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:

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

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

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:

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

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

Property	Description
`iscobol.file.index.log.debug.config.full (boolean)`	**True** = trace complete config details, including the default configuration. **False** = do not trace complete config details. If omitted, *False* is assumed.
`iscobol.file.index.log.debug.prefetch (boolean)`	**True** = trace prefetch details. **False** = do not trace prefetch details. If omitted, *False* is assumed.
`iscobol.file.index.log.error (boolean)`	**True** = trace errors in the log file. **False** = do not trace errors in the log file. If omitted, *True* is assumed.
`iscobol.file.index.log.error.atend (boolean)`	**True** = trace EOF errors. **False** = do not trace EOF errors. If omitted, *False* is assumed.
`iscobol.file.index.log.error.notfound (boolean)`	**True** = trace NOTFOUND errors. **False** = do not trace NOTFOUND errors. If omitted, *False* is assumed.
`iscobol.file.index.log.file`	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 iscobol.file.index.log properties to true or false if you wish to disable or enable a particular log. If omitted, no log is created.
`iscobol.file.index.log.info (boolean)`	**True** = trace generic information in the log file. **False** = do not trace generic information in the log file. If omitted, *True* is assumed.
`iscobol.file.index.log.profile (boolean)`	**True** = trace profile information in the log file. **False** = do not trace profile information in the log file. If omitted, *False* is assumed.
`iscobol.file.index.log.warning (boolean)`	**True** = trace warning information in the log file. **False** = do not trace warning information in the log file. If omitted, *True* is assumed.

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
`iscobol.file.index.anyunlock (boolean)`	**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. **False** = The lock is removed only if the UNLOCK is performed on the same OPEN instance where the READ WITH LOCK was performed. This property can be set only if iscobol.file.index.autolock_allowed (boolean) is set to true. If omitted, *False* is assumed.
`iscobol.file.index.autolock_allowed (boolean)`	**True** = More locks on the same record can be acquired in the same run unit. **False** = Only one lock on the same record can be acquired in the same run unit. If omitted, *False* is assumed.
`iscobol.file.index.batchaddition (boolean)`	**True** = write operations are buffered in order to speed up performance. **False** = write operations are not buffered. If omitted, *False* is assumed.
`iscobol.file.index.batchaddition.records`	Specifies the number of write operations that must be buffered at a time. If omitted, *10* is used.
`iscobol.file.index.bulkaddition (boolean)`	**True** = files open for output are managed with bulk addition (indexes are written at the close of the file) to speed up performance. **False** = files open for output are managed without bulk addition. If omitted, *False* is assumed.
`iscobol.file.index.connect (boolean)`	**True** = connect to c-tree server during runtime initialization. **False** = connect to c-tree server at the first open of a c-tree file. If omitted, *False* is assumed.
`iscobol.file.index.data_suffix`	Specifies the extension for the data file. If omitted, then *.dat* is assumed.
`iscobol.file.index.datacompress (boolean)`	**True** = data records are compressed using zlib algorithm. **False** = data records are not compressed. If omitted, *False* is assumed.

Property	Description
`iscobol.file.index.datacompress.level`	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. If omitted, then *1* is assumed.
`iscobol.file.index.datacompress.strategy`	This property specifies the strategy for data compression. Possible values are: **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. If omitted, *0* is assumed.
`iscobol.file.index.datacompress.type`	Specifies the algorithm for data compression. Valid values are: **zlib** = zlib algorithm is used. **rle** = rle algorithm is used. If omitted, then *rle* is assumed.
`iscobol.file.index.encrypt (boolean)`	**True** = files are encrypted using ctCAMO encryption algorithm. **False** = files are not encrypted. If omitted, *False* is assumed.
`iscobol.file.index.file_mapping`	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.
`iscobol.file.index.file_mapping_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. At the end of each file name you can put an ordinal number that is the number of the path between iscobol.file.index_path_mapping paths. In this way it's possible to specify a file mapping for each path maggping. Example: iscobol.file.index.1.path_mapping=/dir1,/dir2 iscobol.file.index.1.file_mapping_p=customers1,invoices1,reports2,discounts 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. See Configuring multiple instances for details.
`iscobol.file.index.fixed_length (boolean)`	**True** = files are created with fixed length records. **False** = files are created with variable length records. If omitted, *False* is assumed.

Property	Description
`iscobol.file.index.forcedelete (boolean)`	**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. If omitted, *False* is assumed.
`iscobol.file.index.index_suffix`	Specifies the extension for the index file. If omitted, then *.idx* is assumed.
`iscobol.file.index.inpool`	**True** = files are included in a pool to speed up the next OPEN operations. **False** = files are not included in a pool. If omitted, *False* is assumed. **Note** - the iscobol.file.index.filepool global setting must be set to true, otherwise this property will be ignored.
`iscobol.file.index.keycheck (boolean)`	**True** = extra keys in the file are not allowed. **False** = extra keys in the file are allowed. If omitted, *True* is assumed.
`iscobol.file.index.keycompress (boolean)`	**True** = index files are compressed using zlib algorithm. Leading and padding compression are applied. **False** = index files are not compressed. If omitted, *False* is assumed.
`iscobol.file.index.keycompress.leading (boolean)`	**True** = leading compression is applied on index files. **False** = leading compression is not applied on index files. If omitted, *False* is assumed.
`iscobol.file.index.keycompress.padding (boolean)`	**True** = padding compression is applied on index files. **False** = padding compression is not applied on index files. If omitted, *False* is assumed.
`iscobol.file.index.lock_read_anyhow (boolean)`	**True** = locked records are returned. **False** = locked records are not returned. If omitted, *False* is assumed.
`iscobol.file.index.lock_wait (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, *False* is assumed.
`iscobol.file.index.log.debug.batchaddition (boolean)`	**True** = trace batchadditions details. **False** = do not trace batchaddition details. If omitted, *False* is assumed.

Property	Description
`iscobol.file.index.log.debug.prefetch (boolean)`	**True** = trace prefetch details. **False** = do not trace prefetch details. If omitted, *False* is assumed.
`iscobol.file.index.log.error (boolean)`	**True** = trace errors in the log file. **False** = do not trace errors in the log file. If omitted, *False* is assumed.
`iscobol.file.index.log.error.atend (boolean)`	**True** = trace EOF errors. **False** = do not trace EOF errors. If omitted, *False* is assumed.
`iscobol.file.index.log.error.notfound (boolean)`	**True** = trace NOTFOUND errors. **False** = do not trace NOTFOUND errors. If omitted, *False* is assumed.
`iscobol.file.index.log.file`	Specifies the name of the log file where to trace c-tree client operations. If omitted, no log is created.
`iscobol.file.index.log.info (boolean)`	**True** = trace generic information in the log file. **False** = do not trace generic information in the log file. If omitted, *False* is assumed.
`iscobol.file.index.log.profile (boolean)`	**True** = trace profile information in the log file. **False** = do not trace profile information in the log file. If omitted, *False* is assumed.
`iscobol.file.index.maxinstance`	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.
`iscobol.file.index.maxlencheck (boolean)`	**True** = return an error if record is larger than maxlen bytes.. **False** = record larger than maxlen bytes is allowed. If omitted, *False* is assumed
`iscobol.file.index.memoryfile (boolean)`	**True** = files are created in memory rather than on disk. **False** = files are created on disk. If omitted, *False* is assumed.

Property	Description
`iscobol.file.index.optimisticadd (boolean)`	**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 stragety when adding records.. Unique key violation is checked before writing the whole record. If omitted, *True* is assumed
`iscobol.file.index.password`	This property specifies the password for the connection to the c-tree Server.
`iscobol.file.index.path_mapping`	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.
`iscobol.file.index.prefetch (boolean)`	**True** = read operations are buffered in order to speed up performance. **False** = read operations are not buffered. If omitted, *False* is assumed. The number of buffered records is controlled by iscobol.file.index.prefetch.records. The locking on these records is controlled by iscobol.file.index.prefetch.allowwriters (boolean).
`iscobol.file.index.prefetch.allowwriters (boolean)`	**True** = prefetched records are not locked. **False** = prefetched records are locked. If omitted, *False* is assumed. **Note -** if the file has an implicit or explicit LOCK MODE AUTOMATIC WITH LOCK ON MULTIPLE RECORDS, then all the prefetched records are locked, since these records are retrieved with the same options of the READ NEXT that instigates the read a-head operation.
`iscobol.file.index.prefetch.records`	Specifies the number of read operations that must be buffered at a time. If omitted, *10* is used. **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.
`iscobol.file.index.read_lock_test`	**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. If omitted, *False* is assumed.

Property	Description
`iscobol.file.index.rpc (boolean)`	**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) If omitted, *True* is assumed
`iscobol.file.index.rpc.crc (boolean)`	**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. If omitted, *False* is assumed
`iscobol.file.index.server`	This property specifies the server name and IP address. The value can be one of the following syntaxes: • servername • servername@hostname • servername@IPaddress • #port@hostname • #port@IPaddress If omitted, then *FAIRCOMS@127.0.0.1* is assumed.
`iscobol.file.index.skiplock (boolean)`	**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. If omitted, *False* is assumed.
`iscobol.file.index.transaction (boolean)`	**True** = transaction support is on. **False** = transaction support is off. If omitted, *True* is assumed.
`iscobol.file.index.transaction.dependent`	**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). If omitted, *False* is assumed Note - this setting doesn't match with the 'dependent' information shown by the command *ctutil -info*.
`iscobol.file.index.transaction.logging (boolean)`	**True** = 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. **False** = turns off transaction logging. It is indicated when performance is more important than data safety. If omitted, *False* is assumed.

Property	Description
iscobol.file.index.trxholdslocks (boolean)	**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. **False** = all pending locks are released during a transaction commit or rollback. If omitted, *False* is assumed.
iscobol.file.index.user	This property specifies the user name for the connection to the c-tree Server. If omitted, then *guest* is assumed
iscobol.file.index.versioncheck (boolean)	**True** = check if c-tree client and server versions match during the connection at runtime initialization. **False** = do not check if c-tree client and server versions match during the connection at runtime initialization. If omitted, *False* is assumed.

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@127.0.0.1
iscobol.file.index.1.server=FAIRCOMS@192.168.0.101
iscobol.file.index.2.server=FAIRCOMS@192.168.0.102
```

The above snippet defines three instances that connect to three different servers.

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@127.0.0.1
iscobol.file.index.1.server=FAIRCOMS@192.168.0.101
iscobol.file.index.2.server=FAIRCOMS@192.168.0.102
iscobol.file.index.1.file_mapping=*
iscobol.file.index.1.path_mapping=/dir1
iscobol.file.index.2.path_mapping=/dir2
```

According to the above configuration, all files under the /dir1 folder will go on the server listening on 192.168.0.101, all files under the /dir2 folder will go on the server listening on 192.168.0.102, while all the other files will go on the server listening on 127.0.0.1.

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

Example

```
iscobol.file.index.server=FAIRCOMS@127.0.0.1
iscobol.file.index.1.server=FAIRCOMS@192.168.0.101
iscobol.file.index.2.server=FAIRCOMS@192.168.0.102
iscobol.file.index.1.file_mapping=*
iscobol.file.index.1.path_mapping=/dir1
iscobol.file.index.2.path_mapping=/dir2
iscobol.file.index.1.memoryfile=true
```

According to the above snippet, all the files on instance number 1 will be memory files.

Note - FAIRCOMS@127.0.0.1 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>
- <forcedelete>
- <hugefile>
- <ignorelock>
- <indexfilesuffix>
- <keycheck>
- <keycompress>
- <log>
- <locktype>
- <maxlencheck>
- <map>
- <memoryfile>
- <optimisticadd>
- <prefetch>
- <retrylock>
- <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@127.0.0.1">
          <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 - FAIRCOMS@127.0.0.1 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	Specifies the server name and the host name of the c-tree to connect to. The format can be one of the following syntaxes: • servername • servername@hostname • servername@IPaddress • #port@hostname • #port@IPaddress If the host name or the IP address is omitted, host name defaults to localhost.	FAIRCOMS
user	Specifies the c-tree user name.	n/a
password	Specifies the c-tree user password.	n/a
connect	Indicates whether to connect to c-tree at runtime initialization or wait for the first OPEN operation. It accepts the following values: "yes" : Indicates to connect at runtime initialization. "no" : Indicates to connect during the first OPEN operation. This is the default value.	no
versioncheck	Indicates whether to check that c-tree version matches the c-tree version. This option is turned off by default. It accepts the following values: "yes" : Turn on version matching check. Versions must match otherwise an error is returned at runtime initialization. "no" : Turn off version matching check. This is the default value.	no

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 *type="rle"*. 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 *type="zlib"*. "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>
```

<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.
<padding>	Indicates to compress the padding characters of key values.

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

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. However, if the file has an implicit or explicit LOCK MODE AUTOMATIC WITH LOCK ON MULTIPLE RECORDS, then all the prefetched records are locked, since these records are retrieved with the same options of the READ NEXT that instigates the read a-head operation.

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	When a set of records is prefetched, the prefetched records are locked with a read lock (a shared lock that allows “you” to read but not write) 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. This attribute is disabled by default (prefetch allowwriters="no"). **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.	writers

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="no">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

Example

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

<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 runitlockdetected 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	Synonim
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	Defalt 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

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 Veryan't 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, 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 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 to previous c-tree versions

If you need to interface with a previous c-tree version, you can't rely on the c-tree client library installed along with isCOBOL. You need to replace it with the proper c-tree library taken from the distribution of the c-tree that you wish to interface.

In addition, the file handler class and its alias may be different. The following table lists the file handlers available for the various c-tree versions:

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

[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
```

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.efd_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 ctsrvr 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@127.0.0.1" 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:

-sqlunlink	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.
-sqlinfo	Include the iss dictionary in the c-tree DAT file header.
-sqlllink	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.

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.

Between 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.sqlserver.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.
```

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.

  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.

Despite 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.

Command-line utilities

The following command-line utilities are available:

ctadmn

ctdump

ctfdmp

ctfileid

ctldmp

ctquiet

ctrdmp

ctsqlcdb

ctsqlutl

ctstat

ctstop

cttctx

cttrnmod

ctunf1

ctutil

dbdump

dbload

dbschema

isql

sa_admin

ctadmn

ctadmn manages the c-tree Server. It's 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 ctadmn 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
  - at the first OPEN performed by the main program of a stand-alone runtime session,
  - at the first OPEN performed by the main program of a client session in thin client environment, and
  - 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
```

  - 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.

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 ctrdmp 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 to 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 ]
```

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
```

ctrdmp

The ctrdmp utility is used to restore dumps created with ctdump.

Usage

```
ctrdmp [ dumpscript ]
```

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

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

A failed ctrdmp writes the following message to CTSTATUS.FCS when ctrdmp 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
- *-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 owner_name*: owner of table
    - *-d database_name*: database name (default: ctreeSQL)
    - *-s server_name*: c-tree SQL Server name (default: FAIRCOMS)
    - *-u userid*: userid for logging onto the c-tree SQL Server
    - *-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
  - t ctTRNLOG
  - p ctPREIMG
  - 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:
  - a Add
  - r Read
  - 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]] [-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.

-checkimg

Checks file integrity using the physical file information.

Usage

```
ctutil -checkimg 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:

  - *MaxRecSize* is a five-digit number representing the maximum record length.
  - *MinRecSize* is a five-digit number representing the minimum record length.
  - *NumKeys* is a three-digit number representing the number of keys in the file.
  - *NumSegs* is a two-digit number representing the number of segments in a key.
  - *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.

- *SegSize* is a three-digit number representing the size of a segment in a key.
- *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
```

- *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:
  - A (ANSI convention, default)
  - B (MBP convention)
  - D (Data General convention)
  - I (IBM convention)
  - M (Micro Focus convention)
  - N (NCR convention)
  - R (Realia convention)
  - V (VAX convention)

-copy

Creates a copy of a c-tree file.

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.

-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.

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 b

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 ctutil -filecopy and -copy options 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:

  - *MaxRecSize* is a five-digit number representing the maximum record length.
  - *MinRecSize* is a five-digit number representing the minimum record length.
  - *NumKeys* is a three-digit number representing the number of keys in the file.
  - *NumSegs* is a two-digit number representing the number of segments in a key.

- *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.
- *SegSize* is a three-digit number representing the size of a segment in a key.
- *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]] [-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.
- -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:

  - *MaxRecSize* is a five-digit number representing the maximum record length.
  - *MinRecSize* is a five-digit number representing the minimum record length.
  - *NumKeys* is a three-digit number representing the number of keys in the file.
  - *NumSegs* is a two-digit number representing the number of segments in a key.
  - *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.
  - *SegSize* is a three-digit number representing the size of a segment in a key.
  - *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:

  - *MaxRecSize* is a five-digit number representing the maximum record length.
  - *MinRecSize* is a five-digit number representing the minimum record length.
  - *NumKeys* is a three-digit number representing the number of keys in the file.
  - *NumSegs* is a two-digit number representing the number of segments in a key.

- *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.
- *SegSize* is a three-digit number representing the size of a segment in a key.
- *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:
  - A - ACUCOBOL-GT (default)
  - B - MBP COBOL
  - D - Data General
  - I - IBM
  - M - Micro Focus
  - N - NCR COBOL
  - R - Realia COBOL
  - 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.
  - use A for programs compiled with -dca option or without data compatibility options.
  - use D for programs compiled with -dcd option.
  - use I for programs compiled with -dci option.
  - use M for programs compiled with -dcm option.
  - use N for programs compiled with -dcn option.
  - use R for programs compiled with -dcr 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@127.0.0.1" 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.

-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@127.0.0.1" 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 [-symb=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 *-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.
    - use A for programs compiled with -dca option or without data compatibility options.
    - use D for programs compiled with -dcd option.
    - use I for programs compiled with -dci option.

- use M for programs compiled with -dcm option.
- use N for programs compiled with -dcn option.
- use R for programs compiled with -dcr 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@127.0.0.1" 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.

-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.

  - use A for programs compiled with -dca option or without data compatibility options.
  - use D for programs compiled with -dcd option.
  - use I for programs compiled with -dci option.
  - use M for programs compiled with -dcm option.
  - use N for programs compiled with -dcn option.
  - use R for programs compiled with -dcr 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.
  - use A for programs compiled with -dca option or without data compatibility options.
  - use D for programs compiled with -dcd option.
  - use I for programs compiled with -dci option.
  - use M for programs compiled with -dcm option.
  - use N for programs compiled with -dcn option.
  - use R for programs compiled with -dcr 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@127.0.0.1" 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:
    - 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.
    - 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.
    - W - Synchronous writes (ctWRITETHRU c-tree file mode). This mode is indicated where transaction processing in not suitable yet data safety is still critical.
    - 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:
  - *-u user_name*: user name to connect to the database.
  - *-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 utilityloads 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:
  - *-u user_name*: user name to connect to the database.
  - *-a password*: password to connect to the database.
  - *-z maximum multiple inserts*: maximum number of records to be inserted at one time in each bulk insert (used to improve performance)
  - *-l logfile*: the file into which the error logging is done. stderr is the default.
  - *-b badfile*: the file into which the bad rows that were not loaded, are written. By default badfile is put in the current directory.
  - *-c commit_frequency*: the specified number of records before committing the transaction. The default frequency is 100 records.
  - *-e maxerrs*: the maximum number of tolerable errors. The default number is 50 errors.
  - *-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.
  - *-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

Graphical utilities are provided as executable files for the Windows platform.

Most of them are available also as Java programs so they can run on other platforms as well.

The following graphical utilities are available:

c-treeACEMonitor

c-treeConfigManager

c-treeGauges

c-treeISAMExplorer

c-treeLoadTest

c-treeLogAnalyzer

c-treePerfMon

c-treeQueryBuilder

c-treeSecurityAdmin

c-treeSqlExplorer

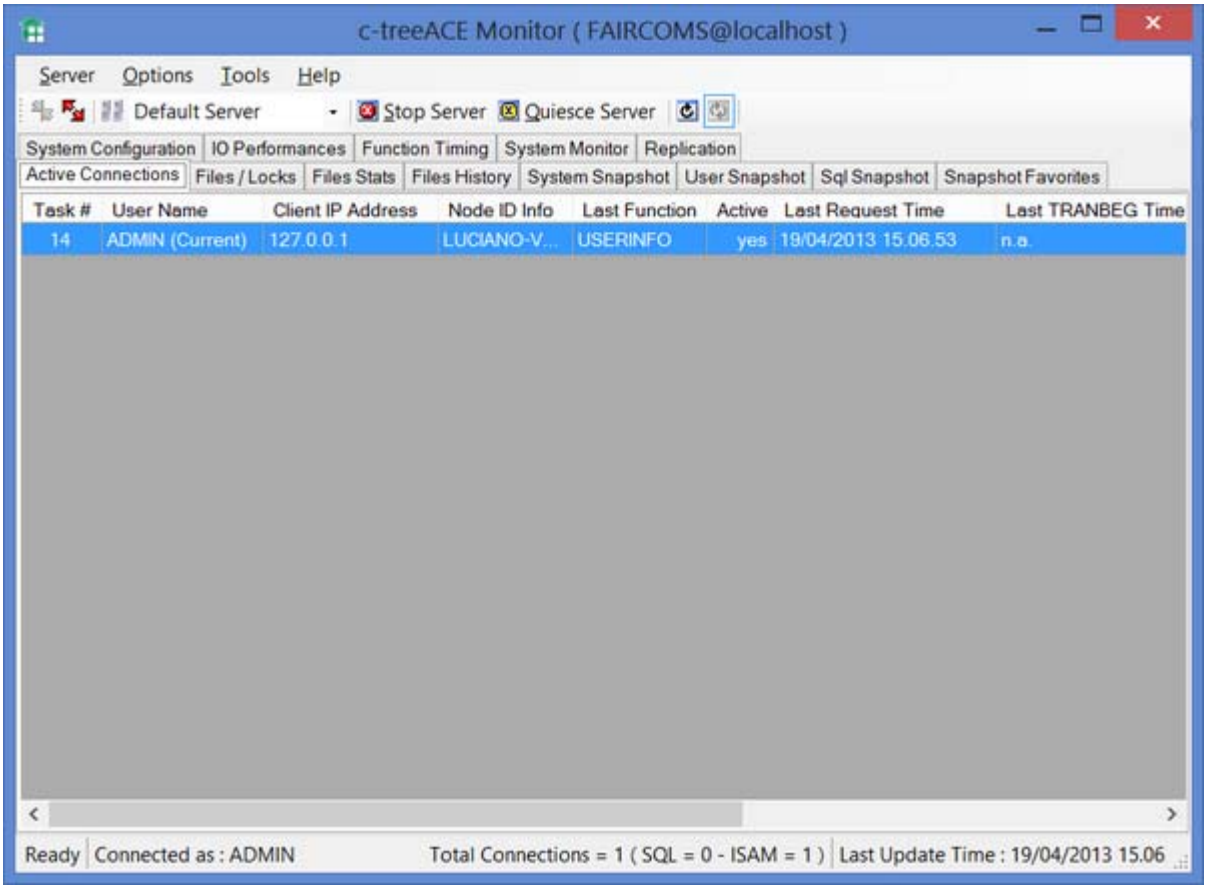
c-treeTPCATest

DrCtree

ErrorViewer

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.

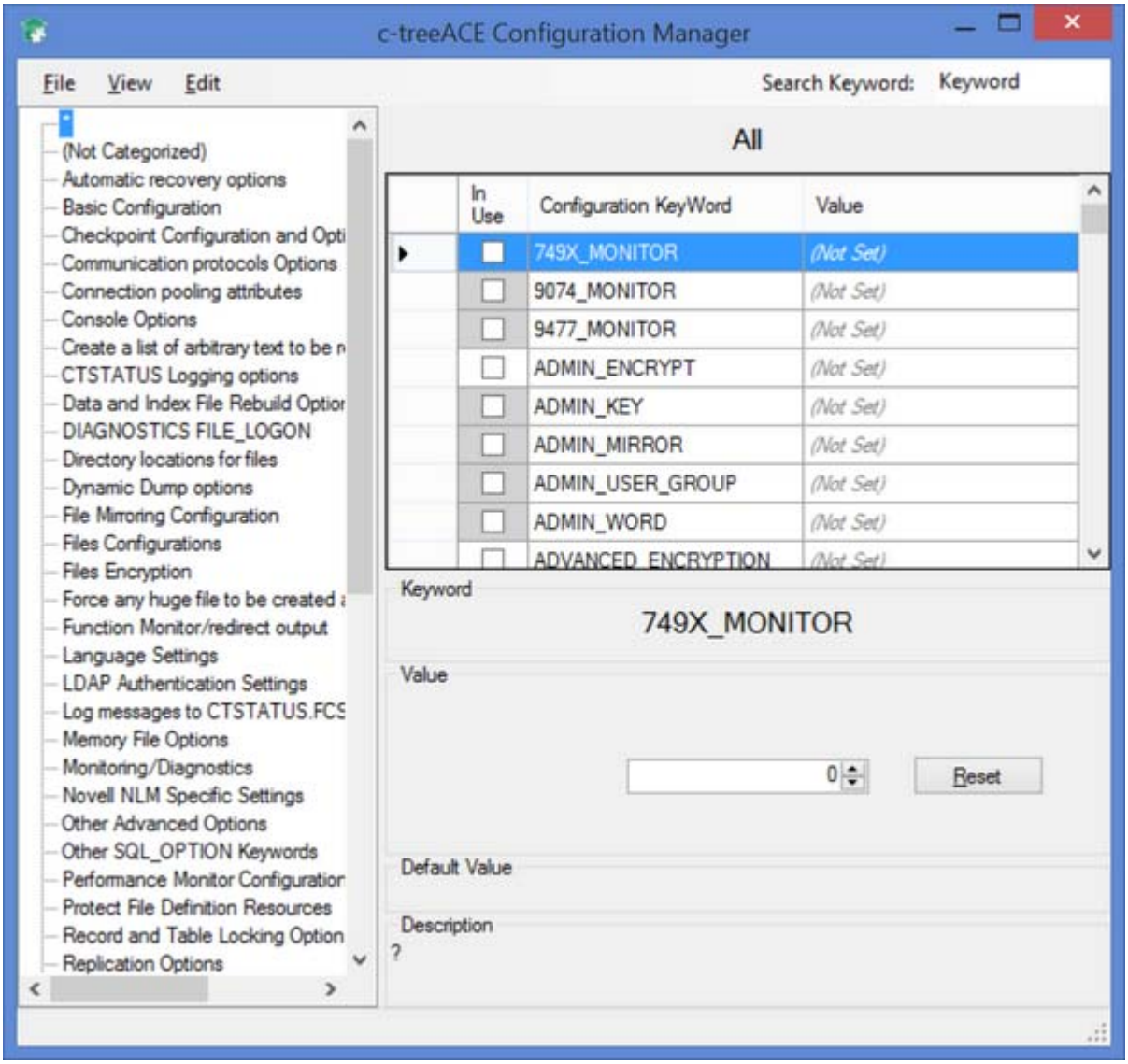


A Java version of this utility is provided through the library *c-treeACEMonitor.jar*.

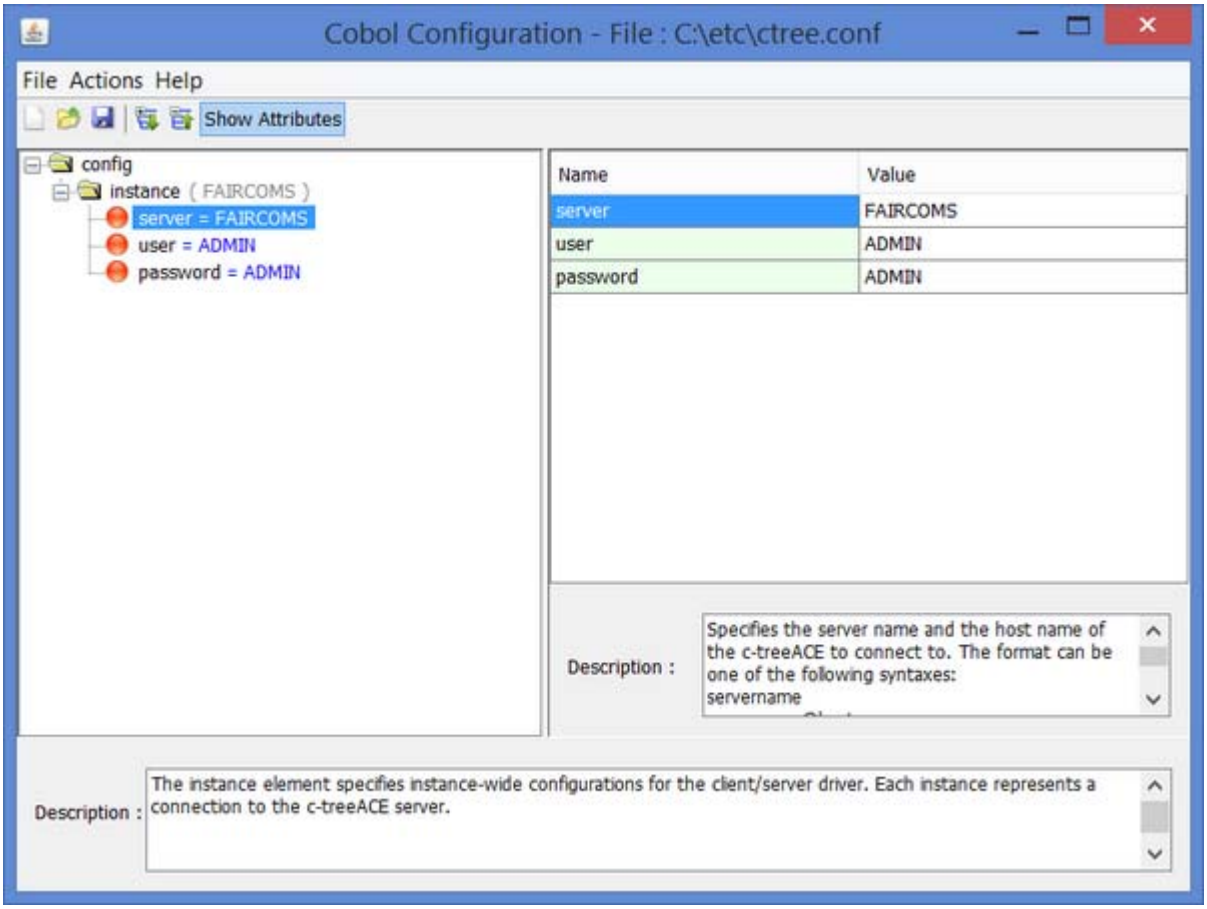
Refer to c-tree Server Administrator's Guide for additional information on this utility.

c-treeConfigManager

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



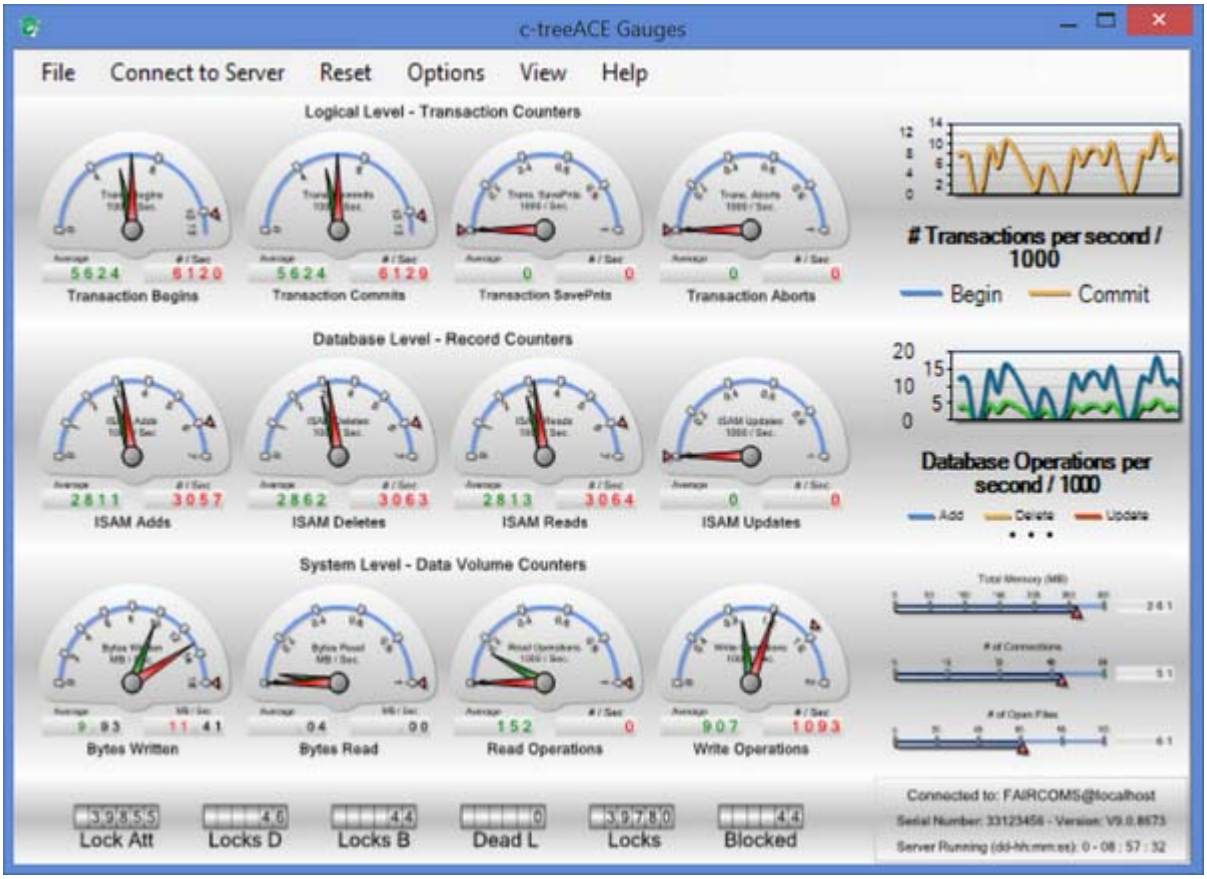
The client configuration file (ctree.conf) can be edited using the Java utility provided through the library *CobolConfig.jar*.



Refer to c-tree Server Administrator's Guide for additional information on these utilities.

c-treeGauges

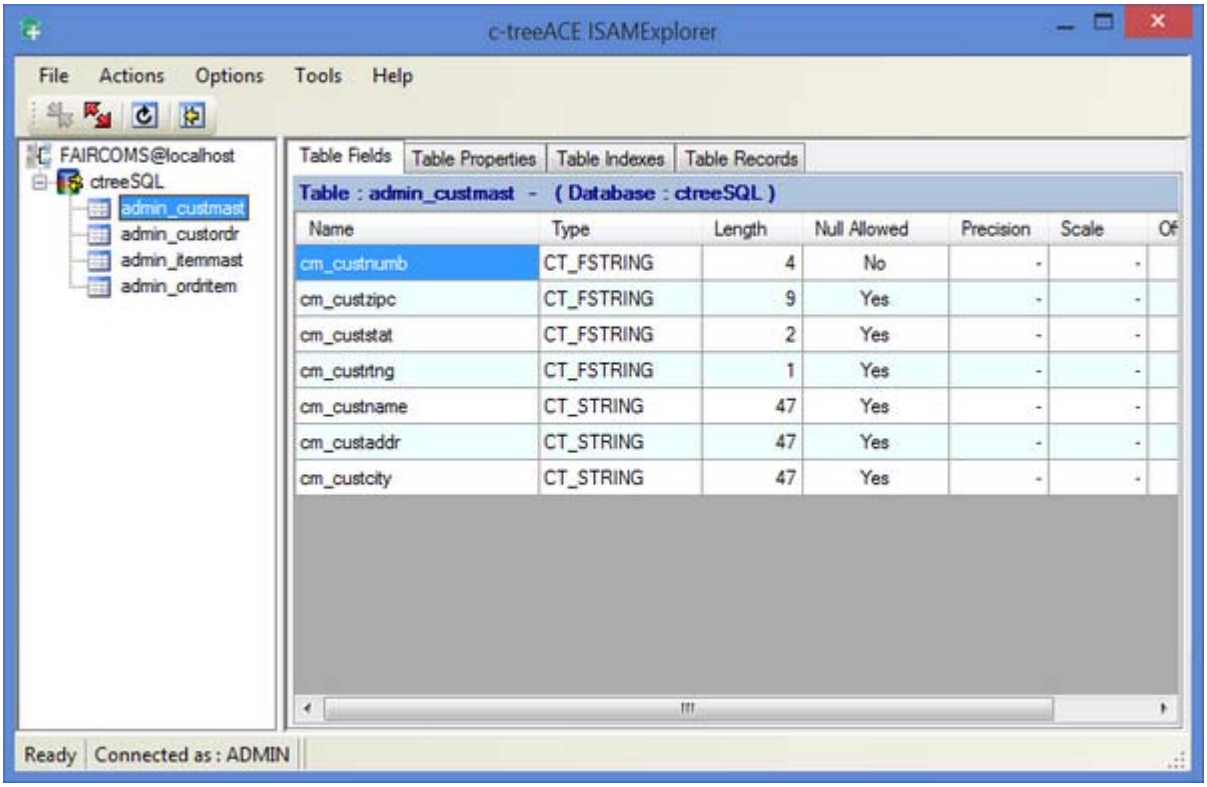
c-treeGauges is a graphical display of performance metrics provided for the Windows platform. While not intended as a full monitoring solution, c-treeGauges makes it easy to begin understanding the internal workings of the c-tree server.



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

c-treeISAMExplorer

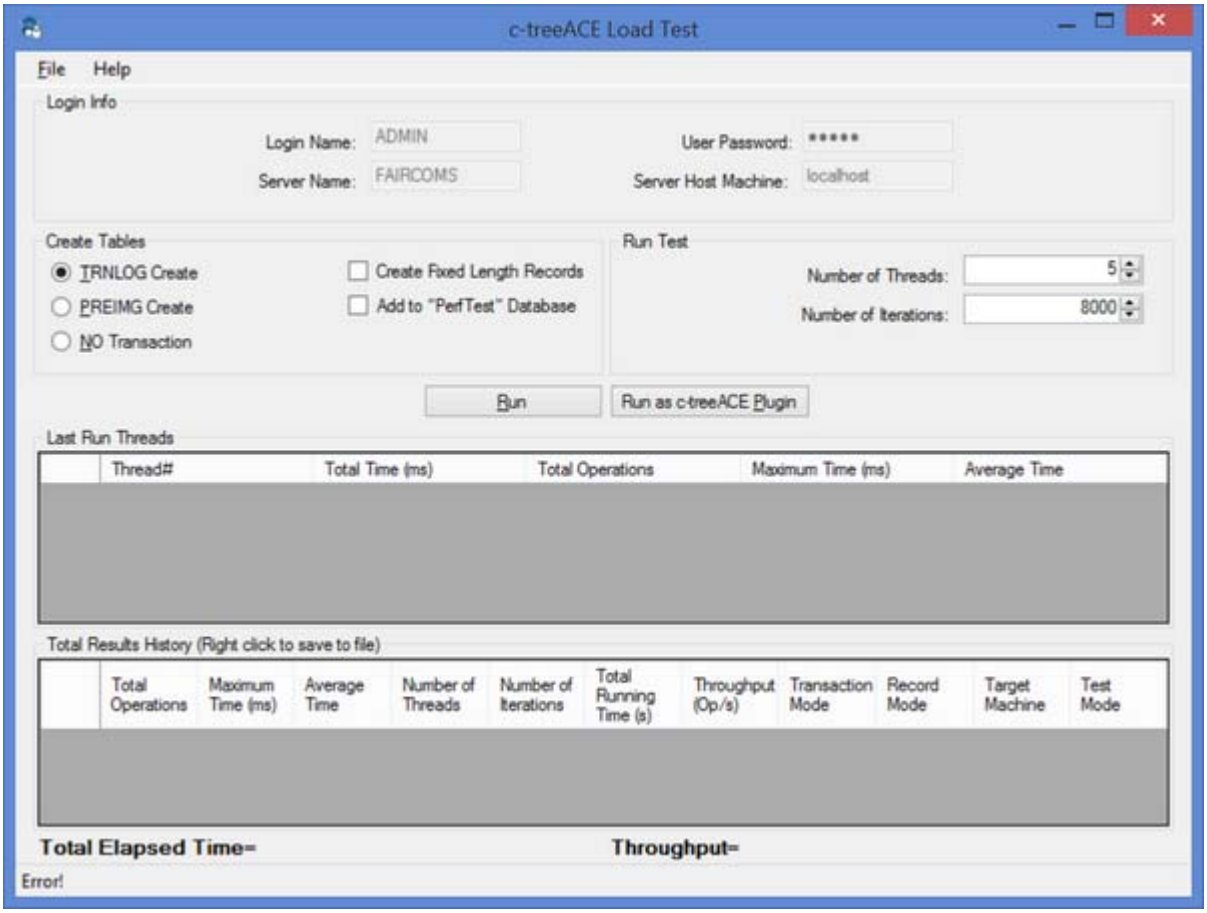
c-treeISAMExplorer is a graphical file management utility provided for the Windows platform. It shows all the files available under a given c-tree server along with their attributes and record content.



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.

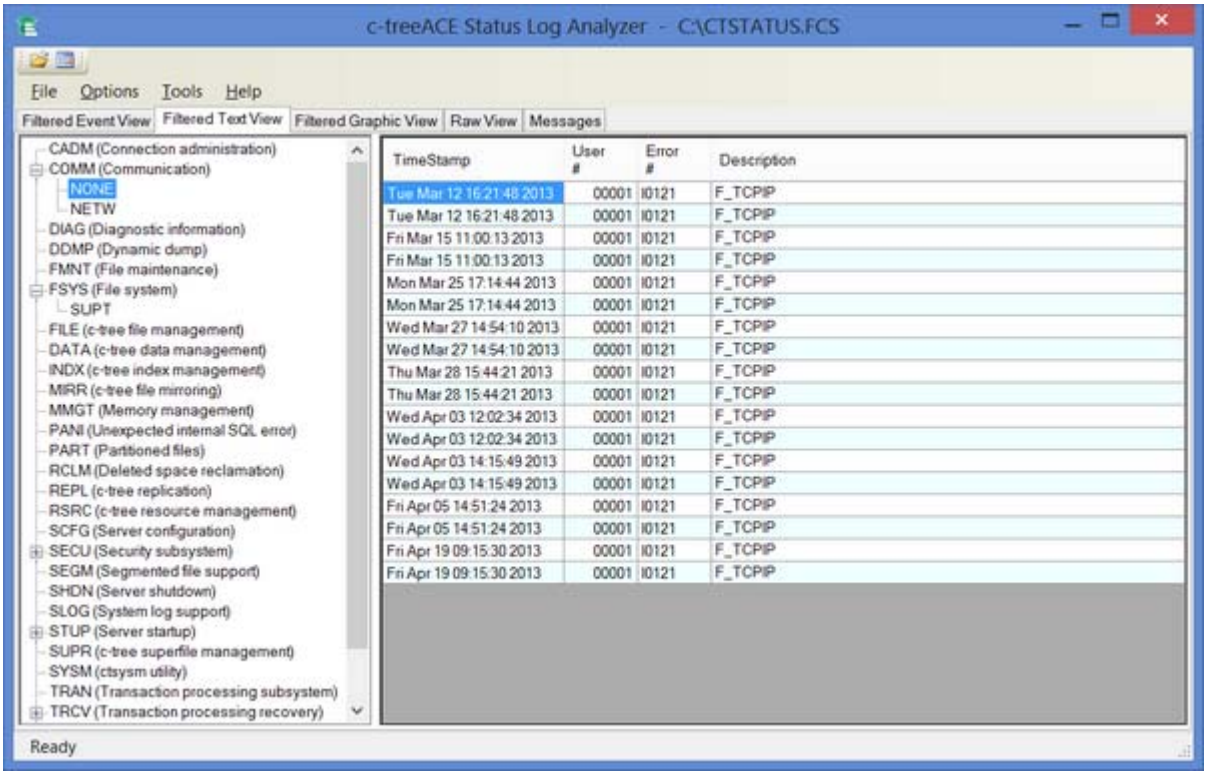


Refer to c-tree Server Administrator's Guide for additional information on this utility.

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.

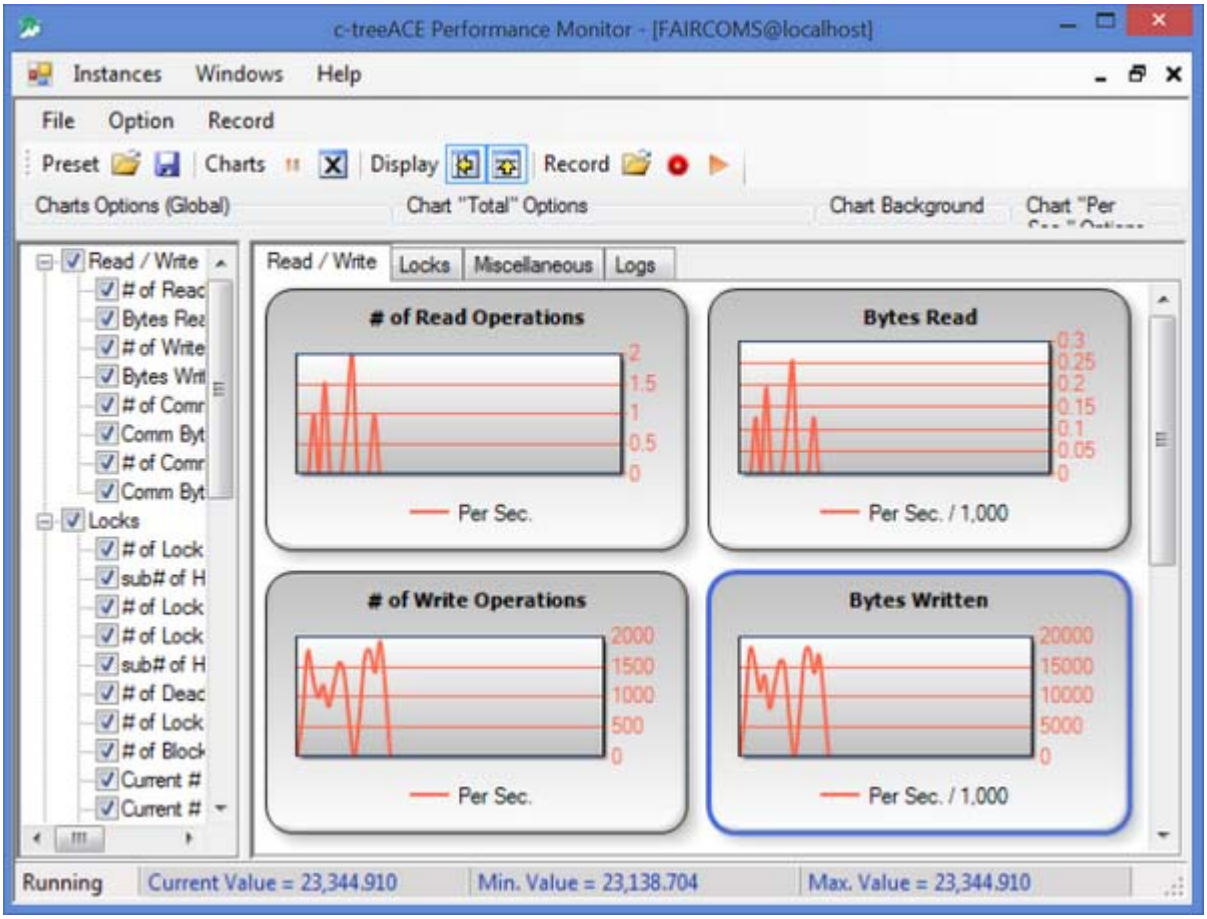
As the utility starts, click on the *File* menu and choose *Open*. Browse for the CTSTATUS.FCS file stored in the c-tree server directories and open it.



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

c-treePerfMon

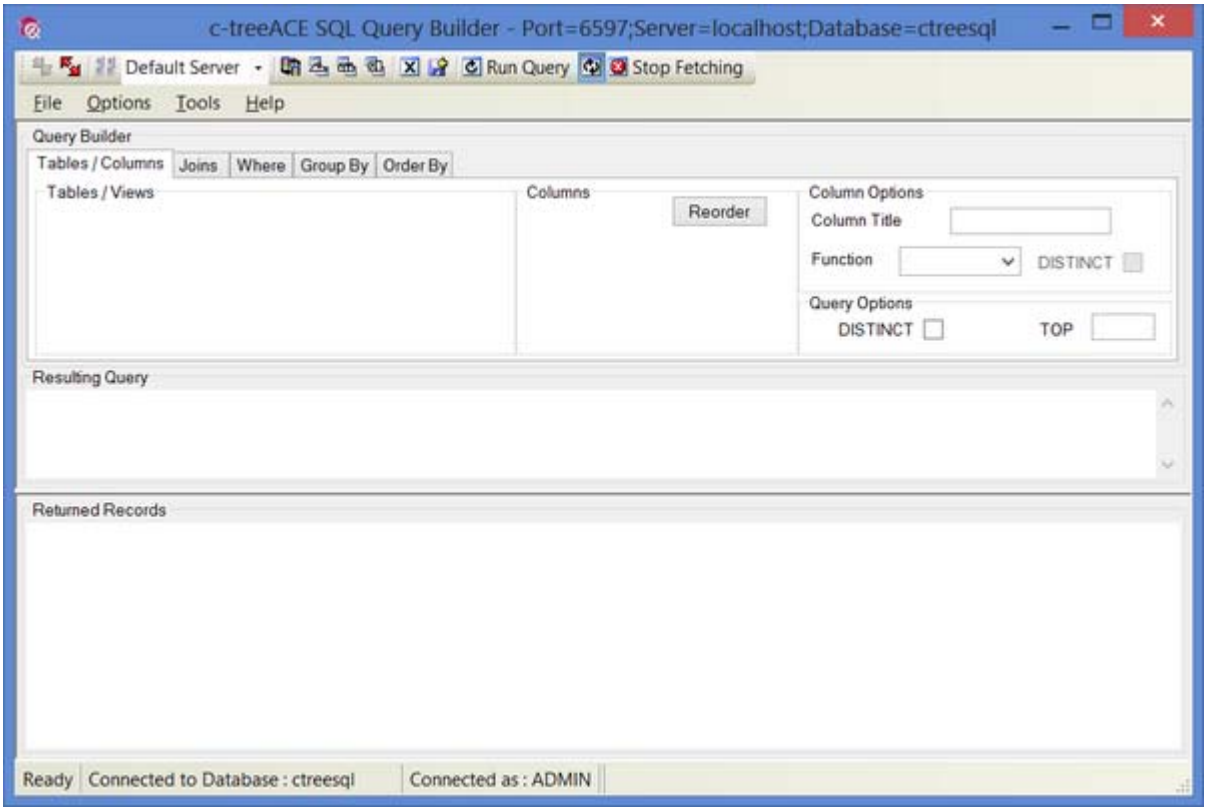
c-treePerfMon is a graphical utility provided for the Windows platform. It graphically displays critical c-tree operational parameters in real time enabling you to quickly spot performance bottlenecks. Sorted by functional categories, c-treePerfMon allows you to hone in on the exact metrics you wish 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 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.

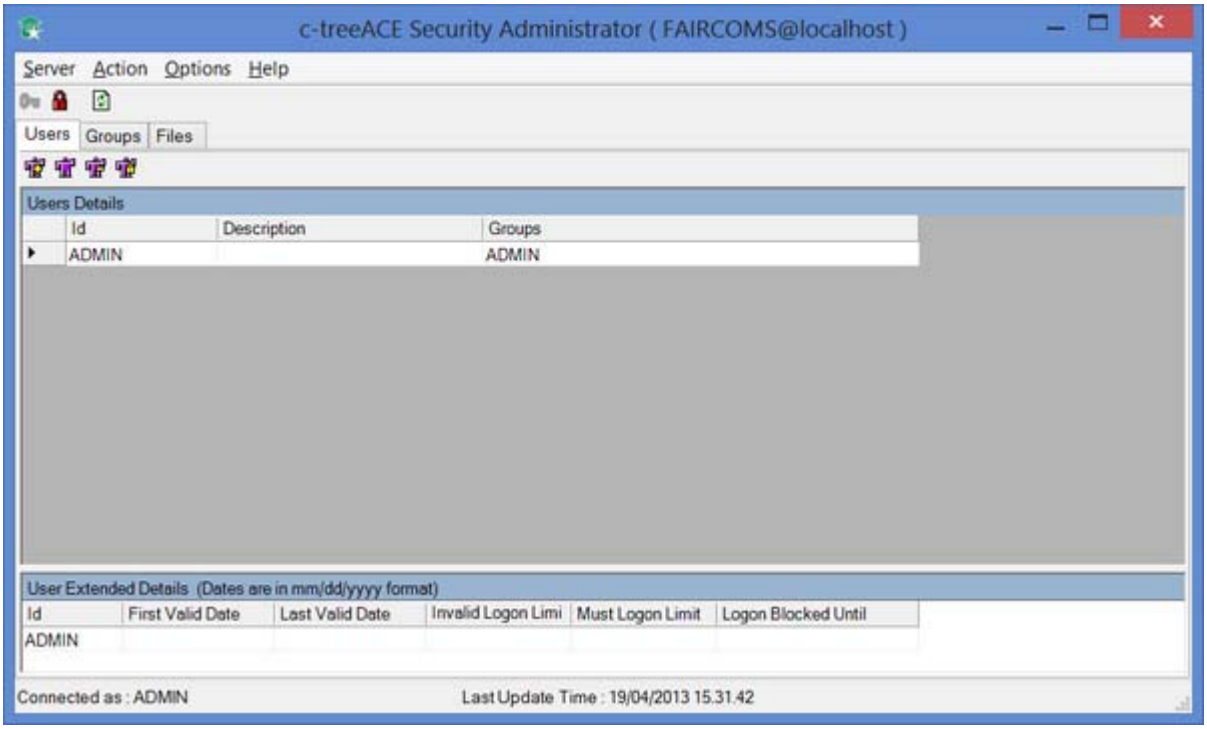
The SQL query is updated in the *Resulting Query* field in real time.

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.



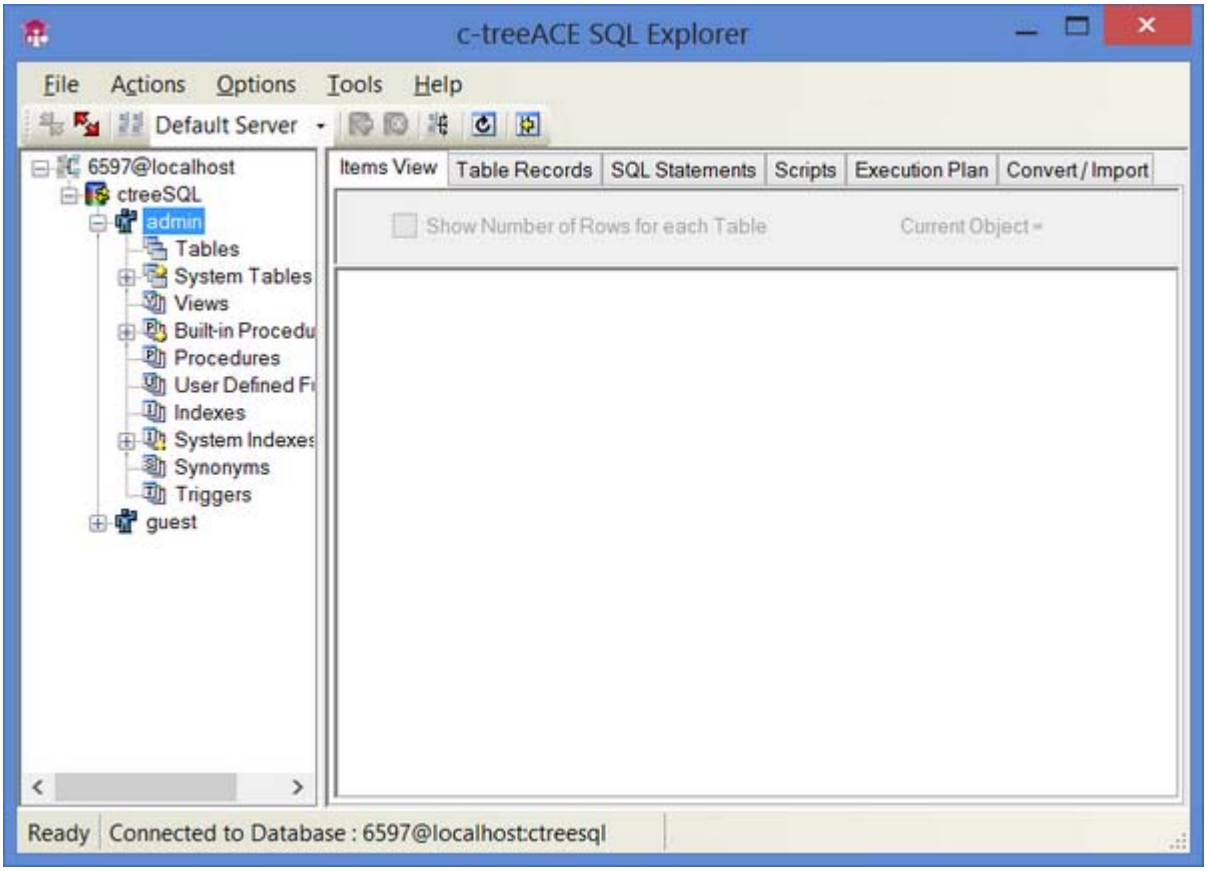
ctreeSecurityAdmin allows to:

- 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.

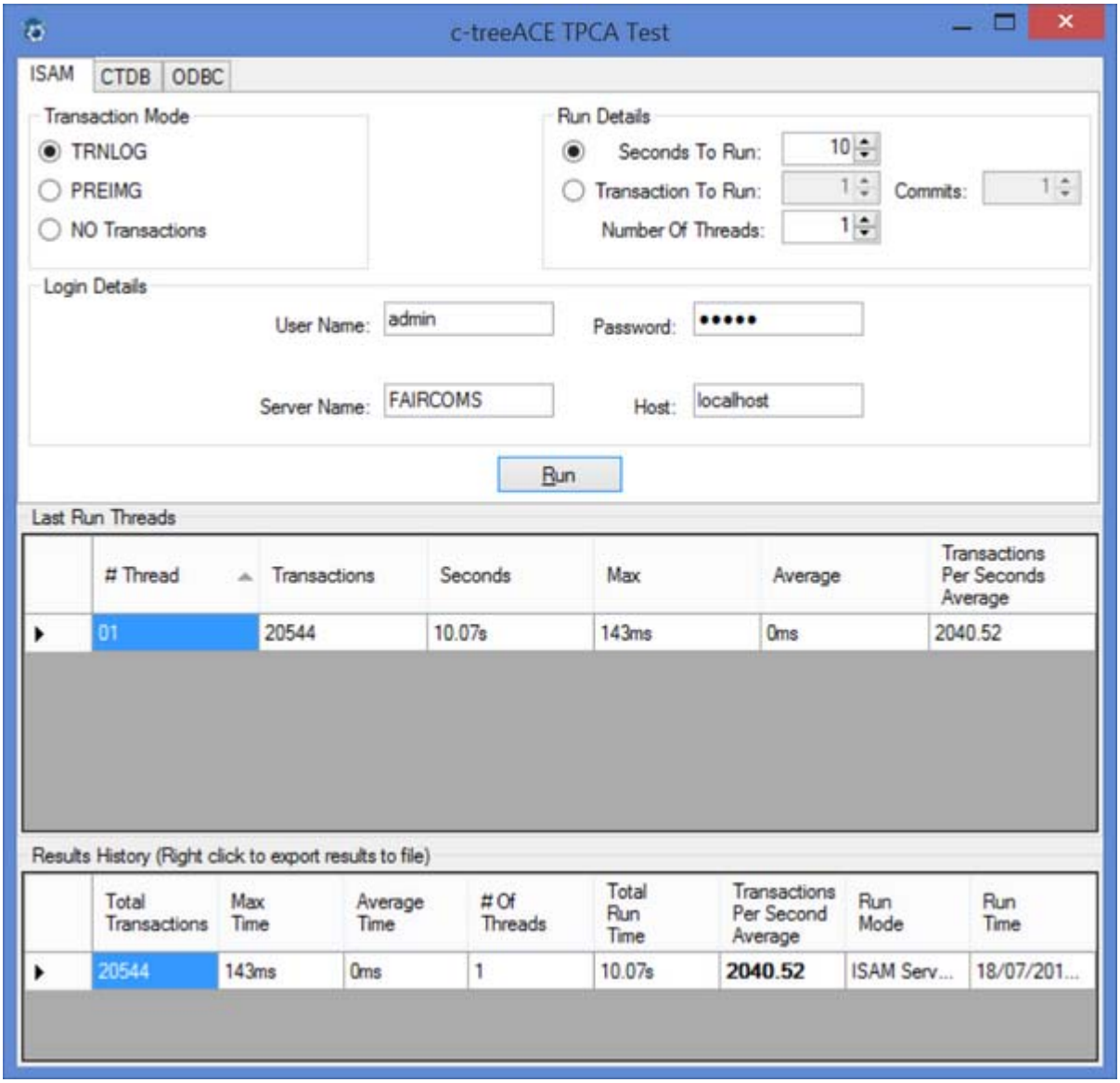
c-tree SQL Explorer also allows to manage users if you start it with the following command:

```
c-treeSqlExplorer -adv
```

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

c-treeTPCATest

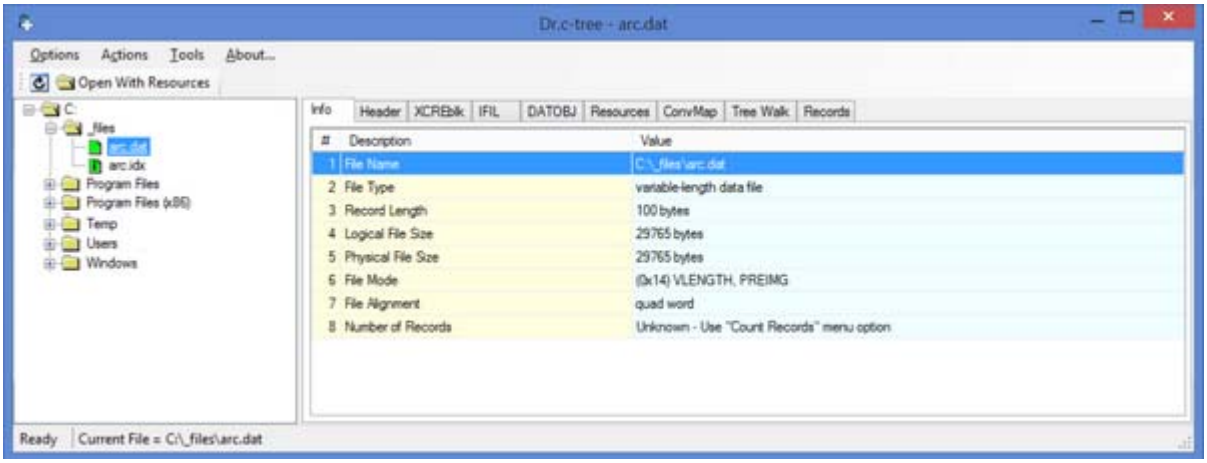
c-treeTPCATest is a graphical performance benchmark provided for the Windows platform.



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

DrCtree

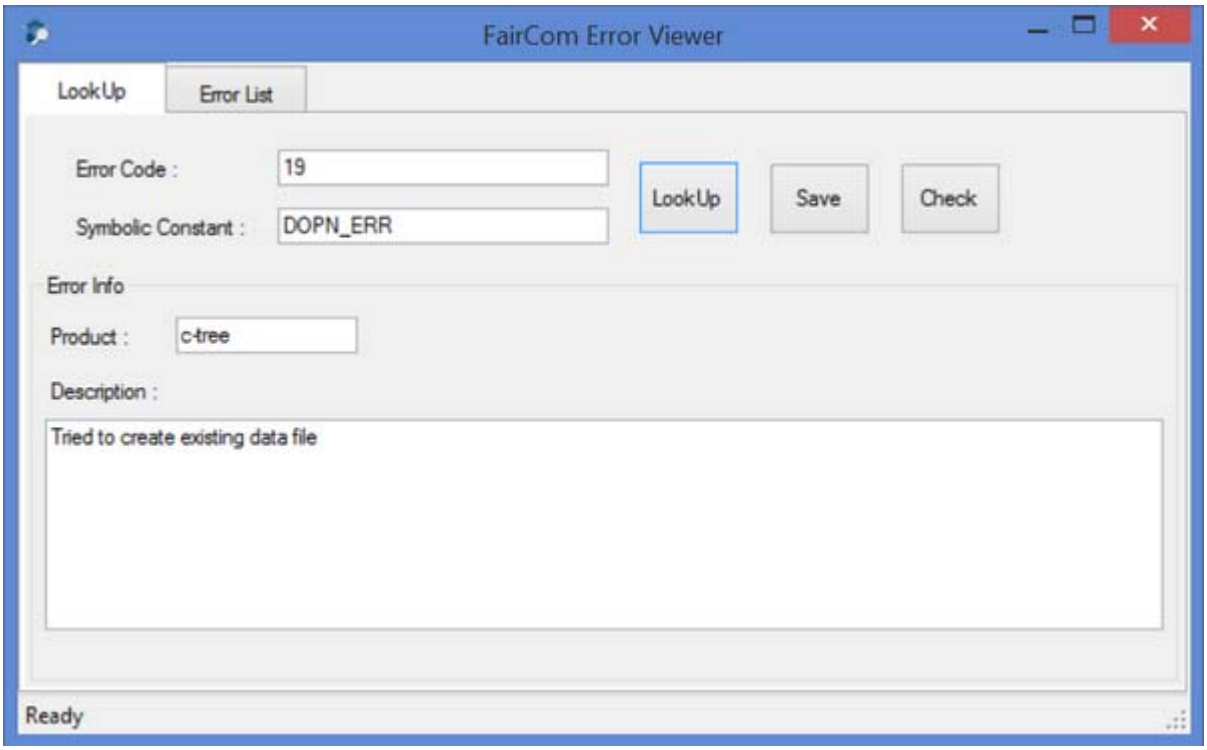
DrCtree is a graphical file management utility provided for the Windows platform. It allows to browse for c-tree files on the local drive by using the file explorer on the left. When you select a c-tree file, information is displayed on the right. Right clicking on the file name or using the Actions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

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. shut down the c-tree server; you can rely on the ctstop utility for this
2. copy the files using system commands or third party utilities
3. 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
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 peformed once a day.

For more information about dump script file entries, see http://www.faircom.com/doc/ctserver/#8380.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, ctrdmp.

The ctrdmp 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 not running when ctrdmp starts because two c-tree servers operating simultaneously interfere with each other.
- Because it is a c-tree server, ctrdmp generates temporary versions of all system files associated with a c-tree server (i.e., files with the extension ".FCS" as described above). Therefore, the dynamic dump file and the ctrdmp utility should be moved to a directory other than the working directory where the c-tree server undergoing recovery resides. This is so the system files created by the recovery program will not overwrite working c-tree server files.

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.:

```
ctrdmp /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 - form non-transaction files the following warning is traced "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 ctreedbs library instead of c-tree library.

The ctreedbs library and its dependent libraries must be available in the library path.

The ctsrvr.cfg configuration file must be available in the working directory as well as the c-tree license file. The c-tree license file can also be stored in another folder and pointed by the FCSRVR_LIC environment variable, e.g.

```
FCSRVR_LIC=/path/to/ctsrvr########.lic
```

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 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 CSTATUS.FCS generated in thec-tree Server's local directory. By default CTSTATUS.FCS is generated in the subfolder *Server/sql/data*. 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 veresions of these components, e.g. a ctree.dll v10.4 is trying to connect to a c-tree server v11.6.

Logical Error	Meaning
36	The most common cause of this error is that you're opening a file that is not a c-tree file. JIsam 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 JIsam 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: 5597and 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.

Chapter 11

Drivers

c-tree provides the following drivers.

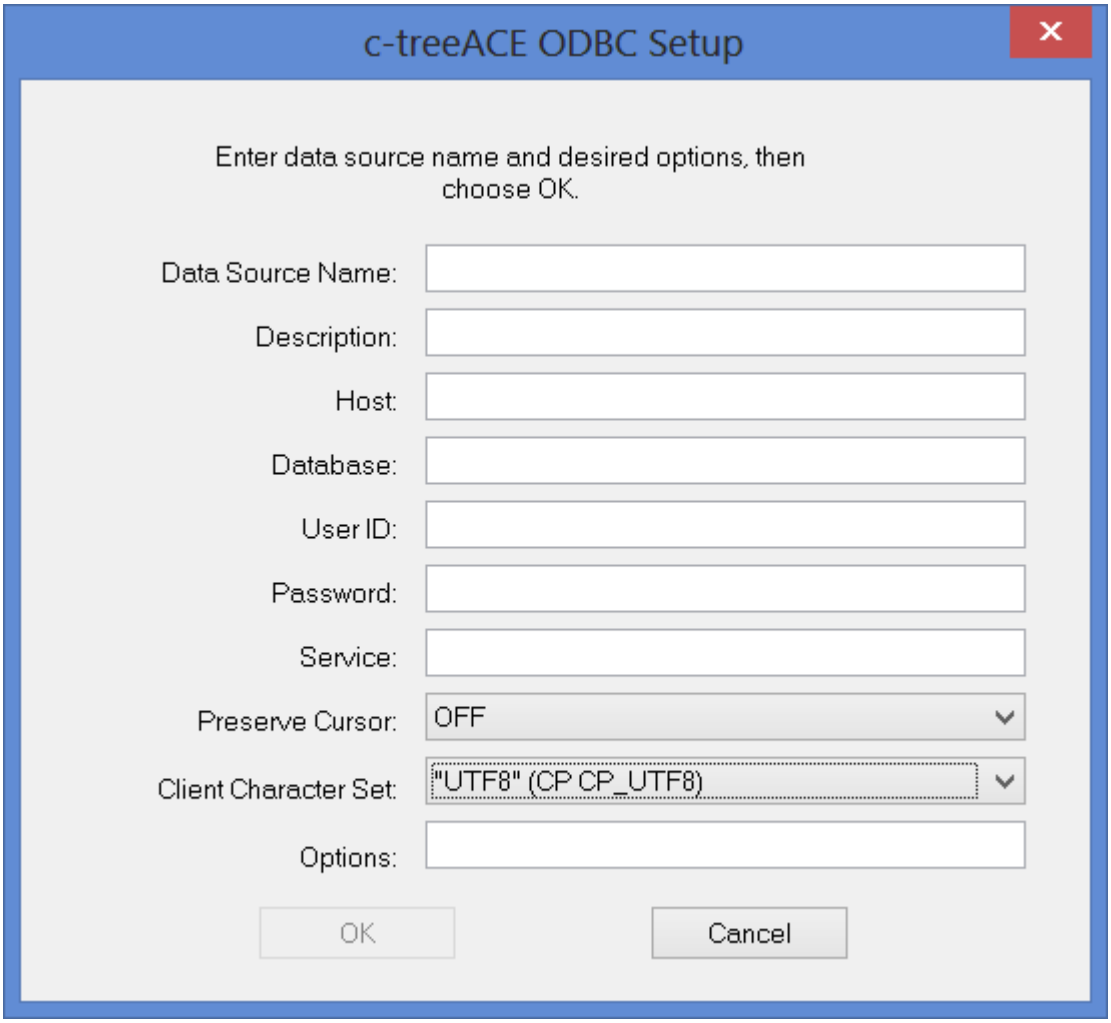
ODBC Driver

The ODBC Driver is installed on Windows machines along with c-tree if the proper option is checked. See “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 "c-treeACE ODBC Driver" from the list of drivers

The following panel will appear:



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 *driver/sql.odbc* in the c-tree directory.

Please consult Faircom's documentation for additional information.

JDBC Driver

The JDBC Driver is installed on Windows machines along withc-tree if the proper option is checked. See "Installing on Windows" chapter for details.

On other platforms the driver library is installed under *driver/sql.jdbc* in the c-tree directory.

The driver library is named "ctreeJDBC.jar" and must be available in the Java Classpath in order to take advantage of its classes.

The following code snippet shows how to connect to c-tree SQL from a Java program:

```
Class.forName ("ctree.jdbc.ctreeDriver");
conn = DriverManager.getConnection ("jdbc:ctree:6597@localhost:ctreeSQL", "ADMIN",
"ADMIN");
```

Please consult Faircom's documentation for additional information.

ADO.NET Driver

The ADO.NET Driver is installed on Windows machines along with c-tree if the proper option is checked. See "Installing on Windows" chapter for details.

Please consult Faircom's documentation for additional information.

PHP Driver

The PHP Driver is installed on Windows machines along with c-tree if the proper option is checked. See "Installing on Windows" chapter for details.

On other platforms the driver library is installed under *driver/sql.php* in the c-tree directory.

Please consult Faircom's documentation for additional information.

PYTHON Driver

The PYTHON Driver is installed on Windows machines along with c-tree if the proper option is checked. See "Installing on Windows" chapter for details.

On other platforms the driver library is installed under *driver/sql.python* in the c-tree directory.

Please consult Faircom's documentation for additional information.

Chapter 12

External References

To retrieve further information about c-tree administration and troubleshooting, the following links can be visited:

c-tree Server Administrator's Guide	http://www.faircom.com/doc/ctserver/
c-treeACE Error Codes	https://docs.faircom.com/doc/ctreepp/60838.htm
c-treeDB Error Codes	https://docs.faircom.com/doc/ctreepp/25734.htm