test link

c-treeACE

V9 Update Guide

Audience

Developers

Subject

FairCom's high-performance NAV and SQL database technology.

FairCom®

Contents

Documentation Overview

Purpose of this Manual

This manual provides detailed descriptions of the new features, enhancements, and other fixes for the entire FairCom DB Professional product line. You will discover that these new features and functionality deliver significant new benefits to your application development efforts. These updates include:

- FairCom DB V9.0 Professional
- FairCom DB SQL V9.0
- FairCom DB SQL c-treePHP
- FairCom DB SQL c-treeDBX
- c-treeVCL
- c-tree ODBC Drivers

Audience

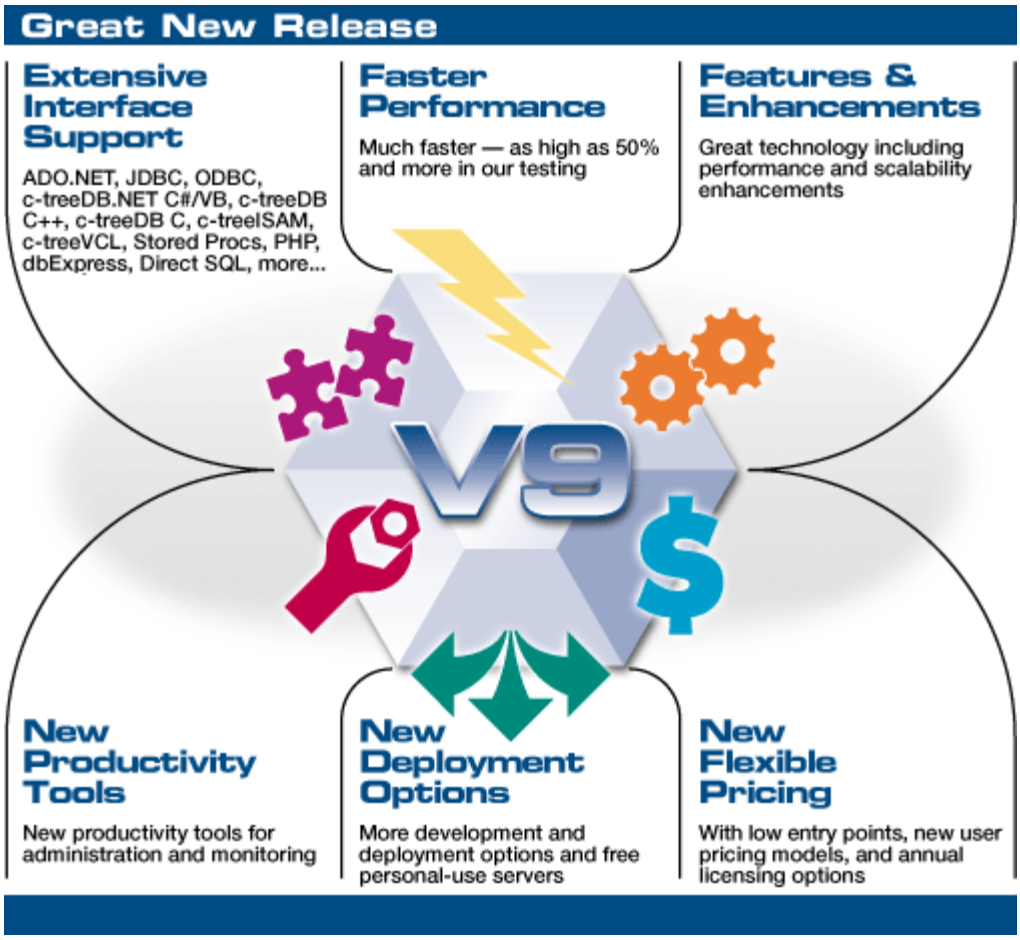
This manual is directed to existing FairCom DB developers interested in learning about the latest technology available from FairCom. Many new features and enhancements have been added to FairCom DB and developers are invited to take a tour through this high level overview of all that FairCom DB has to offer. This manual is presented as an overview for you, the FairCom DB developer, who may already have extensive knowledge of FairCom DB. Complete details can be found in the updated online manuals containing all of the latest information.

Structure

This manual contains the following chapters:

Chapter 1	Highlights and overview of FairCom DB V9.0
Chapter 2	Overview of new FairCom DB Tools
Chapter 3	FairCom DB SQL Enhancements
Chapter 4	New Features available with the FairCom DB ISAM Server
Chapter 5	New FairCom DB Interface Technologies
Chapter 6	Enhancements and new features of FairCom DB for .NET
Chapter 7	Enhancements and new features of FairCom DB DataBase Layer - c-treeDB
Chapter 8	Enhancements and new features of the FairCom DB SDK
Chapter 9	Enhancements and new features of c-treeVCL
Chapter 10	Enhancements and new features of FairCom DB Utilities

1. Highlights of FairCom DB V9.0



Designed around a philosophy of simplicity and ease of use, FairCom DB is the next extension of traditional c-tree Plus database technology. Extending these concepts, FairCom DB now comes in two varieties: **FairCom DB Express** and **FairCom DB Professional**. This manual focuses on **FairCom DB Professional** for existing developers.

c-treeACE®

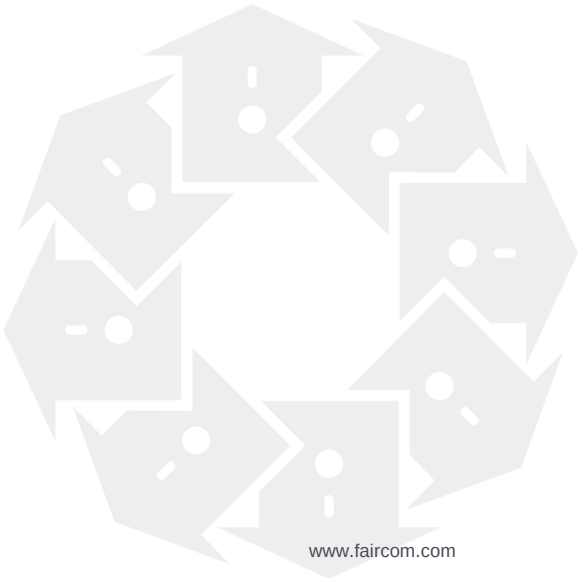
FairCom DB Professional is the traditional c-tree database development package you've come to know and respect for performance, control and flexibility. FairCom DB Professional is a full database development SDK complete with source code supporting nearly any platform with a 'C' compiler. FairCom DB Professional provides absolute control over every aspect of your database and porting when required for a new or unique platform. In FairCom DB Professional you will find support for FairCom's traditional high performance standalone technology.

FairCom DB V9 brings not only a new ease of use philosophy of working with c-tree database technology, it brings about an extensive array of new features and enhancements. Check out the new FairCom DB tools. This broad suite of graphical utilities brings a new level of ease in designing and working with your c-tree databases. The new FairCom DB SQL ADO .NET Data

Provider brings seamless integration into your .NET projects and Visual Studio. Quickly build a data-centric application in minutes with this powerful new interface.

The FairCom DB core has never been more reliable and now includes advanced new file blocking and server quiesce features sure to please the advanced database administrator. FairCom DB API has been greatly enhanced with a huge number of new functions and features, many also available in the FairCom DB API .NET component for your .NET applications.

Check out all that FairCom DB has to offer and see how to put FairCom DB database technology into your applications today. Read on and discover how FairCom's latest FairCom DB database technology delivers exceptional value.



1.1 FairCom DB Tools

Because FairCom DB is designed for simplicity of use, a complete suite of tools is included. The graphical interfaces make it incredibly easy to take full advantage of everything FairCom DB has to offer. From the lowest ISAM data file diagnosis to complex FairCom DB SQL queries, there is a tool for the job.

Java-Based Tools

A set of *Java-based tools* (https://docs.faircom.com/doc/*faircom-graphical-tools/LegacyGraphicalToolsforAllPlatforms.htm*) is available for supported platforms that have Java installed. They are located in *FairCom/V10.0.0/<platform directory>/tools/guitools.java* (where *V10.0.0* corresponds to your FairCom version).

FairCom DB Explorer

Your “one-stop” utility to view and manage ISAM and SQL tables and data in your FairCom system.

FairCom DB Monitor

This Dashboard displays a wealth of statistics for real-time performance monitoring and administration.

Error Viewer

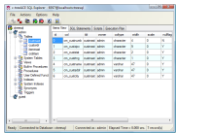
A convenient way of viewing error messages you see in the logs.

Dr. c-tree

A graphical interface for very advanced users to work with data files, index files, and data dictionaries.

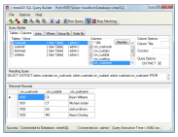
Windows Tools

The tools shown below are available for installations on the Windows platform. These tools can be found in the **FairCom** folder of the Windows **Start** menu.



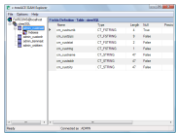
FairCom DB SQL Explorer (page 27)

FairCom DB SQL Explorer is a comprehensive SQL utility for FairCom DB. Define and manage databases, users and tables. Display data. Create and manage your stored procedures, triggers and user defined functions. Use this tool for every aspect of FairCom DB SQL management.



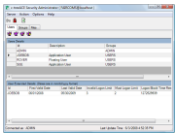
FairCom DB SQL Query Builder (page 29)

FairCom DB SQL Query Builder is a tool to build and execute complex queries against your data. Simply select your tables, join methods and sort criteria from the available options. The powerful FairCom DB SQL engine will return your selected data. Use this tool to optimize your most sophisticated queries for performance.



FairCom DB ISAM Explorer (page 30)

FairCom DB ISAM Explorer is designed to manage your ISAM tables when not using FairCom DB SQL. With FairCom DB ISAM Explorer you can create and manage tables and take advantage of every c-tree option available for the utmost in control.

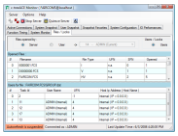


FairCom DB Security Administrator (page 31)

FairCom DB Security Administrator is a quick and easy graphical utility to modify FairCom DB user, group and file security attributes.

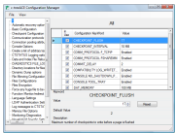
FairCom DB Performance Monitor (page 34)

FairCom DB Performance Monitor provides graphical real-time monitoring for many FairCom DB SNAPSHOT statistics. Using graphical charts, your data is displayed and can be color coded to enhance your analysis.



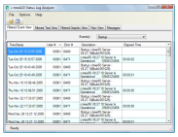
FairCom DB Monitor (page 35)

FairCom DB Monitor is a comprehensive monitoring tool to capture and view all of the FairCom DB SNAPSHOT statistics. Bookmark your favorite statistics for analysis.



FairCom DB Configuration Manager (page 33)

The Configuration Manager allows you to conveniently edit your FairCom DB configuration file, *ctsrvr.cfg*. The complete list of FairCom DB configuration keywords is now at your fingertips.

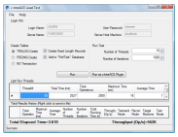


FairCom DB Status Log Analyzer (page 39)

FairCom DB Status Log Analyzer quickly displays events in the FairCom DB Status Log, *CTSTATUS.FCS*. With this tool, you can spot unexpected problems, as every event is flagged by category and color coded.

FairCom DB Gauges (page 37)

FairCom DB Gauges is a monitoring tool displaying your FairCom DB *SNAPSHOT* statistics in an ultra-modern panel of gauges, much like the dashboard of your car. Quickly view current performance metrics and watch the gauges spin in real time.



FairCom DB Load Test (page 40)

FairCom DB Load Test is a great tool to performance test FairCom DB. This tool will display benchmark performance results that you should expect from a typical FairCom DB application. Prove to yourself how fast FairCom DB performs directly on your system.

Note: Beginning with FairCom DB V11, the older .NET framework (V2) is no longer supported by the .NET versions of the GUI tools. Only V4 of the .NET Framework is now supported.

1.2 FairCom DB SQL Enhancements

FairCom DB SQL has undergone extensive enhancements in both performance and features. The query optimizer engine underwent a major analysis and many queries now run exponentially faster. Many language constructs were added to make FairCom DB SQL more flexible than ever as well. Additional scalar functions provide convenient operations with data of all types.

New and Enhanced Functionalities

- **Stored Procedures, Triggers, and User Defined Functions are Included by Default**
- **Included Built-In Stored Procedures**
- **Improved Stored Procedure Handling**
- **Automatic Shared Memory Protocol with Local Windows Clients**
- **FairCom Security Handshake for FairCom DB SQL Clients**
- **FairCom DB SQL Client Connection and Login Timeout Support**
- **Advanced Encryption for Tables**
- **A new TRUNCATE Table function**
- **Now Supports Transaction Isolation Levels 1 and 2**
- **Many New Scalar Functions**
- **Advanced Searching of LVARCHAR Fields with the CONTAINS Clause**
- **Complete RIGHT OUTER JOIN Syntax Added**

Performance Enhancements

- **Huge Performance Gains in Complex Query Performance**
- **Query Timeout Option to Prevent Long Running Queries**
- **Diagnostic Logging of Query Times**

Utilities

- **Utility Option to Copy a Database**
- **Types SDK for ISAM Compatibility**

- **FairCom DB ISAM to SQL Migration Toolkit**

Many More ...

Complete Online Documentation

Complete information for all of the updated FairCom DB SQL features are available in the updated manuals available online.

c-treeACE SQL Reference Guide (https://docs.faircom.com/doc/sqlref/)

c-treeACE SQL Java Stored Procedures, Triggers and User Defined Functions Guide

(https://docs.faircom.com/doc/jspt/) c-treeACE SQL ODBC Developer's Guide

(https://docs.faircom.com/doc/odbc/)

c-treeACE SQL JDBC Developer's Guide (https://docs.faircom.com/doc/jdbc/)

c-treeACE SQL ISQL Tools and Reference Guide (https://docs.faircom.com/doc/isql/)

c-treeACE SQL ADO .NET Data Provider Developer's Guide (https://docs.faircom.com/doc/ado_net/)

c-treeACE SQL PHP Developer's Guide (https://docs.faircom.com/doc/php/)

c-treeACE SQL Direct SQL Developer's Guide (https://docs.faircom.com/doc/dsql/)

c-treeDBX Developer's Guide (https://docs.faircom.com/doc/dbx/)

1.3 New Features for FairCom DB ISAM Server

The FairCom DB ISAM Server remains one of the fastest database solutions available, with exceptional control over every aspect of your data. FairCom DB V9 takes scalability and performance to new heights for high end systems. FairCom engineers spent considerable effort in fine tuning and adding advanced improvements to the core multithreaded engine. The result is massive scalability on multi-CPU systems. Testing on 256 CPU machines showed improvements of nearly 100% over V8.14! Put this advanced engineering technology into your next desktop application or enterprise class data warehouse.

Performance

- **Enormous Scalability and Performance with Multi-CPU Systems**
- **Automatic Shared Memory Protocol for Local Clients on Windows**
- **Additional Cache Priming Options**

New Features

- **Quiesce the Server to a Quiet State**
- **Blocking Lock Timeouts**
- **API Call to Change Configuration at Run Time**
- **Rebuild Callback Support for Clients**
- **File Callback Functions for the FairCom DB Server SDK**

Enhancements

- **Enhanced Dynamic Dump Backup Capabilities**
- **More** *SNAPSHOT* **Monitoring Features**
- **Transaction Timeout Feature**
- **Automatic Transaction Processing for NON-TRANPROC Files**
- **Disable the ISAM Communication Port for Security**
- **Default Extended Headers for Enhanced Feature Support**

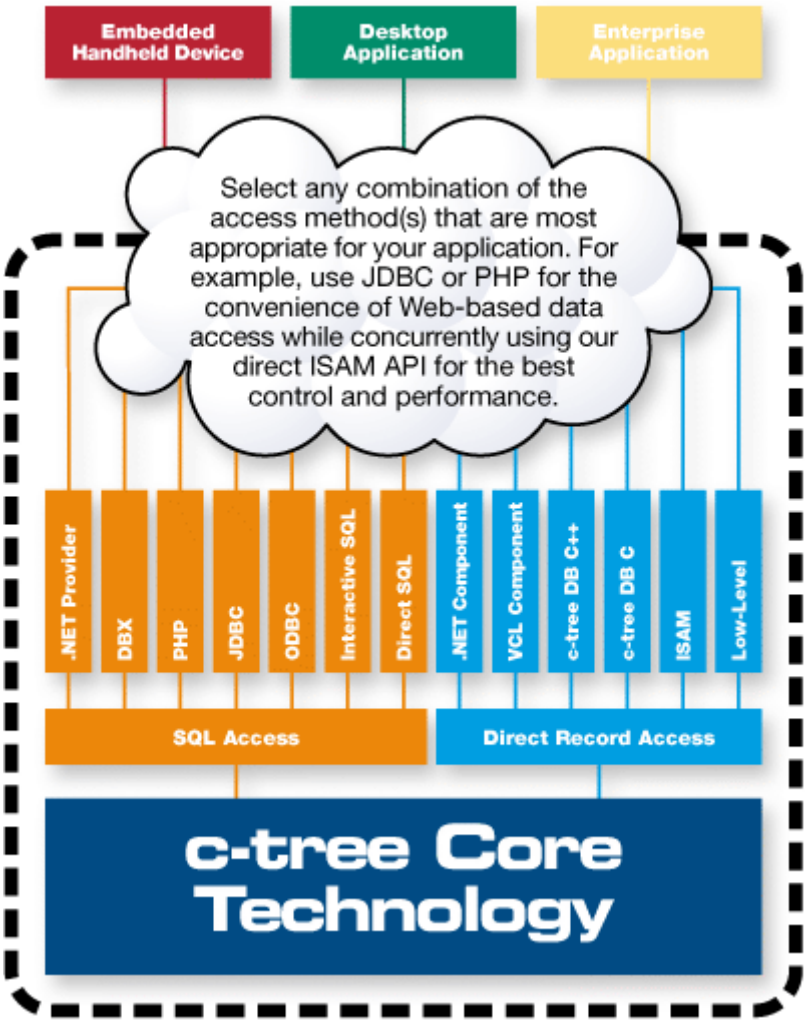
Complete Online Documentation

Be sure to check out the completely updated documentation on our support site.

- c-treeACE Administrator's Guide (https://docs.faircom.com/doc/ctserver/cover.htm)

1.4 Extensive Interface Support

FairCom Database Engine is a unified, multimodel engine with four types of API: Low-Level, ISAM, Navigational "NAV", and SQL. Multiple applications, using any combination of these APIs, can interact simultaneously with the FairCom DB database engine.



FairCom DB V9.0 introduces four new SQL interfaces for even greater choice and control over you application development.

FairCom DB SQL ADO .NET Data Provider

The FairCom DB SQL ADO .NET Data Provider brings the power of ADO .NET into your .NET applications. Quickly connect to your FairCom DB SQL data and build advanced applications with drag-n-drop ease.

The FairCom DB SQL Data Provider seamlessly integrates into your Microsoft Visual Studio 2005 and 2008 environments. With Visual

Studio 2008, put the power of the advanced new LINQ interface into your applications, fully supported by the FairCom DB SQL Data Provider. You will never build database applications the same.

FairCom DB SQL PHP

Today's progressive business environments demand online availability. The method of choice is via the web and PHP has become the language of choice for dynamic web scripting.

FairCom DB SQL is a powerful database technology offering a wide range of interfaces now including PHP support. FairCom's c-treePHP module extends your data access to the world through this popular web interface. c-treePHP modules can be installed with either Apache or Microsoft IIS (Windows) web servers. Quickly design and build web-based applications with direct access to your FairCom DB SQL data.

FairCom DB SQL dbExpress Driver

Borland pioneered the concept of drag and drop ease for data-centric application development. dbExpress is a set of lightweight database drivers that provide fast access to SQL databases.

Make your FairCom DB SQL data available to your next CodeGear application. FairCom DB SQL is directly available to your Borland CodeGear applications with the FairCom DB SQL dbExpress Driver, c-treeDBX. c-treeDBX integrates seamless with the latest CodeGear development environments. Bring advanced CodeGear RAD tools into your next FairCom DB SQL project.

FairCom DB Direct SQL

FairCom DB Direct SQL is the newest FairCom DB SQL interface technology providing a convenient yet programmatic approach to an industry standard SQL interface over your c-tree data. This direct interface combines the robust direct link FairCom DB SQL ODBC technology into an embedded direct programmatic interface rivaling the power of the embedded FairCom DB SQL ESQL interface. Make direct FairCom DB SQL calls into your database directly from your application. The advantage for you is no ESQL precompiling, and no ODBC driver overhead.

SQL Oriented Interfaces

- FairCom DB ADO.NET
- FairCom DB JDBC
- FairCom DB ODBC
- FairCom DB PHP
- FairCom DB SQL Python
- FairCom DB SQL dbExpress
- FairCom DB SQL Direct SQL
- FairCom DB SQL Embedded SQL
- FairCom DB SQL ISQL Interactive SQL
- FairCom DB SQL Stored Procedures

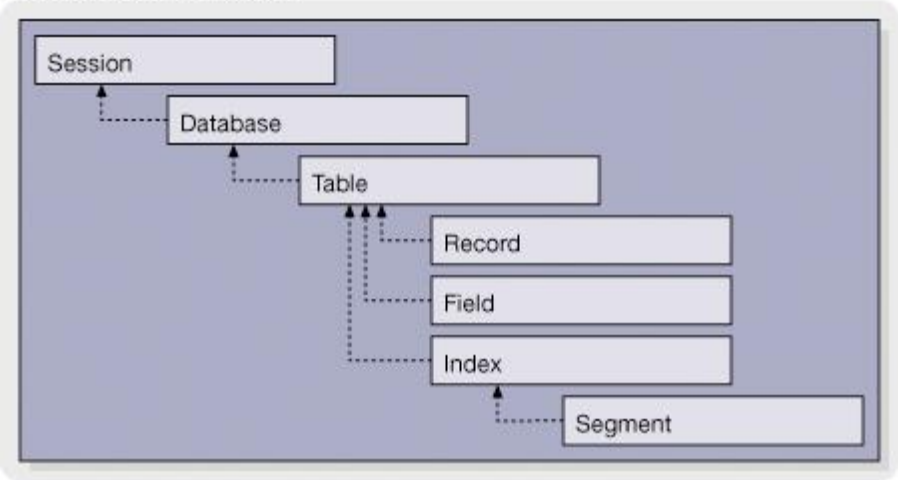
NoSQL Record-Oriented (ISAM) Interfaces

- FairCom DB ISAM C API
- FairCom Low-Level C API
- FairCom DB API .NET C# Database Layer
- FairCom DB API .NET VB Database Layer
- FairCom DB API C++ Database Layer
- FairCom DB API C Database Layer
- FairCom DB API Java Java Database Layer
- FairCom DB API JPA Java Persistence API
- FairCom DB API VCL CodeGear VCL Components

1.5 Additional FairCom DB API and FairCom DB API .NET Features and Improvements

c-treeDB and FairCom DB for .NET continue their growing popularity. Designed for ease of use and quick development, FairCom recommends you consider these great interfaces for any new project. Hiding much of the complexity of c-tree file internals, FairCom DB API and FairCom DB API .NET are easy to get started with yet retain all of the power and performance you expect from FairCom DB.

c-treeDB Relationships



Expanded support for FairCom DB API includes:

- **High Performance Batch Support**
- **Complete FairCom DB Resource Handling**
- **Improved Delphi .NET Compatibility**
- **Extended Unicode Support**
- **Sophisticated Callback Support**
- **Attach and Detach Existing Sessions and Tables**
- **Set Operations Modes**
- **Auto Commit Mode**
- **New Rebuild Functions**

Updated Online Documentation

With nearly 200 new functions and methods added to the C and C++ APIs, FairCom DB API is more flexible than ever for any application. Review the completely updated and revised documentation in the FairCom DB API C and C++ Developer Guides for complete information for all these new FairCom DB API features.

- c-treeDB C API Programmer's Reference Guide (https://docs.faircom.com/doc/ctreedb/)
- c-treeDB C++ API Programmer's Reference Guide (https://docs.faircom.com/doc/ctreepp/)
- FairCom DB for .NET Programmer's Reference Guide (https://docs.faircom.com/doc/nav-dotnet/)

1.6 Advanced Functions for the FairCom DB SDK

When you need control, the FairCom DB SDK should be your choice. Direct calls into the core FairCom DB database engine allow you to take advantage of many high performance options. Partial record rewrites, for example, can gain performance when only updating small portions of a large record. The Quiet and Fileblock functions have many options available to precisely give the functionality you demand. When you need control and performance, consider FairCom DB ISAM technology.

- **Many new API Functions Added**
- **Suspend FairCom DB Operations with QuietCtree()**
- **Block Access to Files with ctFILBLK**
- **Blocking Lock Timeouts**
- **Partial Record Rewrites**
- **List Users Waiting for Locks**
- **Row-Level Permanent Callback Filters**
- **Additional Cache Options for Performance**
- **Recursive Locking Support**
- **Client Connection Timeout Options**
- **Retrieve List of Filenames from FairCom DB**

Updated Online Documentation

- c-treeACE Programmer's Reference Guide (https://docs.faircom.com/doc/ctreeplus/)

1.7 mtPro Build Utility

FairCom DB Professional includes an easy-to-use make utility for configuring your FairCom DB libraries. **mtPro** can be used to build your makefiles and build scripts in an intuitive graphical manner. Simply point and click your way to a quick build of the FairCom DB libraries.

You will find the FairCom DB **mtPro** utility in the *pro/* directory of your FairCom DB installation.

1.8 Easier Navigation in FairCom DB

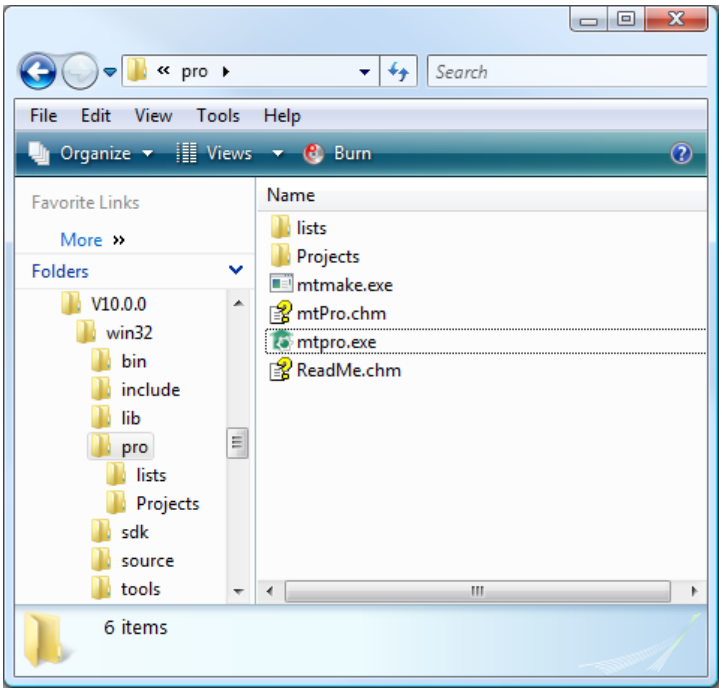
FairCom DB was designed to be easy to use and quick to get started with. As such, the existing directory structure is unchanged in V10.

FairCom DB simplifies your development, including providing the build tools as described above. FairCom DB V10.0 has kept the same structure and layout introduced in V9.

Many c-tree developers are accustomed to building their libraries with the **mtmake** utility and this utility continues to be available. These FairCom DB make utilities are conveniently located in the */pro* directory of your FairCom DB installation. This easy to find location is shown below.

FairCom DB V10 mtmake location

FairCom\V10.0.0\win32\pro



Where do I Go?

A brief description of each of the new directories:

- **bin** - The *bin\ace* directory contains the powerful FairCom DB engine, already started in most Microsoft Windows environments. You will find your data in this area when you begin using FairCom DB. A single directory, *bin\ace\sql*, contains the entire FairCom DB database. Simply copy this directory into your next application deployment. FairCom DB is smaller than 10 megabytes yet rivals the performance of other database products hundreds of times its size!
- **include** - Here you will find all of the c-tree source header files required to build your application. Simply include this folder in your project. Everything is included and ready-to-go.
- **lib** - Each of the c-tree application interface technologies are contained in a separate directory here. With FairCom DB Express the libraries are pre-compiled and ready to go! Libraries are included for most popular programming environments of your chosen platform. For instance, with Microsoft Windows installations, both Visual Studio 2005 and 2008 libraries are included. Simply point your project to the application interface library of choice, link your application, and build.
- **pro -** This directory contains the FairCom DB make utilities for building your own FairCom DB libraries from source. You will find both the traditional **mtmake** utility, with many new updated options, and a new graphical based utility for easily creating new c-tree makefiles.
- **sdk** - This area is where developers can go and get started right away. All of the FairCom DB tutorials for each application interface are contained here, and are ready to load and run. In many cases, simply pick a project, substitute your source code, and you are up and running with FairCom DB database technology in *minutes*.

- **source -** This directory contains all of the traditional FairCom DB client side and standalone source code. Here you can find the source code to many of FairCom's utilities. Modify and customize them as desired to complement your own specific needs.
- **tools** - This area contains the all new sleek and modern FairCom DB tools. Two folders are provided. *cmdline* contains the traditional command line tools, familiar to existing users and retaining the c-tree *admin* and *util* folders. The new *guitools* folder contains the binaries for all of the new FairCom DB tools installed and available from the Windows **Start** menu.

1.9 The Most Up to Date Information

Not only was FairCom's FairCom DB technology updated with many great new features, we were also hard at work updating our documentation to be sure you have the best information available. Be sure to visit the support section of our web site for the most complete and up to date information regarding FairCom DB.

FairCom Home Page - http://www.faircom.com

All of the FairCom DB manuals have been updated to include complete FairCom DB V9.0 information. You are invited to view our latest documentation as you read through this manual for the most complete information.

Release Notes

A complete set of Release Notes detailing specific bug fixes is also available from the documentation portion of our web site. In this document you will find a list of modifications to each core subsystem of FairCom DB technology.

c-treeACE V9.0 Release Notes (https://docs.faircom.com/doc/v103rel/)

1.10 Compatibility Notes

FairCom DB offers significant new features and enhancements. With these new additions come a few points to be aware of when moving from a previous version of c-tree Plus. FairCom strives to maintain backward compatibility whenever possible with existing applications, and FairCom DB V9.0 is no exception: all current data and index files remain completely compatible with no changes. The biggest change is a new directory layout reflecting the ease of use philosophy of the new FairCom DB product. Other changes affect transaction log formats, Unix permissions, and some FairCom DB SQL system table changes. These are all described in the following sections and should be relatively easy to navigate as you upgrade.

Default Extended Headers for Enhanced Feature Support

Extended headers provide support for a range of enhanced c-tree features including

- *HUGE* files
- Six-byte transaction numbers
- Segmented files
- Transaction dependent creates and deletes
- Restorable deletes

This extended support is now enabled by default with FairCom DB for all newly created files, regardless of the function call to create the files. Previously, this mode was only enabled with calls from an *Xtd8* specific function and defining the *XCREblk* structure.

The advantage of this new approach is that 6-byte transaction numbers are used by default, which avoids potential unexpected transaction number overflows, or in some cases, encountering error **R6BT_ERR** (745, *6BTRAN* file required).

This feature can be disabled with the following keyword should this be necessary for backward compatibility:

```
COMPATIBILITY REVERT_TO_V6HDR
```

Standalone applications can disable this support by setting the *cth6flg* global variable to any non-zero value.

Ensure Matching Client and Server Versions for 100% Comptibility

Due to the many new features available in the FairCom DB database core technology it has been necessary to ensure only FairCom DB version 9 clients connect to the FairCom DB database engine for maximum compatibility. To enforce this new restriction, a connection handshake has been introduced. Prior versions of c-tree clients connecting to a new FairCom DB database engine will now receive error **LMTC_ERR** (530, client does not match server).

New Commit Read Lock Support Requires Record Locks for Update

FairCom DB introduces a new feature to prevent “dirty” record reads. Commit read locks enable an implicit, high performance, low-level record lock, ensuring consistent data record reads in high volume transaction environments. This new behavior is on by default. If a record update, under transaction control, is updated or deleted without an explicit lock held, error **CMLK_ERR** (768, commit lock error: make sure record update performed with lock) is now returned.

Please review the section "Commit Read Locks for Guaranteed Data Reads (page 52)" for complete details.

Transaction Log Format

While FairCom always attempts to maintain backward compatibility whenever possible, transaction logs from earlier versions are generally not always compatible with newer FairCom Server formats.

Note: Unless otherwise mentioned in the version-specific Update Guides, existing data and index files are usually not affected by transaction log changes.

Removing Transaction Logs and Upgrading the FairCom Database Engine

The FairCom process makes it easy to upgrade the FairCom server using your existing files, as long as you remove prior transaction logs in a safe manner. The steps shown below are appropriate any time you are upgrading. Notice that you will shut down your server two times during this process (steps 2 and 5) to allow all files to be brought to a consistent state.

1. Have all clients cleanly exit from your existing FairCom server.
2. Perform a normal **controlled shutdown** of the server using one of the methods described here, depending upon your installation:
   - Server Console Window - From the FairCom Server console window click “Control” and then click “Shutdown the Server”
   - Windows Toolbar - Right-click the c-tree Server icon in the Windows Tooltray and choose “ShutDown the Server”
   - Windows Service - From the Windows Control Panel, choose “Administrative Tools”, then choose “Services”. Locate the **FairCom Server** in the list of services running on your machine. Right-click the c-tree Service and choose “Stop”.
   - Use the client command line utility, **ctadmn**, and follow the prompts.
   - Use the client command line utility, **ctstop**.

   Remember that the Administrator user ID is "admin" (case insensitive) and the default password is "ADMIN" (case sensitive). The default c-tree Server name is "FAIRCOMS".
3. Block the ability of any clients to attach to the FairCom server.
4. Restart the **existing** FairCom server with no clients attached and allow a successful automatic recovery to take place. This ensures all files are brought to a consistent state in the event there is any data remaining in the transaction logs.
5. Perform another normal **controlled shutdown** of the FairCom server as described earlier.
6. Remove all existing transaction logs and associated files (*L*.FCS*, *S*.FCS*, *D*.FCS*, *I*.FCS*, and **.FCT*).

> **Note**: We do not recommend removing *FAIRCOM.FCS* unless specifically instructed to do so in the notes accompanying the new FairCom version.

Several other types of log dependencies should be considered before deleting transaction logs:

1. Replicas
2. Deferred indexes
3. Record update callbacks

If these options are in use, ensure they have processed all the log data before the logs are deleted. Otherwise the dependent data could be out-of-sync. (2 & 3 don't always use the transaction logs)

7. Copy your new FairCom server directory in its entirety into the existing Server directory. Note you might want to protect your existing *ctsrvr.cfg* and *ctsrvr.set* files so you don't loose any custom settings. Also review the new *ctsrvr.cfg* file accompanying any FairCom server upgrade to leverage new best practice settings.

> **Note:** Client compatibility can prevent connections to the new FairCom Database Engine. It is always advised to use the most recent matching client version with your FairCom server version.

8. Unblock the ability of any clients to attach.
9. Start the FairCom server in your usual manner and begin using your existing data.

FairCom has added logic to notify you when transaction logs may be incompatible. Please review the section "*Detection of Transaction Log Incompatibilities*" in the *FairCom Server Administrator's Guide* (https://docs.faircom.com/doc/*ctserver/*) for details.

HUGE Files are now Default with FairCom DB SQL

c-tree supports files larger than 4 GB. These are known as *HUGE* files and require additional attributes at file creation. FairCom DB SQL previously created all tables as standard c-tree data files which, by default, were not *HUGE*.

The new FairCom DB SQL engine now creates all tables as *HUGE* allowing nearly unlimited amounts of data to be stored. For backward compatibility, however, it is possible to revert to the original FairCom DB SQL behavior by adding the following server configuration keyword to *ctsrvr.cfg*:

```
SQL_OPTION NO_HUGEFILE
```

Backward Compatibility Changes with c-treeSQL Databases

FairCom DB SQL V9.0 introduces many new changes. A compatibility change with previous c-treeSQL databases exists with the system tables. 64-byte identifiers are now used whereas, previously, 32-bytes was the limit. While FairCom DB SQL can function on tables previously created with this 32-byte limit, you will not be able to increase the identifiers (including new ones) to 64-bytes as your existing system tables will not accommodate the larger size. Contact your nearest FairCom office if you need assistance in migrating your databases to the new format.

LOG_WRITETHRU support for Unix

FairCom DB now supports the server configuration keyword `COMPATIBILITY LOG_WRITETHRU` option on Unix systems. `COMPATIBILITY SYNC_LOG` is now considered a legacy keyword and FairCom recommends using `COMPATIBILITY LOG_WRITETHRU` instead. This change provides consistency of options between the Windows and Unix versions of FairCom DB configuration options.

Note: On the Solaris operating systems, `COMPATIBILITY LOG_WRITETHRU` uses O_DSYNC synchronous writes for the transaction logs when the `COMPATIBILITY SYNC_LOG` or `COMPATIBILITY LOG_WRITETHRU` configuration options are specified in the server configuration file. A new c-tree Server configuration option, `COMPATIBILITY DIRECT_IO`, can be used to revert to the previous behavior of using direct I/O.

New Unix Default File Permissions Mode

Previously on Unix systems, files were created with full permissions (file mode 0666). When creating new files, FairCom DB now defaults to a permission mode of 0660 (read/write access for owner and group; no access for world).

Please see the section “File Permission Mode for Files Created by c-tree on Unix Systems” for complete information concerning this change.

Record Based c-treeDB Filters

c-treeDB filters are now record based, rather than table based. Originally, once such a filter was activated, all ISAM contexts would share the same condition. This new approach avoids some unexpected behaviors observed with the previous implementation.

The section “c-treeDB Filters are Now Record Based, Rather Than Table Based” has complete details.

c-treeSQL and c-treeDB BINARY and VARBINARY Compatibility Issues

The V8 series of c-treeSQL Servers could conflict in the handling of *CT_ARRAY* field types when accessed by both c-tree ISAM and c-treeSQL. The following example illustrates the complexity of the issue.

Consider a simple c-tree Plus ISAM table with a *CT_ARRAY* field of 16 bytes imported into a c-treeSQL database. It was noted that the same table created with a simple c-tree Plus ISAM API created the *CT_ARRAY* field with 16 bytes (the *DODA* entry of the *CT_ARRAY* field had a length of 16) while an equivalent table created with c-treeSQL (using ISQL for example), created the table with a 20 byte *CT_ARRAY* field. **Queries on this table can produce a c-treeSQL Server crash.**

The c-treeSQL Server handled `BINARY` (mapped as *CT_ARRAY*) and `VARBINARY` (mapped as *CT_2STRING*) fields differently from c-treeDB. c-treeSQL maintains four bytes prepended to the data area to indicate the length of the field, however, c-treeDB did not require any length information about the data, and did not maintain this same four byte addition.

Furthermore, *CT_2STRING* field handling was even more complex as there already existed two bytes for the *CT_2STRING* field length, followed by the additional four bytes added by c-treeSQL.

For a customer, there are three potential situations that exist:

1. If a customer is using only c-treeSQL interfaces, ODBC, JDBC, ISQL, etc., the *CT_ARRAY* and *CT_2STRING* fields contain an extra four bytes at the beginning indicating how many bytes follow. As long as only c-treeSQL interfaces are used to create and maintain the tables, there are no issues.
2. If a customer is using only c-treeDB or c-tree Plus ISAM interfaces, the *CT_ARRAY* and *CT_2STRING* fields do not contain the extra four bytes at the beginning to indicate how many bytes follow. As long as c-treeSQL interfaces are **NOT** used to access these tables, there are no issues.
3. If a customer mixes c-treeSQL and c-treeDB operations in the same table, there exists a compatibility issue, possibly resulting in a c-treeSQL Server crash. Fields written by c-treeSQL and read by c-treeDB are safe: the data simply contains an extra four bytes at the start which must be accounted for at the application level. **Fields written by c-treeDB and read by c-treeSQL are not safe: the first four bytes of the data are interpreted as a four byte integer by c-treeSQL indicating how many bytes will follow.** This scenario will mostly likely result in a c-treeSQL Server crash with a memory violation exception.

Note: For V8 c-treeSQL installations, it is possible to design a c-treeDB or c-tree Plus application to properly account for this four byte discrepancy and still allow c-treeSQL to function properly over the tables.

This requires the tables to be created in one of two manners. The tables can be created with the extra four bytes of storage at the beginning of the *CT_ARRAY* or *CT_2STRING* fields with a non-c-treeSQL interface, and then import them into a c-treeSQL database. Alternatively, create the tables directly with a c-treeSQL interface.In either case, it is the application's responsibility to determine and maintain the field length in the allotted space. However, this places a burden on the application developer with a dependency that is not generally expected.

Addressing the issue

The permutations of tables created by c-treeSQL and c-treeDB made this an extremely complex issue. There is no absolute way to detect that a *CT_ARRAY* or *CT_BINARY* field is written by c-treeSQL or by c-treeDB. It is important to note that this issue exists only for c-treeSQL users when *CT_ARRAY* or *CT_2STRING* columns are used interchangeably with non-c-treeSQL API interfaces.

This complex handling of fields was addressed with a number of different steps:

1. An additional c-treeDB resource member was introduced to mark *CT_ARRAY* and *CT_BINARY* fields.
2. Field handling was modified in c-treeSQL Version 9.0.
3. The c-treeSQL Import utility was modified to touch c-treeDB resources for imported tables.
4. Utility programs were created to identify and mark tables with potential issues.

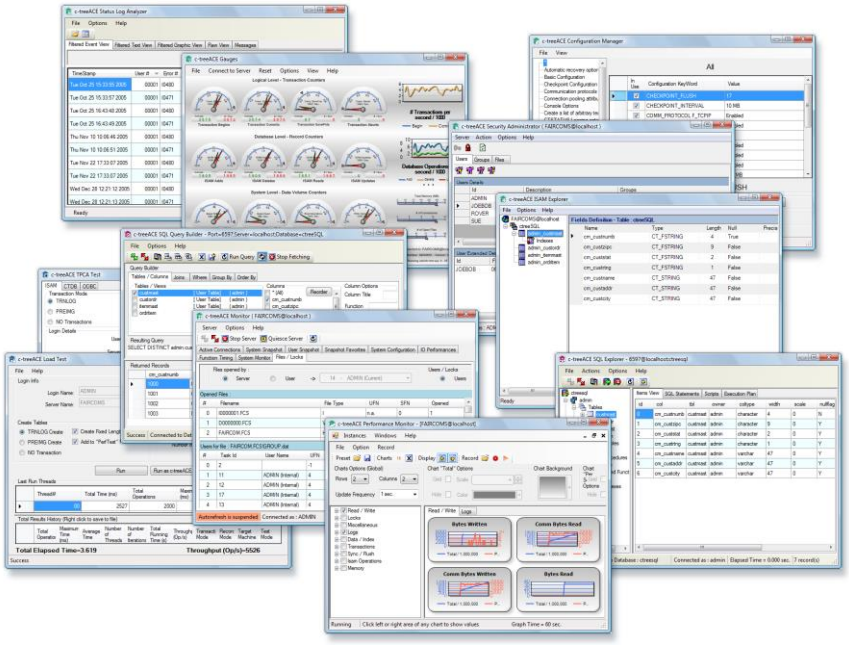
NULL Handling in Filter Expressions

Prior to V9, records values with the NULL attribute set are returned even when a filter was in effect. Beginning with V9, these records do not pass this filter condition. V9 now checks if a field is NULL and treats a NULL value as distinct from a value of 0.

2. FairCom DB Tools

Introduction

FairCom has extended the simplicity of FairCom DB with a set of tools to enable you to work productively and intuitively. Everything you need, ranging from table and data management to monitoring performance, to administering your users and security, is quickly at your fingertips. See how FairCom DB brings simplicity and control to your application development.

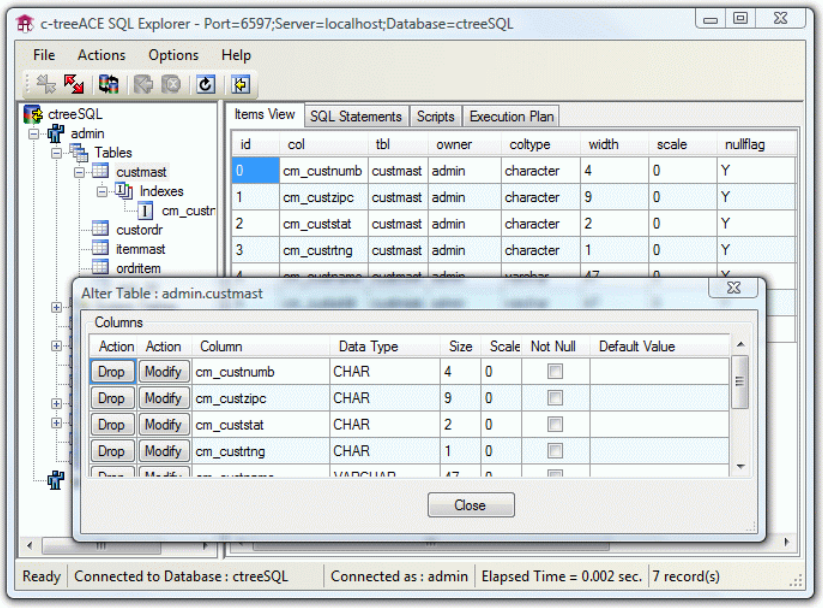


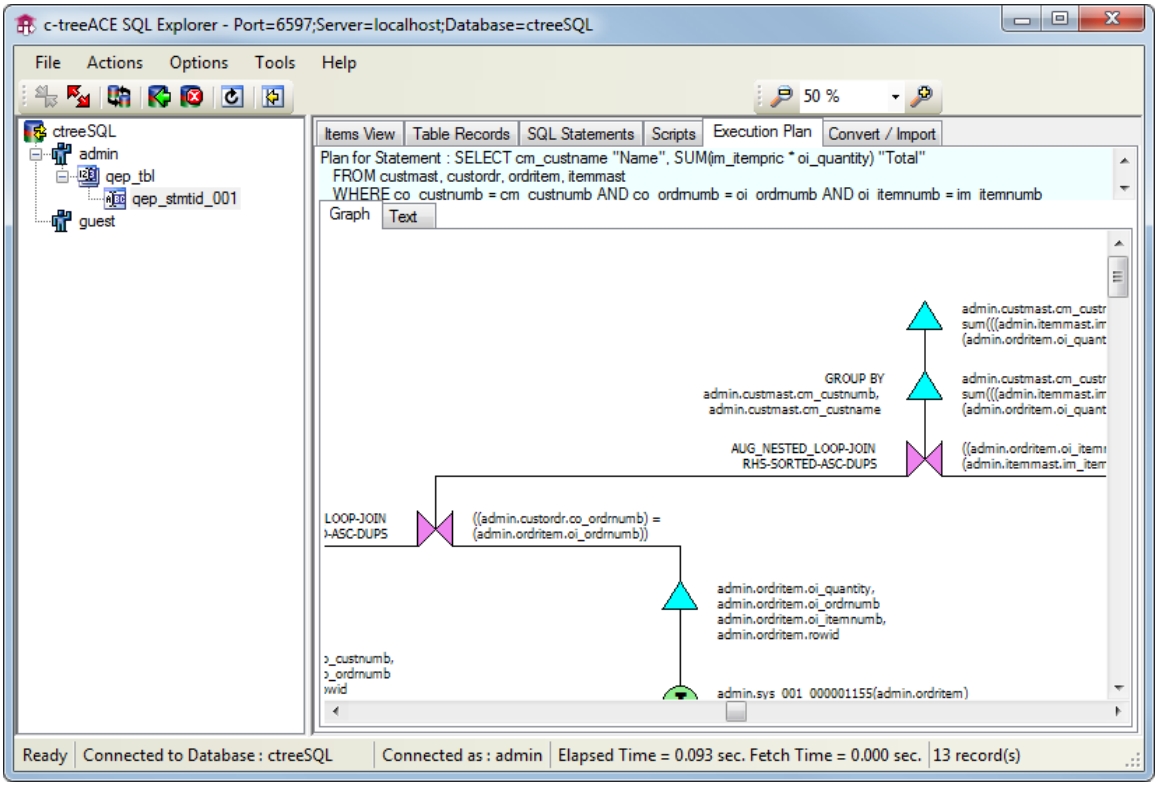
2.1 FairCom DB SQL Explorer

The FairCom DB SQL Explorer is your “one-stop” utility to view and manage your FairCom DB SQL tables and data. Constructed with the database administrator in mind, the FairCom DB SQL Explorer provides for nearly every conceivable need. Here is a brief list of possibilities:

- **Execute Custom FairCom DB SQL Statements**
- **Load, Edit and Run FairCom DB SQL Scripts Interactively**
- **View query execution plans to profile and examine complex queries for optimization analysis**
- **Create and Manage Users**
- **Create and Manage Databases**
- **Create, Alter and Drop Tables and Indexes**
- **Create and Drop Views**
- **Create and Drop Stored Procedures and Triggers**
- **Create and Drop Synonyms**
- **Export Schemas**
- **And Much More**

Most operations can be instigated with a simple click of the mouse.

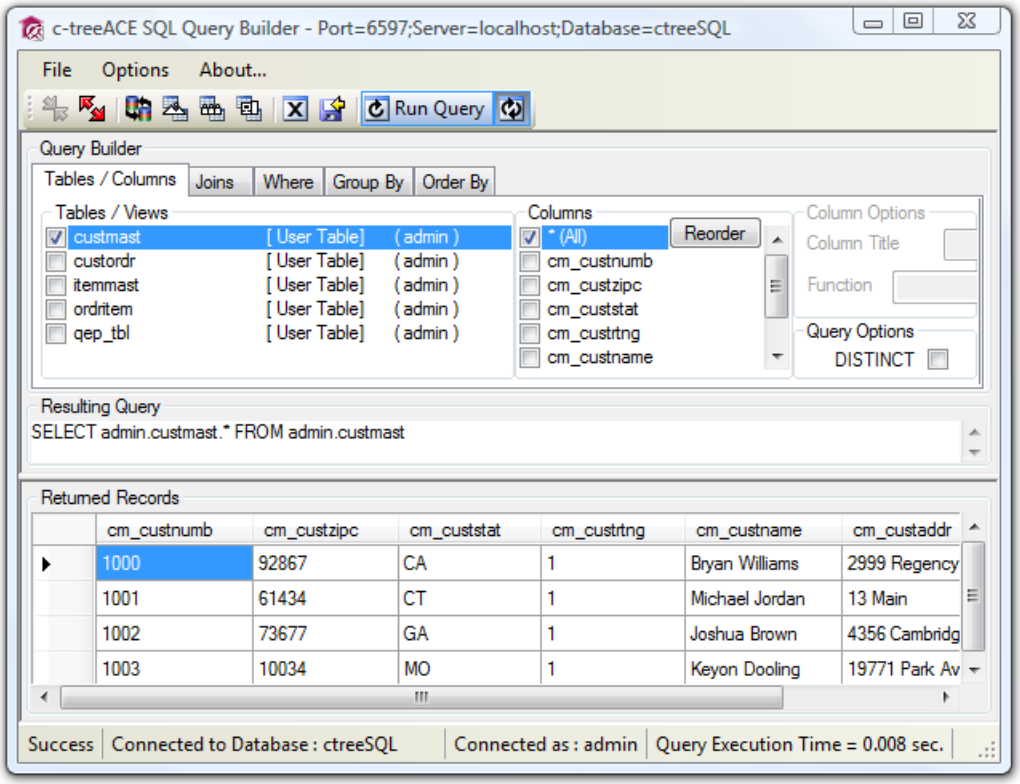




2.2 FairCom DB SQL Query Builder

FairCom DB SQL Query Builder brings the construction of queries to simple point and click ease. Design and test your query options before you include them in your application. Create joins over multiple tables within your database and instantly view the resulting SQL command generating the desired result set. It is easy to browse your results after you run your queries.

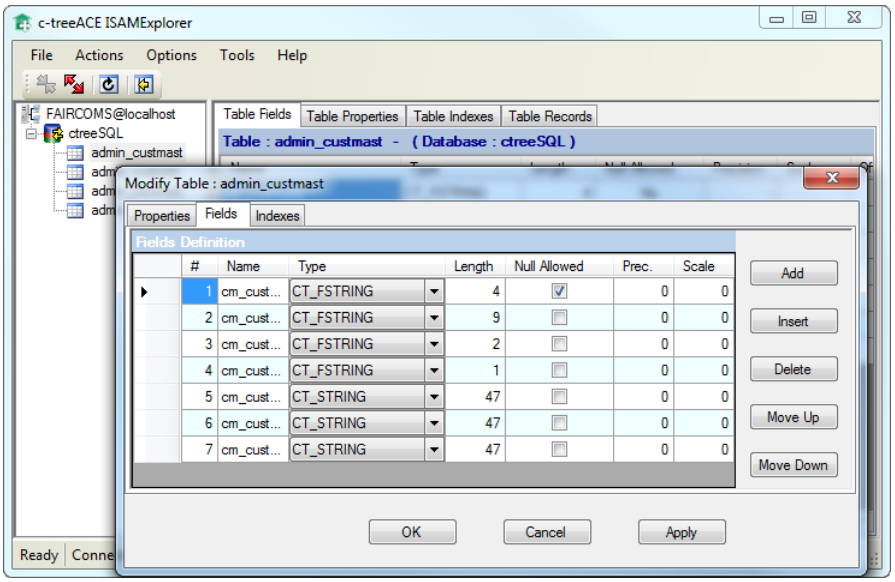
- **Quickly Build and Test Complex Queries**
- **Dynamic Selection of Tables and Columns**
- **Easy Joins and with Drop Down Selections**



2.3 FairCom DB ISAM Explorer

FairCom DB ISAM Explorer is an exciting client tool offering a graphical interface for rapid database development using c-treeDB methodology. Building and managing your FairCom DB tables has never been easier. With FairCom DB ISAM Explorer, you are just a mouse click away to:

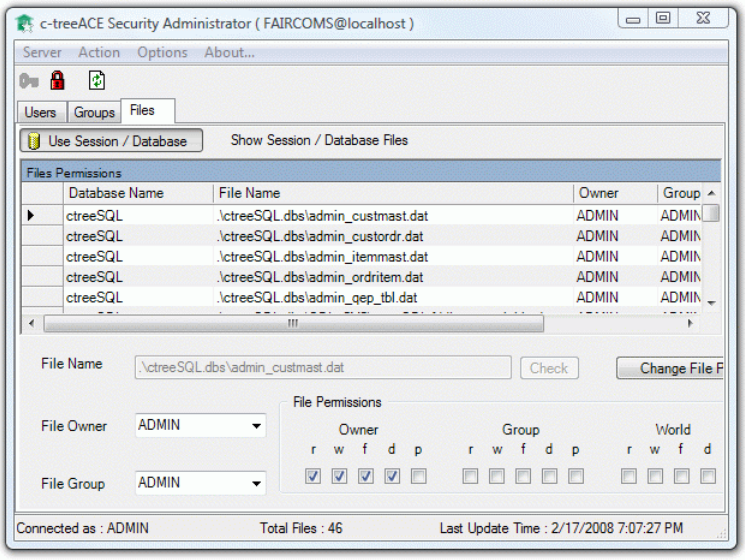
- **Create and Drop Databases**
- **Create, Alter and Drop Tables**
- **Create and Drop Indices**
- **Add Existing Files**
- **Browse Data**

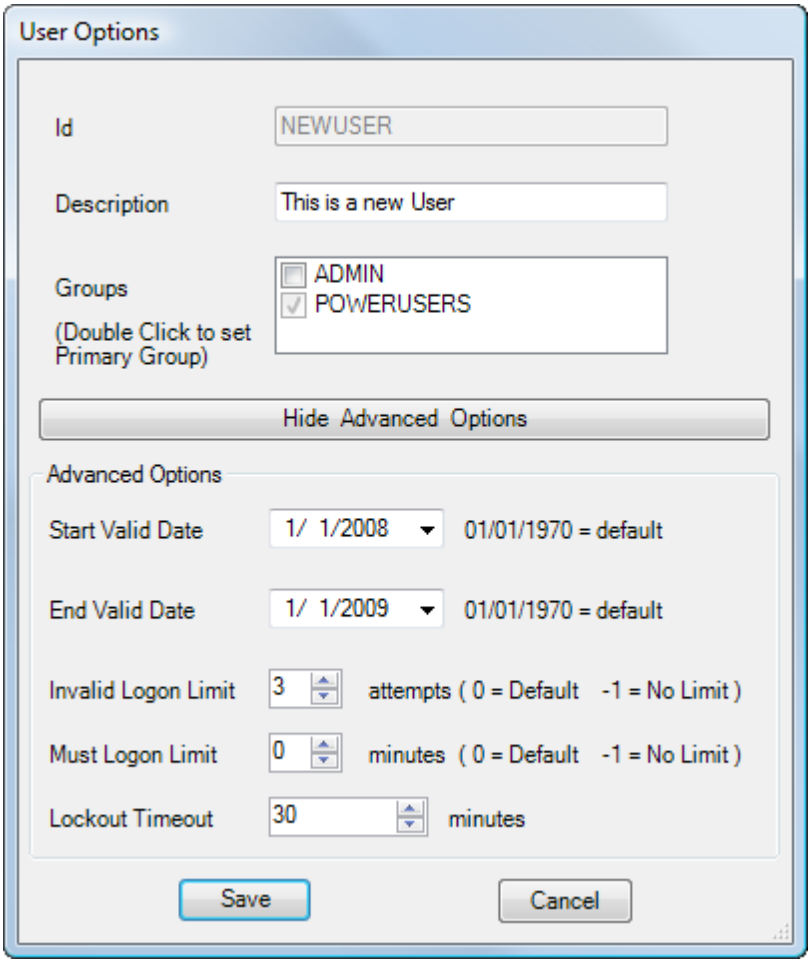


2.4 FairCom DB Security Administrator

FairCom DB Security Administrator has been designed to manage FairCom DB fast and intuitively. No more time lost wading through command lines and menus. Every operation is a right-click away at your finger tip.

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

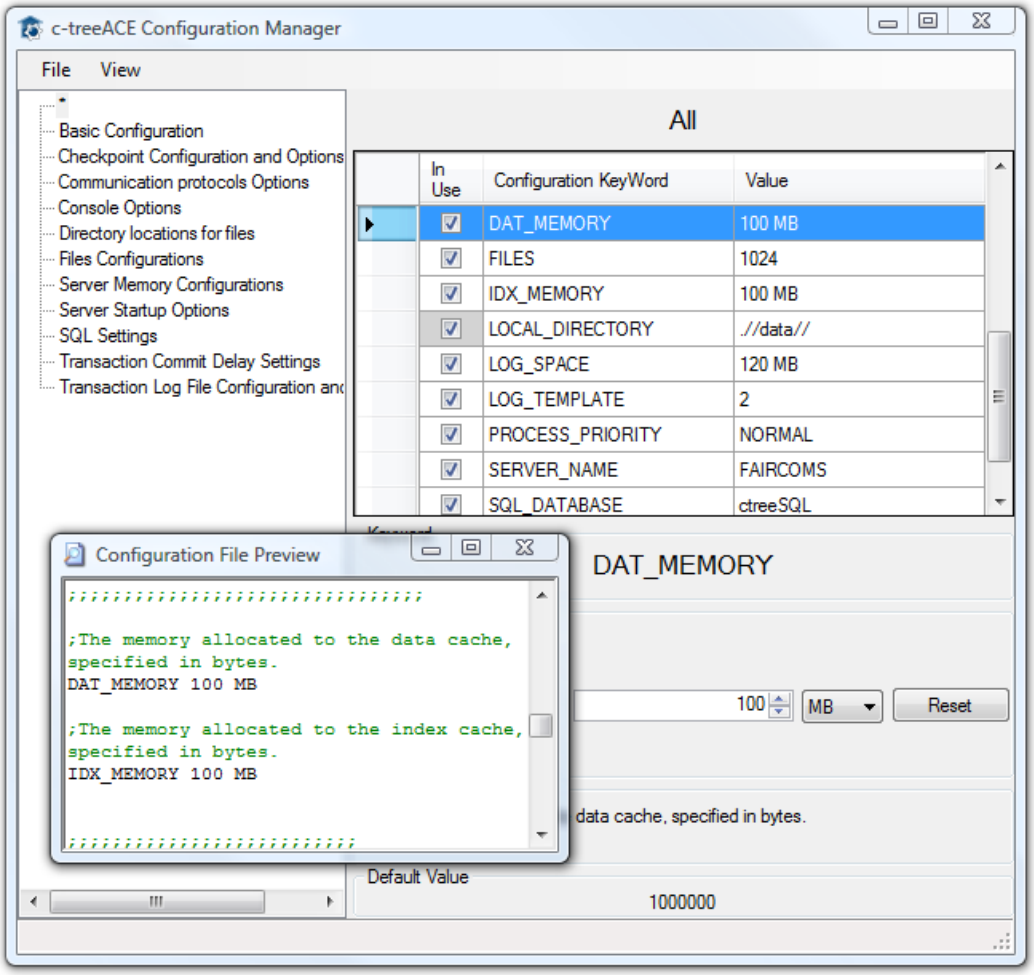




2.5 FairCom DB Configuration Manager

FairCom DB Configuration Manager provides an easy way to manage and keep track of the multitude of FairCom DB configuration keywords. Sorted by category, you can now quickly find and specify the exact keyword and syntax required for your FairCom DB configuration. Quickly test changes to your configuration for optimal FairCom DB performance.

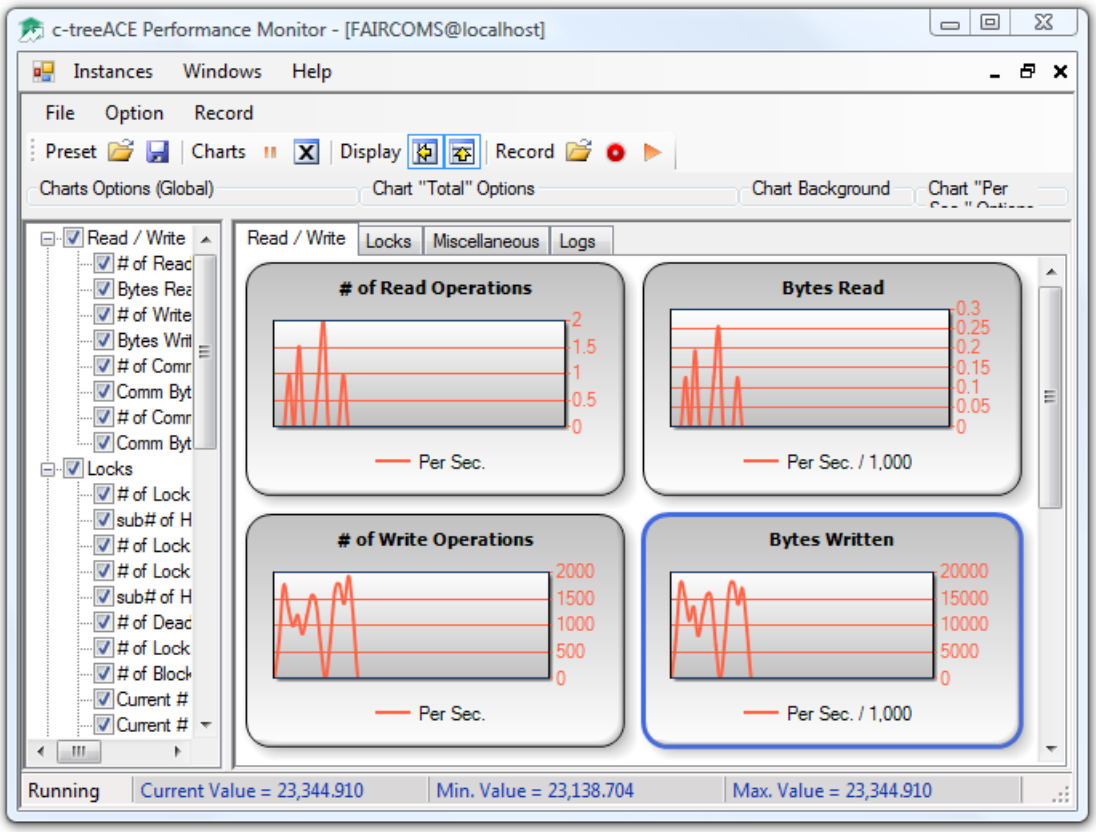
- **Categorical Arrangement of Keywords**
- **Drop Down Selection of Allowed Values**
- **Preview of Final FairCom DB Configuration File**



2.6 FairCom DB Performance Monitor

The FairCom DB Performance Monitor graphically displays critical FairCom DB operational parameters in real time enabling you to quickly spot performance bottlenecks. Sorted by functional categories, FairCom DB Performance Monitor 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.

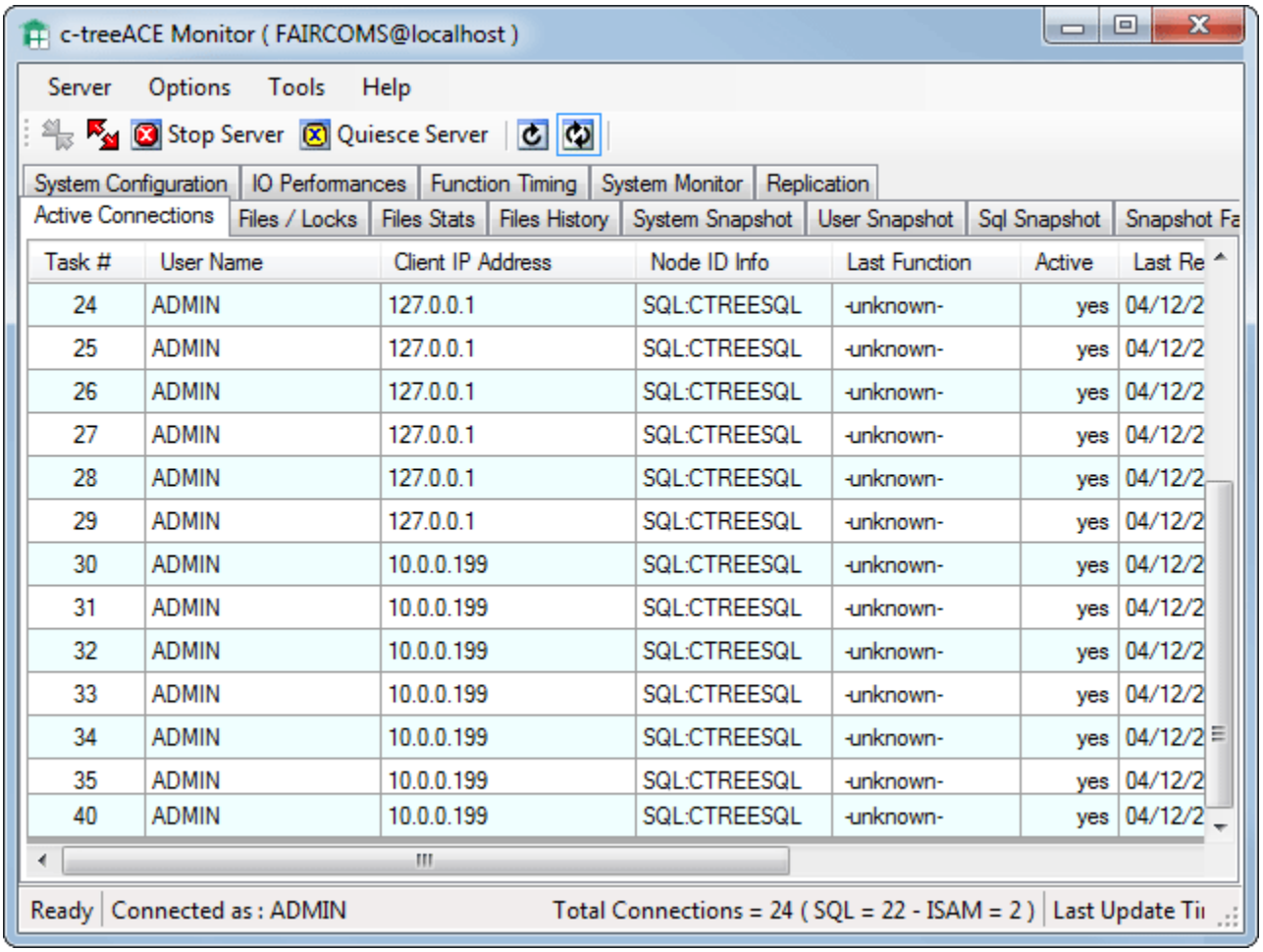
- **Visual Display of Statistics**
- **Categorized SNAPSHOT Metrics**
- **Save Your Monitoring Configuration**
- **Record Periods of Monitoring for Later Analysis Playback**
- **Completely Customizable Monitoring Windows**

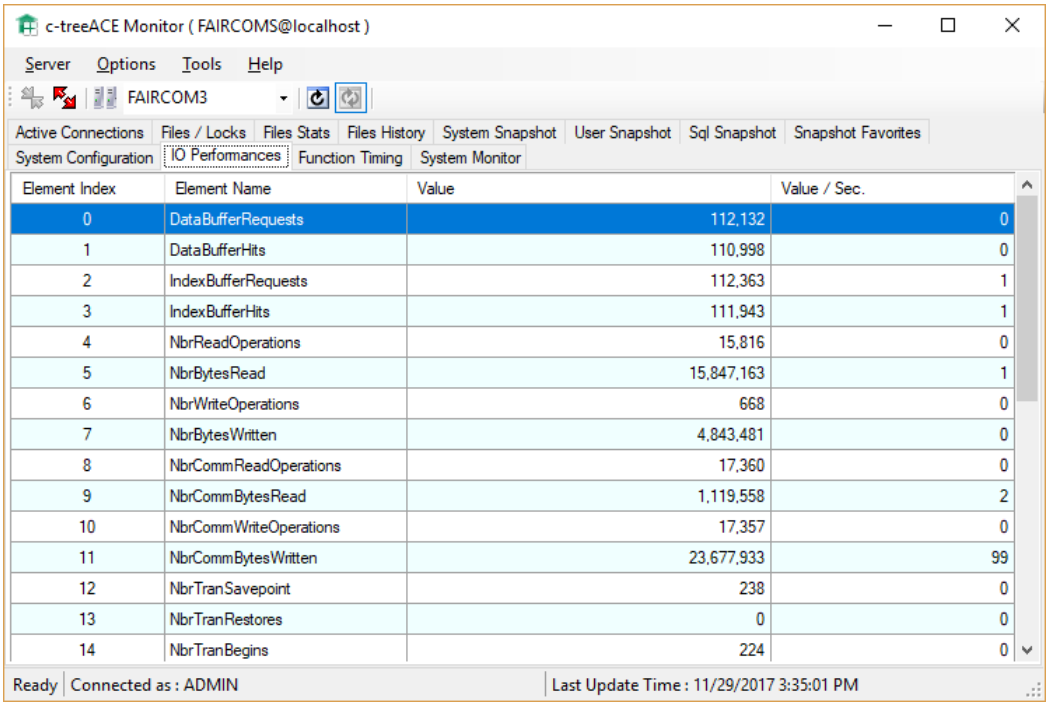


2.7 FairCom DB Monitor

FairCom DB provides a wealth of statistics for performance monitoring. While these are invaluable for automated monitoring systems (for example, IBM's Tivoli system) to gather and analyze, it is useful to quickly view these statistics at any time. The FairCom DB Monitor tool captures and organizes these statistics in real-time for any FairCom DB administrator to observe. FairCom DB Monitor also performs these tasks:

- **Collect Function Timing Statistics**
- **Advanced File Usage Statistics**
- **Quiesce FairCom DB**
- **View Current Configuration Options**
- **Disconnect a User**

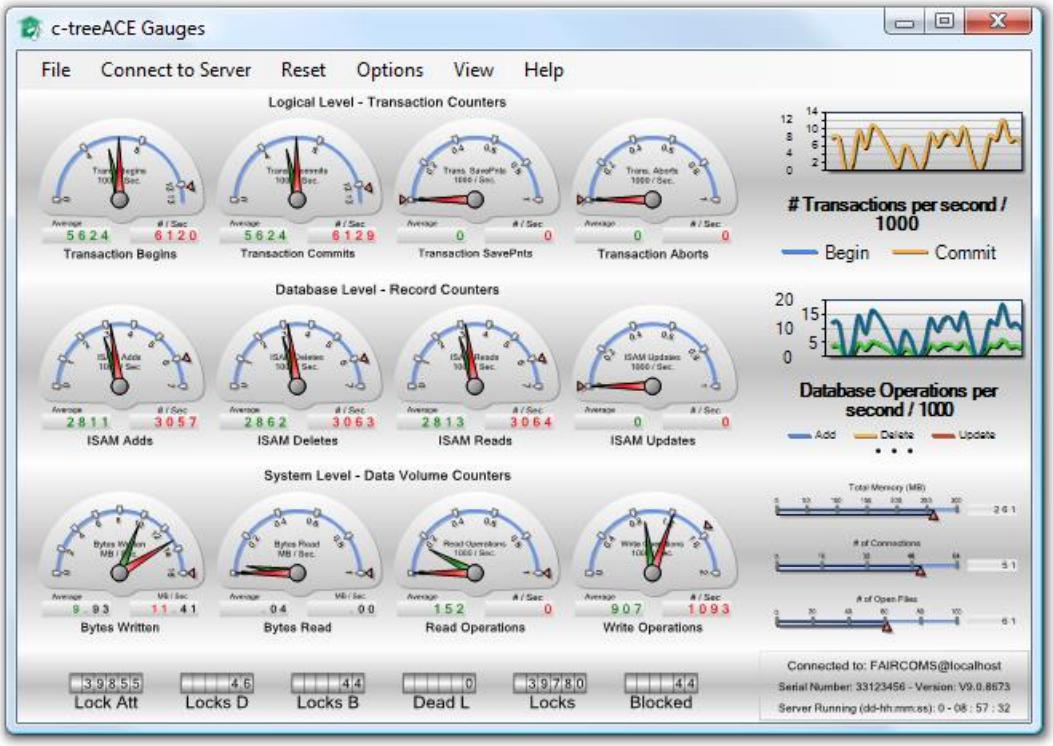




2.8 FairCom DB Gauges

FairCom DB Gauges bring a unique view to the inner operations of FairCom DB. Much like the dashboard of your automobile, FairCom DB Gauges graphically depicts valuable FairCom DB performance metrics as a collection of real time gauges. While not intended as a full monitoring solution, FairCom DB gauges make it easy to begin understanding the internal workings of FairCom DB.

- **Two Views to Examine Real Time FairCom DB Operations**
- **Cumulative and Average Values Maintained for Comparison**

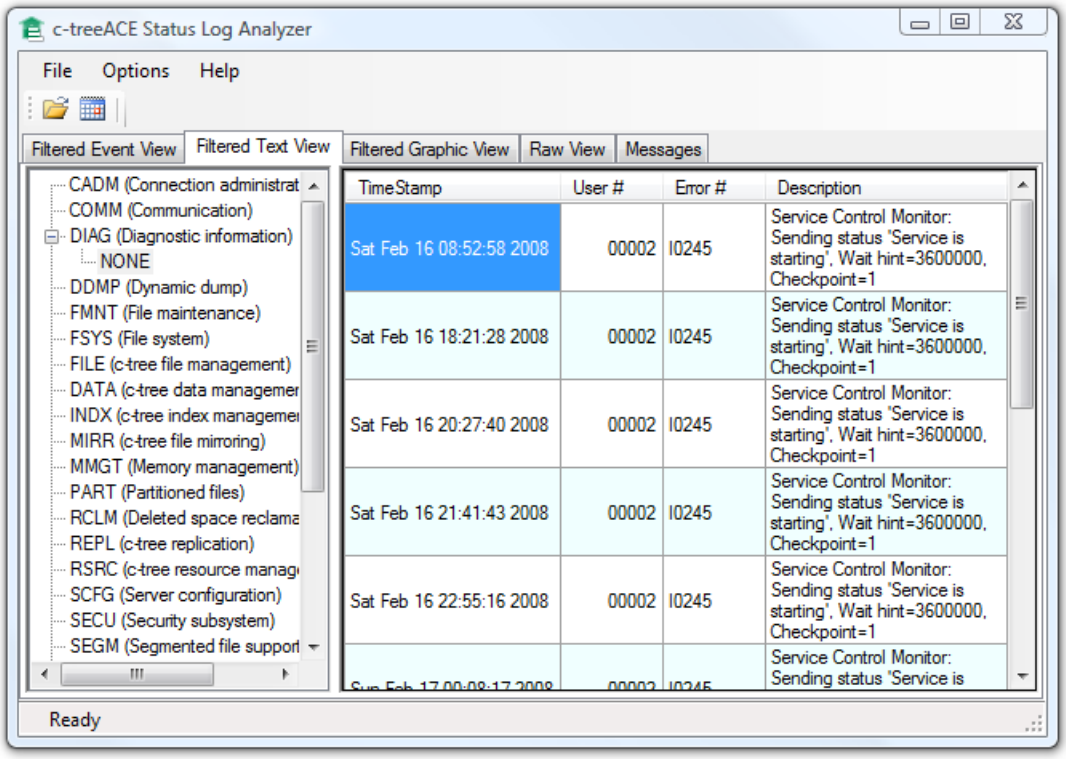



c-treeACE Gauges
File
Connect to Server
Reset
Options
View
Help
Transaction Counters (Logical Level)
Begins
Commits
Aborts
SavePnts
Record Counters (Database Level)
Reads
Adds
Updates
Deletes
Data Volume Counters (System Level)
Bytes Read
Read Operations
Bytes Written
Write Operations
Communication Counters (c-tree Level)
Comm. Bytes Read
Comm. Read Ops.
Comm. Bytes Written
Comm. Write Ops.
Log Operation Counters (System Level)
Log Write Ops.
Log Bytes Written
Log Read Ops.
Log Bytes Read
Log Flush Ops.
Log Ext. Ops.
Data/Index Caches (c-tree Level)
Data Buffer Reqst
Index Buffer Reqst
% Data Cache Used
% Index Cache Used
% Data Cache Hits
% Index Cache Hits
Sync/Flush Operations (System Level)
c-tree File Sync
Disk Sync
Tran Log Sync
Tran Logkbs Sync
Begin Tran Flush
Comm/Abort Flush
LOCK Operations (System Level)
Locks Held
Blocked Relod.
Locks Blocked
Lock Attempts
Locks Freed
Locks Denied

2.9 FairCom DB Status Log Analyzer

FairCom DB records information about activity and problems in the text readable status log file *CTSTATUS.FCS*. This is the place where you should look first anytime you are tracking down an issue with FairCom DB. FairCom DB Status Log Analyzer makes it easy to view and analyze this valuable file. Sorting entries into operational categories, FairCom DB Status Log Analyzer then color codes which entries require attention with intuitive green, yellow and red.

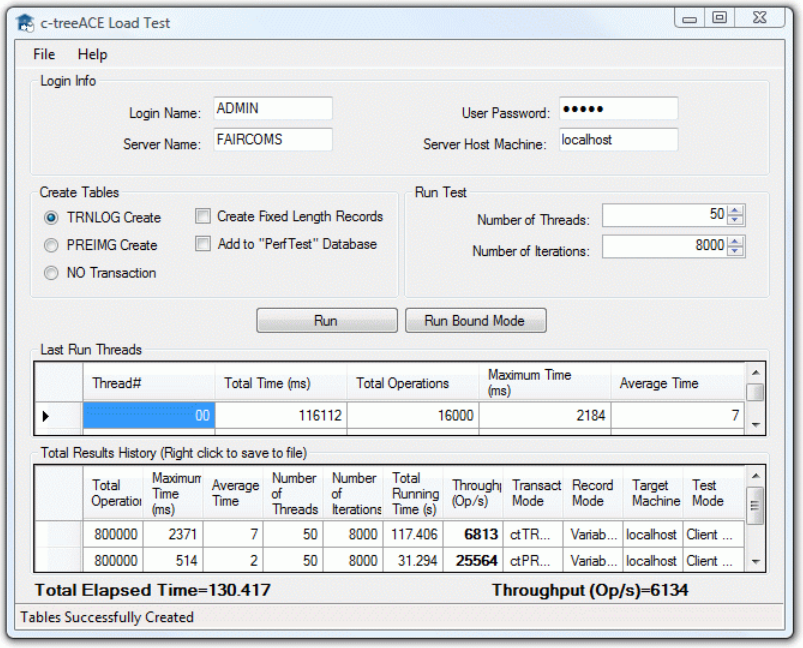
- **Sorted Entries by Operational Category**
- **Color Coded Messages with Critical Entries Highlighted in Red**



2.10 FairCom DB Load Test

FairCom DB Load Test brings the same performance tests used in FairCom's QA and analysis labs to your desktop. Quickly test and analyze throughput on specific machines and configurations. With multiple options to simulate various data streams, you can easily pinpoint performance bottlenecks. Combined with the FairCom DB Performance Monitor and FairCom DB Monitor utilities, you can directly observe FairCom DB performance on your exact system.

- **Specify Multiple Threads of Operation**
- **Selectable Transaction Processing Levels**
- **Complete Display of Test Results for Each Run**



3. FairCom DB SQL Enhancements

Introduction

FairCom DB SQL is faster than ever. The query optimizer has undergone extensive analysis and revision to enhance performance. The SQL syntax has enjoyed numerous additions. Many additional scalar functions have been added for even more flexibility in your SQL database design. A major enhancement has been the merging of the previously separate c-treeSQL Servers into a single engine: the c-treeSQL Server and the c-treeSQL Server Java Edition. The result is FairCom DB SQL with dynamic support for Java stored procedures, triggers and user defined functions deployable in a small easy to package footprint. One engine -- one solution.

And don't forget, FairCom DB SQL is a superset of the FairCom DB ISAM Server. Thus all of the enhancements made to the FairCom DB ISAM Server are also available with FairCom DB SQL Be sure to review the FairCom DB ISAM Server chapter and discover all of the latest benefits available for you.

3.1 Improved Query Optimizer Performance

The FairCom DB SQL Optimizer underwent extensive testing and analysis with the result being dramatically improved performance in many complex queries. A brief list of the major improvements are listed here.

- **Improved Performance of MAX and MIN Operators**
- **Improved Performance of the BETWEEN Operator**
- **Improved Performance with Particular Predicate Restrictions**

3.2 Stored Procedures, Triggers, and User Defined Functions Now Standard Features

Designed around simplicity, FairCom DB SQL is now a complete package for your entire SQL development. Java based stored procedures, triggers and user defined functions used to require a separate server package, the c-treeSQL Server Java Edition. Now, these powerful features are standard equipment. One engine provides all. If your specified Java environment is available, FairCom DB SQL will load it. No more messing around with separate installation packages.

3.3 Advanced Encryption for FairCom DB SQL Tables

FairCom DB SQL supports the ability to encrypt tables from the `CREATE TABLE` command. Advanced encryption, including standard AES (Rijndael), Twofish, Blowfish, and DES is available for industry-standard hardening of data. The AES encryption standard may be required with some forms of data, such as that used in the health care industry (such as to conform to U.S. HIPAA regulations) and the financial industry.

FairCom CAMO is also available. CAMO or "Camouflage" is an older, legacy method of hiding data, which is not a standards-conforming encryption scheme, such as AES. It is not intended as a replacement for Advanced Encryption or other security systems.

3.4 Default HUGE Files for Tables

HUGE files (c-tree files larger than four gigabytes) are frequently encountered with today's massive storage needs. To accommodate these increased table sizes, FairCom DB SQL now creates files as *HUGE* by default. Before this change, FairCom DB SQL created new tables as standard c-tree files (not *HUGE*). An optional `STORAGE_ATTRIBUTES` clause containing `'HUGE'` was available to create FairCom DB SQL *HUGE* tables.

Reversing the previous behavior, the `STORAGE_ATTRIBUTES` clause now supports a `NOT HUGE` option to create standard size c-tree tables.

3.5 Quickly TRUNCATE Tables

FairCom DB SQL supports a new feature to quickly delete all the rows of a table. Execute the `TRUNCATE TABLE` command to delete all the rows of a table in a single action. `TRUNCATE` is now on the list of FairCom DB SQL reserved words.

3.6 Maximum Field Lengths for Non LONGVAR Fields Raised to 8K

The maximum FairCom DB SQL field length has been increased to 8,192 bytes. The maximum field length in V8.14 was 2000 bytes. Any field requiring more than 2000 bytes required either a

LONGVARCHAR or a LONGVARBINARY type. These long types presented challenges in handling as they have limitations including the impossibility of defining indices on them. The long types also preclude the ability to easily access data as they require particular handling when using ODBC and JDBC.

Note: If you add a field with the maximum size, you may be unable to create an index on the field unless you raise the server PAGE_SIZE setting as the key may be larger then the index node size.

3.7 Additional Scalar Functions Available

FairCom DB SQL now supports additional scalar functions providing your SQL applications greater flexibility.

Function	Description
COT	Returns the cotangent of the expression.
CURRENT_DATE	Returns the current date as a DATE value.
CURRENT_TIMESTAMP	A synonymous replacement for SYSTIMESTAMP
CURRENT_USER	Returns the string identifier of the database user.
EXTRACT	Returns a date time field from a date time expression.
LOCALTIME	Returns the current time as a TIME value.
LOCALTIMESTAMP	Returns the current data and time as a TIMESTAMP value.
LOG	Returns the natural log of an expression.
OCTET_LENGTH	Returns the number of bytes in a string.
OVERLAY	Replaces characters in a string.
POSITION	Returns the first occurrence of a character in a string.
SESSION_USER	Returns the value of the FairCom DB SQL session identifier.
SUBSTRING	Returns the substring of a character string.
TRIM	Removes leading and/or trailing characters from a string.

3.8 Advanced Searching with CONTAINS Clause and LVARCHAR Fields

The SQL language provides powerful search capabilities. FairCom DB SQL provides the LIKE clause for some field types, however, the LIKE predicate clause does not allow searching LONG fields, such as LVARCHAR.

To extend advanced searching for LVARCHAR fields, FairCom DB SQL now provides CONTAINS, as an available search condition predicate.

3.9 Support for SQL Transaction Isolation Levels 1 and 2

FairCom DB SQL now supports both SQL transaction isolation levels 1 and 2, providing greater data integrity.

3.10 Additional Search Options for FairCom DB SQL LONG Field Types

A limitation of the FairCom DB SQL LVARCHAR and LVARBINARY fields is the inability to use SQL functions such as UPPER() or LOWER() to perform case insensitive searches. To address this disadvantage of these field types, a proprietary FairCom DB SQL options clause has been introduced allowing a case insensitive search. The ctoption(icontains) clause will allow a case insensitive search on a specific CONTAINS query. Include this clause on a query similar to the following:

```
SELECT * FROM mytable WHERE bigfield CONTAINS 'Search Phrase' ctoption(icontains)
```

The option is only valid for the current query. ctoption(icontains) can be used with both FairCom DB SQL LONG fields allowing both character and binary searching.

3.11 Query Timeout Options

FairCom DB SQL now supports a timeout option for an executing query. This feature can ensure that an unintended query statement does not consume excessive processing time. This feature can be enabled in ODBC, JDBC and the FairCom DB SQL ADO .NET Data Provider.

3.12 Complete RIGHT OUTER JOIN Syntax

FairCom DB SQL has supported a RIGHT OUTER JOIN clause with an alternate syntax as follows:

```
SELECT * FROM T1, T2 WHERE T1.C1 (+) =  T2.C3;
```

In a right outer join, the information from the table on the right is preserved: the result table contains all rows from the right table even if some rows do not have matching rows in the left table. Where there are no matching rows in the right table, FairCom DB SQL generates null values.

A better syntax, much more convenient for SQL developers, is to explicitly state the condition in a SQL statement using the full syntax of RIGHT OUTER JOIN. FairCom DB SQL has been enhanced to allow this new syntax.

3.13 DEFAULT Clause with ALTER TABLE

The `ALTER TABLE` SQL command alters the schema of an existing table by adding new columns or modifying existing columns of the specified table. FairCom DB SQL now has support for specifying a default value when adding or modifying a column. With the `ADD` and `MODIFY` column operations an optional `DEFAULT` value can be specified for each of the columns.

3.14 ODBC and JDBC Driver Socket SEND/RECV Timeout

A send/receive timeout option has been added such that a FairCom DB SQL ODBC or JDBC client can request a timeout for a socket. If the client experiences a lengthy wait for the server to reply, the client can continue to work after closing the connection.

A FairCom DB SQL ODBC driver can set the timeout with a call to the **SQLSetConnectAttr()** ODBC API function and the *SQL_ATTR_CONNECTION_TIMEOUT* parameter with the time value in seconds.

With the FairCom DB SQL JDBC Driver, it is possible at connection time to set a timeout value in milliseconds using the **DriverManager.getConnection(***string, properties***)** method.

3.15 ODBC Driver Login Timeout

The FairCom DB SQL ODBC Driver for Windows now supports a configurable timeout on driver connection login. An application can set the login timeout by calling the **SQLSetConnectAttr()** ODBC API function with the *SQL_ATTR_LOGIN_TIMEOUT* attribute and a timeout value.

3.16 Improved FairCom DB SQL Java Configuration

FairCom DB SQL now considers existing environment variables during startup processing for the Java environment used with stored procedures. The following variables are used in the FairCom DB SQL configuration file, *ctsrvr.cfg,* to set the parameters for the Java JVM and compiler. These configuration settings are required for FairCom DB SQL Stored Procedures and Triggers and User Defined Functions support:

```
SETENV CLASSPATH=
SETENV JVM_LIB=
SETENV JAVA_COMPILER=
SETENV DEBUG_JVM
```

3.17 Reserved Keywords With Microsoft Excel and ODBC

A new feature was added to allow a query from Microsoft Excel and ODBC on tables containing fields identified with FairCom DB SQL reserved keywords. The FairCom DB SQL ODBC Driver now considers field names preceded by a table name (i.e. tablename.fieldname) as a valid field even when they are reserved keywords. This is done by automatically wrapping the qualified fieldname in double quotes, which has the effect of making it case-sensitive. Because this

behavior only applies to qualified identifiers it can be avoided by specifying only the fieldname without the tablename.

To use this particular feature with Microsoft Excel and the FairCom DB SQL ODBC Driver, a data source should specify the string “DHQQI” in the new **Options** field of the Data Source Configuration window. This will turn on this special processing for only this connection.

3.18 Additional ORDER BY Clause Usage

FairCom DB SQL previously did not support an `ORDER BY` clause in a subquery. This support has been added. Note however, subqueries with a combination of `TOP`, `GROUP BY` and outer references remain unsupported.

FairCom DB SQL now also supports an `ORDER BY` clause in a `FOR UPDATE` query.

Note: This is a non-standard SQL feature as specified by SQL92. Additionally, full cursor update is not supported by FairCom DB SQL.

3.19 Copy a Database with the FairCom DB SQL Maintenance Utility

Occasionally, it is necessary to copy or otherwise change the name of a FairCom DB SQL database. Simply renaming the directories and files in the FairCom DB SQL environment will not suffice. To facilitate a rename ability, a copy database function has been added to the FairCom DB SQL Database Maintenance utility, **ctsqlcdb**. The added *-copy* option will copy an existing database into a new database, leaving the original database intact.

3.20 PREIMAGE Tables in FairCom DB SQL

A performance gain can be obtained in some situations by avoiding transaction logging of files, thereby foregoing the protection of recovery. With FairCom DB ISAM it is possible to create files with such a transaction mode using *ctPREIMG* as the file mode. This support is now extended to FairCom DB SQL Tables as an option when they are created. The additional option, `'PREIMG'`, has been added to the FairCom DB SQL keyword `STORAGE_ATTRIBUTES`.

3.21 FairCom Security Handshake Now Available in all FairCom DB SQL Products

FairCom’s FairCom DB Servers include an optional security mechanism that ensures only authorized c-tree clients can connect to the server. The server checks to make sure that there is a matching “handshake” between the client and server. This support is now extended to FairCom DB SQL.

The handshake is enabled for all FairCom DB SQL client interfaces including the FairCom DB SQL ODBC driver, the FairCom DB SQL JDBC Driver, the FairCom DB SQL ADO .NET Data Provider, as well as underlying standard FairCom DB ISAM clients.

This feature is a custom feature. FairCom generates a custom version of the FairCom DB SDK and the FairCom DB Servers for customers requiring this level of security. Contact your nearest FairCom office to discuss incorporating this security feature into your FairCom DB SQL solution.

3.22 Updated FairCom DB SQL Reserved Words

FairCom DB SQL has many new enhancements and functions. With these additions, the reserved words used by FairCom DB SQL have increased. Refer to the FairCom DB SQL Reserved Words Appendix for the complete list of new FairCom DB SQL reserved words.

4. New Features for FairCom DB ISAM Server

Introduction

The FairCom DB ISAM Server remains one of the fastest direct record access database technologies available. Consider the FairCom DB ISAM Server whenever you need advanced high speed throughput and control not found anywhere else. With FairCom DB V9.0 many new features have been added. New scalability enhancements enable massive performance gains on high end multi-CPU systems; in some cases 10 to 50 percent faster than V8.14. Added Quiesce and Fileblock features give administrators maximum control over their files. For the ultimate in data integrity, FairCom DB has undergone extensive new advancements in Dynamic Dump backup capabilities. Read on for the latest available features that are in FairCom DB V9.0.

4.1 Temporarily Suspend FairCom DB Operations

From its inception, FairCom has provided tools and features for the application developer to produce applications requiring as little attention from the end user as possible. FairCom DB fulfills this goal with remarkable success. The FairCom DB Server is designed to maximize uptime to near 100%. Install it, start it, and forget about it.

Secured, reliable backups are a critical component of business continuity plans. The c-tree Server dynamic dump backup feature allows for unattended operation. Submit the dynamic dump script to the c-tree Server and walk away knowing your data is safely stored away, guarding against any misfortune your hardware may endure.

Periodically, there may be times where an administrator may wish to open a maintenance window for more detailed and comprehensive data management. Stopping the c-tree Server can be a challenging task in a high availability setting with large numbers of users. Coordinating this down

time can be a frustrating experience. What is needed is a functionality to halt c-tree Server operations cleanly, while allowing user applications to remain connected. This function is frequently referred to as “quiesce”.

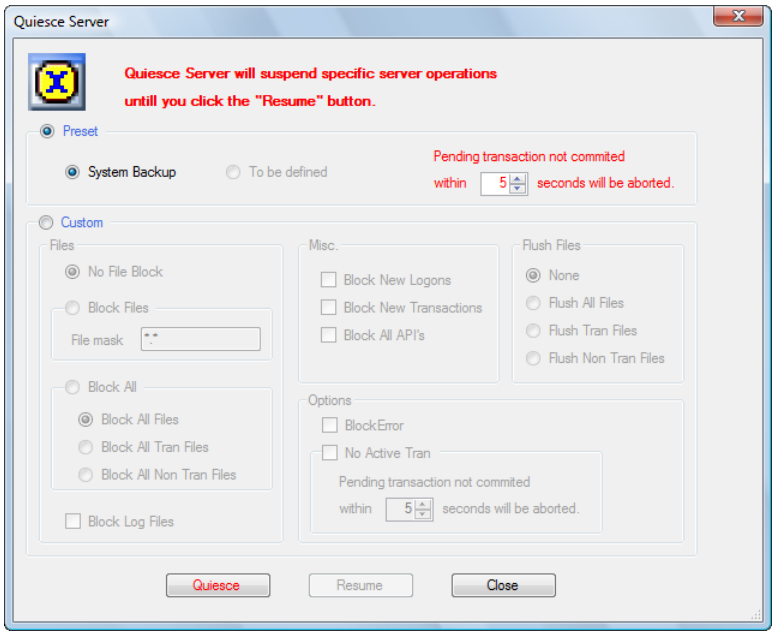
A powerful new FairCom DB functionality is now available to suspend, quiet or quiesce, FairCom DB operations and later re-enable them. This allows FairCom DB administrators to perform maintenance or other on demand activities without having to stop an application. Users are temporarily held back from operations. Files can be readily accessed for backup, especially useful for hardware based disk snapshot utilities.

A new option has been added to the FairCom DB administrator utility, **ctadmn**, to enable the quiet state. From the main **ctadmn** menu choose the Quiesce Server option. Used in the appropriate setting, this new option will be a valuable tool to many FairCom DB administrators.

The power behind this feature is also available to developers to include in their own applications and utilities. Include this functionality directly into your FairCom DB application with a call to **QuietCtree()**. **QuietCtree()** is a new API addition to the FairCom DB SDK with many options to give you maximum control.

You can take advantage of the Quiesce mode via several methods:

- FairCom DB Administrator Utility **ctadmn** (Option 8 on the Main Menu)
- **QuietCtree** API Call
- FairCom DB Server Monitor tool (Quiese button on the Main Menu)



4.2 Enhanced Dynamic Dump

FairCom DB can be configured to provide a powerful backup capability to ensure data integrity via the Dynamic Dump feature. Periodic backups provide a means to recover from problems or roll back the database to a former known “good” state at a specific point in time.

The FairCom DB Dynamic Dump feature provides a safe, secure method to back up your data while the server is fully operational. This avoids the downtime of shutting down the server for a complete backup of all of the files. Using transaction logs, Dynamic Dump is able to provide an online solution for data backup. Powerful restore and roll forward utilities give you ultimate control over your final database state.

While the FairCom DB Dynamic Dump does provide facilities for files not under transaction control, these files do not receive the benefit of data consistency, as data and index files are backed up individually. By default, the Dynamic Dump also stores an entire backup into a single data stream, a file, regardless of the number of files backed up. In addition, when restoring these files, the directory structure had to be in place to allow the files to be properly relocated. Even more restrictive was the fact that important non-c-tree files could not be backed up.

FairCom has greatly enhanced the Dynamic Dump and restore features of FairCom DB. Now, applications that don't employ transaction processing can enjoy full online backup capabilities with data consistency maintained. New options allow the Dynamic Dump to store the backed up files in a "native" file and directory format for simple copy and restore operations. Needed directory paths are detected and re-created as needed during the restore operation. Finally, even non c-tree files can now be backed up and restored.

- **Automatic Promotion of Files to Transaction Processing for Full Dynamic Dump Potential**
- **Automatic Restore of a Dynamic Dump for "Ready-to-Go" Files**
- **Non-ctree Files Included in a Dynamic Dump**

4.3 FairCom DB CPU Configuration Options

FairCom DB can now be configured to run on specified CPUs on Windows and Solaris operating systems with multi-CPU configurations. For some applications and environments with a large number of CPUs available, this may improve performance.

The `CPU_AFFINITY` keyword allows an administrator to set the processor affinity mask for the FairCom DB process. By default, the FairCom DB process will use all available processors.

- `CPU_AFFINITY <cpu list>`

Refer to the *FairCom DB Administrator's Guide* for complete details about this new configuration option.

Note: A default FairCom DB installation is restricted to 2 CPUs. This limit can be increased with an appropriate activation key available from your nearest FairCom office. You will receive a notification on startup if you attempt to run a default installation on a hardware platform with more than 4 CPUs (counting all cores). To take advantage of more than 2 CPUs, you can either purchase the necessary activation key, bind your FairCom DB process to 2 CPUs, or specify the CPU_AFFINITY keyword to allow the FairCom DB process to bind itself to 2 CPUs.

4.4 Administrators Can Now Define the FairCom DB Port Number

FairCom DB supports a new option that sets the TCP/IP port. Previously, the TCP/IP port used was computed with the `SERVER_NAME` keyword, which computed the port as 5001 plus the sum of the ASCII values of the characters in the server name. The new `SERVER_PORT` keyword makes it easy for a server administrator to set the TCP/IP port. The new keyword is used as follows:

```
SERVER_PORT <port_number>
```

where *<port_number>* is the TCP/IP port to use.

Example

```
SERVER_PORT 5597
```

If both `SERVER_NAME` and `SERVER_PORT` are specified in the server configuration file, `SERVER_PORT` takes precedence over `SERVER_NAME`.

When a client prefixes the server name with the pound sign (#), the specified server name is now interpreted as a numeric port. Otherwise, the specified server name is converted to a numeric port using the original approach. For example: *#6000@localhost* is interpreted as port 6000 (the new approach), and *6000@localhost* is interpreted as port 5198 (the original approach).

In addition, FairCom DB on Unix systems now logs the TCP/IP port number it is using to the server status log.

4.5 Default Extended Headers for Enhanced Feature Support

Extended headers provide support for a range of enhanced c-tree features including

- *HUGE* files
- Six-byte transaction numbers
- Segmented files
- Transaction dependent creates and deletes
- Restorable deletes

This extended support is now enabled by default with FairCom DB for all newly created files, regardless of the function call to create the files. Previously, this mode was only enabled with calls from an *Xtd8* specific function and defining the *XCREblk* structure.

The advantage of this new approach is that 6-byte transaction numbers are used by default, which avoids potential unexpected transaction number overflows, or in some cases, encountering error **R6BT_ERR** (745, *6BTRAN* file required).

This feature can be disabled with the following keyword should this be necessary for backward compatibility:

```
COMPATIBILITY REVERT_TO_V6HDR
```

Standalone applications can disable this support by setting the *cth6flg* global variable to any non-zero value.

4.6 Transaction Timeout

There are occasions where it is valuable to limit the time that a FairCom DB transaction is allowed to span. Long held transactions can cause a number of application related issues. Examples of this includes holding locks on a record, or preventing updates to be available to other users in a timely manner. A new FairCom DB configuration option is available to limit the time of a pending transaction. Specify the following in the c-tree Server configuration file, *ctsrvr.cfg* to limit the time of a transaction:

- `TRAN_TIMEOUT`

4.7 Blocking Lock Timeout

A blocking lock timeout feature is available to avoid excessively long blocking lock waits. When used, this feature returns error **UTIM_OUT** (827) to the caller of the lock request. The function **ctLOKTIMOUT()** is used to set, change and clear this timeout feature.

A FairCom DB configuration keyword has been defined to set a value for a blocking lock timeout on server startup. Specify the following in your *ctsrvr.cfg* file:

- `BLOCKING_LOCK_TIMEOUT_SEC`

4.8 Commit Read Locks for Guaranteed Data Reads

Without explicit read or write locks, it is possible to have a partially updated record buffer returned in a high transaction volume environment. The returned record buffer could consist of partial old data, and partial newly updated data from a transaction commit operation from a concurrent thread. While the occurrence of this event is extraordinarily rare (on the order of one in a million record reads) it is an important issue in a high volume situation.

FairCom DB introduces a new feature to prevent these “dirty” record reads. Commit Read Locks enable an implicit, high performance, low-level record lock, ensuring consistent data record reads in high transaction volume environments. This new behavior is on by default. If a record update, under transaction control, is updated or deleted without an explicit lock held, an error **CMLK_ERR** (768, commit lock error: make sure record update performed with lock) is now returned.

A server configuration keyword is available to disable this feature for backward compatibility.

- `COMPATIBILITY NO_COMMIT_READ_LOCK`

4.9 Automatic Transaction Processing for non-TRANPROC files

This new feature extends automatic transaction support to include recoverable transactions without any application changes. This feature permits files to be automatically created with *PREIMG* or *TRNLOG* support. Appropriate transactions are automatically started such that, except for the configuration entries, the application itself would not require any changes. An open file would also set the file to *PREIMG* provided the index was capable.

Applications that don't currently employ transaction processing can enjoy many of the benefits of transaction processing, such as online backup capabilities with complete data consistency maintained.

- `AUTO_PREIMG`
- `AUTO_TRNLOG`

4.10 Prime Cache By Key

The FairCom DB `PRIME_CACHE` configuration option supports priming the data cache with the specified number of bytes of data from the specified data file, in physical order from the start of the data file. FairCom DB now supports priming the data cache in forward AND reverse order by index.

- `PRIME_CACHE_BY_KEY`

4.11 Rebuild Callback Support in Client/Server Mode

This modification introduces the ability for applications to register a rebuild callback function in client/server mode using the **SetCallbackOnRebuild()** API function, which was originally supported in standalone mode only. If a client registers a client-side rebuild callback function, during a rebuild the server sends messages to the c-tree client library causing it to call the client-side callback function when the internal rebuild loop counter reaches a value that is a multiple of the *step* parameter passed to **SetCallbackOnRebuild()**. If *step* is set to 1, the callback function is called once per record/key. The rebuild outputs a rebuild status message.

4.12 Scaling Factors for Configuration Keyword Values

FairCom DB data and index cache configuration options now support specifying a scaling factor used when interpreting cache memory sizes. The supported scaling factors are:

- KB: interpret the specified value as a number of kilobytes.
- MB: interpret the specified value as a number of megabytes.
- GB: interpret the specified value as a number of gigabytes.

4.13 Disable the FairCom DB Communication SubSystem

FairCom DB can be configured to disable its communication subsystem, such that only bound clients can use the server.

- `COMM_PROTOCOL DISABLE`

When this option is specified the communication subsystem is disabled at startup, and remains disabled during the entire lifetime of the server process. This feature is useful when FairCom DB is loaded as a .dll or shared library into an application server process. Although external clients are prevented from using FairCom DB, threads in the application server process can use the FairCom DB subsystem. This option can also be used to prevent ISAM-level access to FairCom DB SQL.

When this option is in effect, the server logs the following message to *CTSTATUS.FCS*:

```
c-tree Server communication subsystem is disabled.
```

4.14 Automatic Shared Memory Protocol for Local Connections on Windows

A c-tree TCP/IP client library compiled on the Microsoft Windows OS now supports automatically switching to the shared memory protocol when the specified server is running on the same system as the client and the server is configured to use the shared memory protocol. Shared memory offers greatly enhanced performance for local connections on the same physical machine, sometimes up to five times the speed of TCP/IP. For communications intensive applications, this can offer a significant performance boost.

The FairCom DB Monitor utility will display the communications protocol associated with each connection. The FairCom DB administration utility, **ctadmn**, has also been updated to display either *SQL_TCPIP* or *SQL_SHAREMM* depending on which protocol the SQL client is using. In addition, support for terminating FairCom DB SQL clients (both TCP/IP and shared memory clients) is available in both the FairCom DB Monitor utility and the **ctadmn** utility.

4.15 Additional SNAPSHOT Options

The FairCom DB *SNAPSHOT* feature now includes support for collecting disk read and write timings on a per-file basis when high-resolution timer support is activated.

Enable c-tree Disk Read/Write Times by File

Collection of disk I/O timings is disabled by default and is enabled with one of the following methods:

- Use the FairCom DB Monitor utility.
- Add the option `DIAGNOSTICS SNAPSHOT_IOTIME` to the FairCom DB configuration file, *ctsrvr.cfg*.
- Use the **ctstat** utility's *-iotime* option.
- Use the **ctSNAPSHOT()** API function.

Programmatically Output c-tree Disk Read/Write Times by File

The **ctSNAPSHOT()** API function now supports a mode that writes snapshot statistics for all files open by FairCom DB to disk. Use the **ctstat** utility's *-file* option or call **ctSNAPSHOT()** using the *ctPSScsvFile* or *ctPSStextFile* mode as shown below to write snapshot statistics for all open files to the file *SNAPFILE.FCS* in comma-delimited or human-readable format.

Enable c-tree Function Call Times by File

Collection of function timings is disabled by default and can be enabled by any of the following methods:

- Add the option `DIAGNOSTICS SNAPSHOT_WRKTIME` to the FairCom DB configuration file, *ctsrvr.cfg*.
- Use the FairCom DB Monitor Utility.
- Use the **ctstat** utility's *-wrkstat* option.
- Use the **ctSNAPSHOT()** API function.

Programmatically Output c-tree Function Call Times by File

The FairCom DB *SNAPSHOT* API supports a mode that writes function timings for all files open by FairCom DB to disk. Use the **ctstat** utility's *-funcfile* option or call **ctSNAPSHOT()** using the *ctPSScsvFunc* or *ctPSStextFunc* mode as shown below to write function timings for all open files to the file *SNAPFUNC.FCS* in comma-delimited or human-readable format.

4.16 SNAPSHOT Histogram Support

The *SNAPSHOT* feature includes a histogram of transaction times. Modifications were done to support the generalized collection and reporting of the histogram data. These routines will support a histogram capability for other *SNAPSHOT* measurements. We also added histograms of waiting times for blocked lock requests: one for waiting times for blocked data record lock requests, and one for waiting times for blocked index lock requests (please note that the index locks are not controlled by the user). There is a small amount of overhead associated with mutex calls to collect clean statistics.

The configuration file can change the default histogram intervals (box width) for the transaction time histogram (default of 50,000 µsec or 0.05 seconds) and the lock waiting time histograms (default of 10,000 µsec or 0.01 seconds):

- `SNAPSHOT_TRANTIME_USEC <tran time histogram interval width in microseconds>`
- `SNAPSHOT_LOCKWAIT_USEC <lock wait histogram interval width in microseconds>`

4.17 Change Configuration Options at Run Time

The ability to change FairCom DB configuration settings at run time provides a flexible means to monitor FairCom DB health. Previously, you had to shutdown your server, change your server configuration settings, and restart. FairCom now provides the ability to dynamically change

certain FairCom DB settings such as the function, checkpoint, memory, and request time monitors.

FairCom DB supports a new API that can be used to change configuration options at run time without restarting. With this feature, you can have your application, or utility and maintenance programs, periodically activate/deactivate particular monitor settings.

- **SetSystemConfigurationOption**

4.18 Advanced Commit Delay Options

For advanced control of the intricate commit delay timing statistics, additional controls have been added to the commit delay time calculation.

It is recommended that these values be carefully profiled as they can impact performance in many unexpected ways.

- `COMMIT_DELAY_SCALE`
- `COMMIT_DELAY_BASE`

4.19 FairCom DB Server SDK File Callback Options

The FairCom DB Server SDK allows an application developer to create a custom FairCom DB database engine with specific application code for increased performance. This feature makes advanced callback capabilities available to the developer. Custom operations during c-tree file create, open, and close operations can be implemented with the following functions available in the *ctuserx.c* module.

- **ctFileCreateCallback()**
- **ctFileOpenCallback()**
- **ctFileCloseCallback()**

4.20 Assignment of Default File Permissions to User Groups

A new class of configuration entries permits default file permissions to be assigned to one or more groups including two special groups: *WORLD* and *OWNER*. The primary need for this capability is to enforce permission flags on files that have already been created without a permission mask (i.e., the permission mask is zero at file create). A zero permission mask is equivalent to granting everyone all rights:

OPF_ALL | GPF_ALL | WPF_ALL

Note: *ALL* does not include the special *NOPASS* flag that permits a file to be opened for reading without supplying the file password. To grant *NOPASS* permission, it must be included explicitly.

- `FILE_PERMISSIONS`

4.21 FairCom DB Stack Dumps for Windows and UNIX

FairCom DB is able to dump a stack trace into a file during a self-initiated shutdown by taking advantage of a system utility, **pstack.** This new feature adds a similar capability for the Windows operating system by creating a mini-dump containing information about the stack trace.

The stack is dumped into a file named *stack<pid>_<lognum>.mdmp* (under Solaris the name is *pstack<pid>_<lognum>.txt*). Visual Studio 7.0 is required to inspect the file for Windows stack dumps. It is also suggested to enable Dr. Watson on the Windows system of interest when attempting to generate a dump file with information needed to trace a continuing FairCom DB stack dump.

This feature requires an external DLL, *dbghelp.dll*, which is part of the Windows installation, and is dynamically loaded at stack dump time. It also requires that this DLL exports the function **MiniDumpWriteDump()** which may not be the case with older versions of this DLL. In the case where the DLL cannot be found or it does not contain the function, the stack dump fails and a message is logged in *CTSTATUS.FCS* without any other consequence.

FairCom DB for AIX 5.2 and later now also supports a process stack trace when a fatal error occurs. FairCom DB uses the AIX **procstack** utility to log the stack trace to the file *procstack_<pid>_<sernum>.log*, where *pid* is the FairCom DB process ID and *sernum* is an ever-increasing number (starting at 1).

4.22 File Permission Mode for Files Created by c-tree on Unix Systems

Previously, FairCom DB always created files with full permissions (file mode 0666) on Unix systems. Now FairCom DB defaults to a permission mode of 0660 (read/write access for owner and group; no access for world) for the files it creates.

When using FairCom DB, the permission mode assigned to files created by FairCom DB can be set with the server configuration keyword `FILE_CREATE_MODE` to specify the desired file permission mode.

- `FILE_CREATE_MODE`

4.23 Retry Options

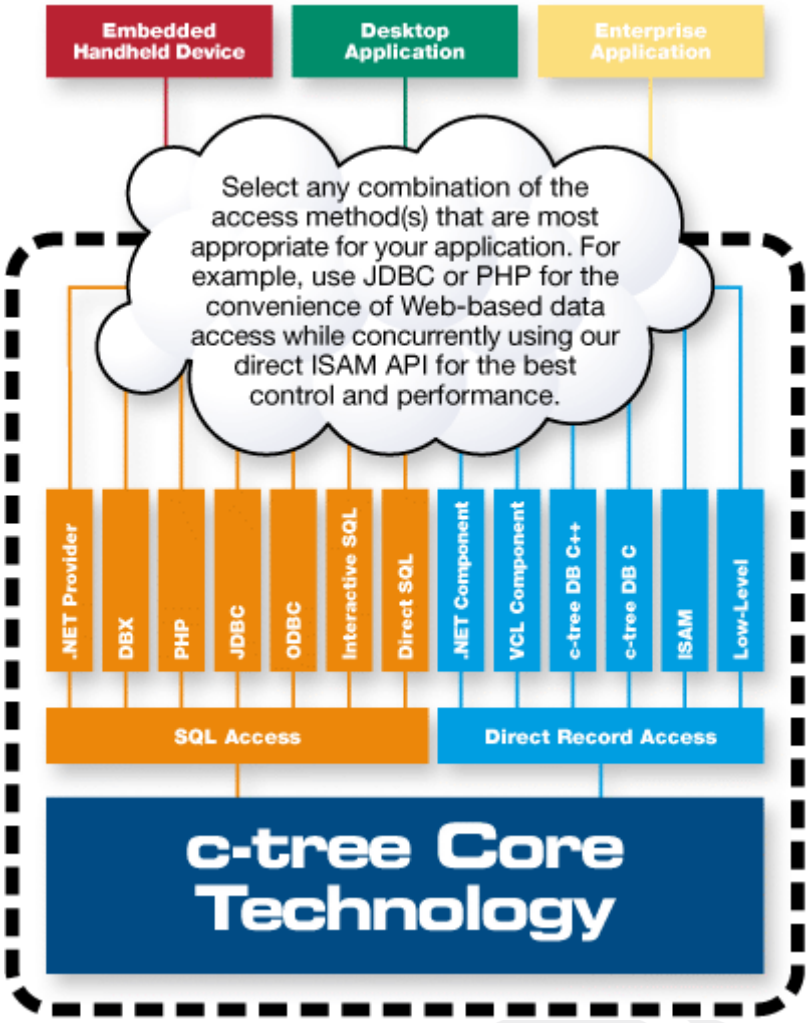
FairCom DB now supports introducing limits and delays before internally retrying an ISAM record read operation when an **ITIM_ERR** (160) is encountered. The configuration option `ITIM_RETRY_DEFER <defer_time>` specifies the time in milliseconds for which FairCom DB sleeps a thread that encounters error **ITIM_ERR** during an ISAM record read operation before retrying the ISAM record read operation. The maximum number of **ITIM_ERR** retries for a particular ISAM record read operation is determined by the `ITIM_RETRY_LIMIT` configuration option.

- `ITIM_RETRY_LIMIT`
- `ITIM_RETRY_DEFER`

5. Additional FairCom DB SQL Interfaces

Introduction

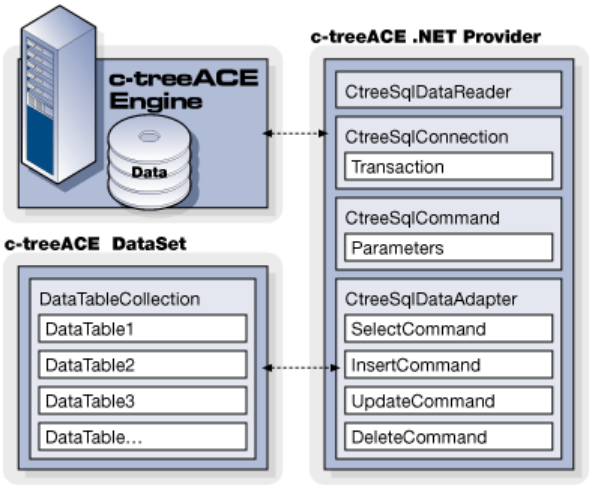
FairCom DB SQL introduces four new exciting interfaces. The powerful FairCom DB SQL ADO .NET Data Provider brings your data into your .NET applications easily with drag and drop simplicity. c-treePHP modules extend your FairCom DB SQL data easily to the web. CodeGear and Delphi users will appreciate FairCom DB SQL dBX. The all new FairCom DB Direct SQL brings true SQL power inline with your application. See how FairCom DB SQL can empower your next application.



5.1 FairCom DB SQL ADO .NET Data Provider

The .NET Framework is a computing platform greatly simplifying application development in the highly distributed environment of the Internet. ADO.NET provides a rich set of components to create distributed, disconnected and data-sharing applications. ADO.NET is an integral part of the .NET Framework, providing relational data access to systems such as FairCom's FairCom DB SQL.

A .NET Data Provider is a bridge used to connect ADO.NET applications to a database, execute commands and retrieve results. The FairCom DB SQL .NET Data Provider gives you access to your FairCom DB SQL data from a .NET application. The .NET Data Provider is lightweight, creating a thin layer between the data source and your code, thus increasing performance without sacrificing functionality. A .NET Data Provider consists of a set of classes implementing interfaces specified in Microsoft's specification for .NET Data Providers.



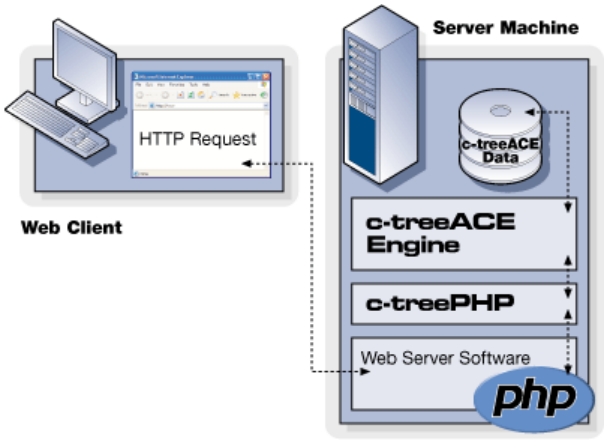
The FairCom DB ADO.NET Data Provider gives you access to your FairCom DB SQL data through this easy to use interface. With integrated support for Visual Studio, your application development could never be easier. Drop your components onto a Windows form and immediately begin accessing the power of FairCom DB SQL with your data.

5.2 FairCom DB SQL PHP

Today's progressive business environments demand online availability. The method of choice is via the web and PHP has become the language of choice for dynamic web scripting.

PHP (Hypertext PreProcessor) is a widely-used general-purpose scripting language that is especially suited for web development and can be embedded into HTML. FairCom DB SQL is a powerful database technology offering a wide range of interfaces now including PHP support. FairCom's FairCom DB PHP module extends your data access to the world through this popular web interface.

FairCom DB PHP Interface technology gives the means to access data from FairCom DB SQL.



FairCom DB PHP modules can be installed with either Apache or Microsoft IIS (Windows) web servers.

5.3 FairCom DB SQL dbExpress

Embarcadero's dbExpress provides developers superb database connectivity from their applications. dbExpress is a set of lightweight database drivers providing fast access to SQL

database servers. When you deploy a FairCom DB SQL database application utilizing dbExpress, you need only include the FairCom DB SQL dbExpress Driver dll with the application files you build.

The FairCom DB SQL DBX Interface technology provides the driver necessary to connect to the FairCom DB SQL database engine. Quickly build client applications using the advanced RAD Studio development environment and efficiently access your FairCom DB SQL data through either Delphi or C++Builder.

5.4 FairCom DB Direct SQL

The FairCom DB Direct SQL interface (DSQL) is an inline SQL application programming interface (API) designed for C/C++ developers who wish to embed SQL statements directly into their programs.

The FairCom DB DSQL API is designed as a foundation for building your own SQL APIs. It simplifies your code. In fact, many of the SQL GUI tools authored by FairCom utilize this API.

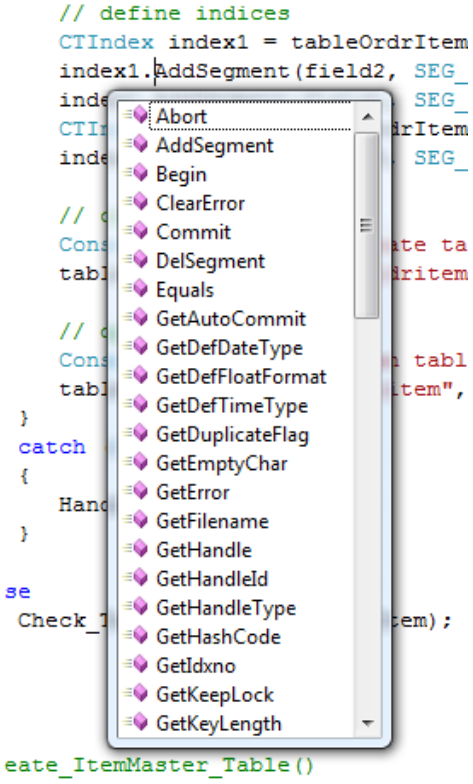
FairCom DB DSQL gives application developers control and ease of deployment over other embedded APIs, such as ODBC, and DSQL requires no time-consuming pre-processing steps or the presence of an ODBC manager.

Consider using FairCom DB DSQL for your next embedded application solution.

6. Improved FairCom DB API .NET

Introduction

FairCom DB API .NET is an advanced high-level programming API exposing the core classes of c-tree technology in the .NET framework. New methods and enhanced features have been added to make this popular programming interface even more valuable for your high performance database applications.



6.1 New Delphi .NET Support

With the first release of c-tree Plus for .NET in V8.14, the supported development environments included C#, J#, and VB .NET. FairCom now extends FairCom DB API .NET support for the Delphi .NET environment as well. Now you can leverage your existing Delphi .NET experience with FairCom DB.

6.2 Delphi .NET Compatibility when using Create()

Delphi used the name **Create()** as its default class constructor method. *CTSession*, *CTDatabase* and *CTTable* classes each had **Create()** methods that conflicted with the Delphi default. An alternate **Create()** method was added to each of the above classes in the form of **CTSession.CreateSession()**, **CTDatabase.CreateDatabase()** and **CTTable.CreateTable()** to alleviate this discrepancy.

6.3 Batches for c-treeDB

There are situations where a record fetch, a record insert or a record delete operation has to be applied to a group of related records. For example, if an invoice record is being displayed, it may be necessary to retrieve all invoice items for that invoice as well. c-tree Plus has a batch feature which allows for the high performance transfer of large number of records at once. This greatly improves application performance. FairCom DB API .NET has been enhanced with this batch support.

The following types of batch operations may be performed:

Retrieve records by partial key

All records with keys matching a partial target key are loaded into an internally maintained buffer region. If there are more records than fit in the buffer, those that fit are loaded, and subsequent calls will retrieve the remaining records.

Retrieve records by index range

Retrieve all records that match an index range expression. All matched records are loaded into an internally maintained buffer region. If there are more records than fit in the buffer, those that fit are loaded, and a subsequent call will retrieve the remaining records.

Retrieve records by physical order

All records of a table are loaded by physical order into an internally maintained buffer region. If the selected records do not fit in the buffer, those that fit are loaded, and subsequent calls will retrieve the remaining records.

Insert a group of records

A group of new records are loaded into an internally maintained buffer region and this group of records are inserted into a table.

Delete a group of records

All records with a key matching a partial target key are deleted.

Review the complete documentation in the FairCom DB API .NET Developer Guide for complete information.

- c-treeDB.NET Programmer's Reference Guide (https://docs.faircom.com/doc/nav-dotnet/)

6.4 Support for Resources

It can be advantageous at times to attach auxiliary information to a particular table that does not conform to the record structure of that table. Examples of this include features such as versioning or special flags relating to the status of the table. Generally, this information is not repeated in every record of the table. You could create a special record with a special key value that you do not process as you do your regular records. Ultimately, however, this forces exception handling routines for this special case and can impose a heavy maintenance cost on the life cycle of the application.

FairCom DB provides a unique feature created for exactly this purpose: Resources. Resources are special variable-length records stored within your data file, whether you use fixed or variable length records. A set of API functions provide access to create, update and delete these resource records. These records do not require a key in an index, therefore your program does not access them via routine data handling functions.

Resources provide critical support for many advanced FairCom DB features. FairCom defined resources allow seamless functionality of many of the c-tree Plus ODBC Drivers, incremental ISAM functionality, conditional index support, FairCom DB API and FairCom DB SQL. These resources continue to be important as new technology is added. Resources are added either automatically by some FairCom DB features or manually by the developer. The use of resources requires the *RESOURCES* define in the FairCom DB library, which is the default.

FairCom DB API .NET now offers support for adding, managing and deleting c-tree resources. Review the complete documentation in the FairCom DB API .NET Developer Guide for complete information on this new FairCom DB API .NET feature.

CTResource Class Methods

- **Add**
- **Delete**
- **Find**
- **First**
- **GetData**
- **GetDataLength**
- **GetName**
- **GetNumber**
- **GetType**
- **IsLocked**
- **Next**
- **SetData**
- **SetName**
- **SetNumber**
- **SetType**
- **Unlock**
- **Update**

6.5 Attach and Detach Existing Sessions

There are situations where an existing connection to c-tree already exists, via a call to a low level or ISAM c-tree initialization function, however, FairCom DB for .NET functionality is required without terminating the existing connection and starting a new c-treeDB session. The following functions were added to FairCom DB for .NET to permit a session handle to be attached and detached from an existing c-tree connection.

- **CTSession.Attach**
- **CTSession.Detach**
- **CTSession.GetAttachMode**

6.6 Attach and Detach Tables

A FairCom DB API .NET object can be attached and detached to an already open data file. Applications may need to open a table using FairCom DB ISAM and FairCom Low-Level functions and then attach the table to FairCom DB API .NET to take advantage of full FairCom DB API .NET functionality.

- **CTTable.Attach**
- **CTTable.Detach**

6.7 Retrieve Field, Index, and Segment Status

New functionality was added to FairCom DB API .NET to support enhanced field and segment handling. The status of a field handle is a bitmap describing one or more changes that have occurred with the field object.

- **CTField.GetStatus**
- **CTIndex.GetStatus**
- **CTSegment.GetStatus**

6.8 Exclusive Sessions and Databases

New functions and methods were added to FairCom DB API .NET to allow the setting of the session exclusive flag and to permit the retrieval of the session exclusive flag. The session exclusive flag controls if the session dictionary file is opened in exclusive mode or not. When a session exclusive flag is set, a successful logon will only succeed if no other users are currently logged on to the same session dictionary. This behavior is an extension of an exclusive table open.

- **CTSession.SetExclusive**
- **CTSession.IsExclusive**
- **CTDatabase.SetExclusive**
- **CTDatabase.IsExclusive**

6.9 Retrieve the FairCom DB Configuration

New functions and methods were added to FairCom DB API .NET to retrieve c-tree Plus system configuration values, as well as important dynamic aspects of the system, such as the memory usage and the number of files in use. To determine if a particular system configuration option is active, call **GetSystemConfig()**, passing the corresponding pre-defined constant for that option, and check if the value returned is non-zero.

- **CTBase.GetSystemConfig**

6.10 Move Segments in an Index Definition

New methods were added to FairCom DB API .NET to allow a key segment to be moved within the index to allow for easy editing of index properties.

- **CTSegment.MoveSegment**

6.11 Many New Method Additions

FairCom DB API .NET has many other new methods added for additional flexibility in working with FairCom DB.

CTBase

- **GetTransactionMode**
- **SetTransactionMode**
- **GetOperation**
- **SetOperation**
- **GetAutoCommit**
- **SetAutoCommit**

CTDatabase

- **SetExclusive**
- **IsExclusive**
- **Rename**

CTTable

- **SetOwner**
- **Rebuild**

CTRecord

- **GetErrorIndex**

6.12 New Mode for Table Rebuilds with Missing Index Files

FairCom DB API .NET lacked a facility to rebuild data and index files, due to the impossibility of using **RBLIFIL()** function under modes of transaction dependent creates. FairCom DB API .NET could only rebuild a table with the table opened with the *CORRUPT_OPEN* mode and perform a full alter table by calling **Rebuild()** or by implicitly calling **Alter(***ALTER_TABLE.FULL***)**. While this method works quite well when a table is corrupt, it did not work if one or more of the index files were missing.

A new open mode of *DATAONLY_OPEN* was introduced to FairCom DB API .NET. When an **Open()** is passed with the *DATAONLY_OPEN* mode, FairCom DB API .NET can invoke a full alter table call and rebuild all the indices, if they are missing or not.

6.13 Default Index for Physical Data File Traversal

The default index traversal mode, *CTDB_DATA_IDXNO*, is exposed in .NET through the *IDXNO* attribute.

Previously, to traverse a table in physical order through FairCom DB for .NET one would set the default index (in this case using no index) as follows:

```
record.SetDefaultIndex( (int) IDXNO.DATA);
```

This call required a cast to an *int*. An overloaded method has been added to avoid this cast issue. The following is now properly allowed as a syntax:

```
record.SetDefaultIndex( IDXNO.DATA )
```

7. Many Additional FairCom DB API Features

Introduction

The FairCom DB API C and C++ APIs have been enhanced with over 200 new functions and methods providing extensive support for a huge range of FairCom DB features. A broad summary of these new features are listed here. Please refer to the FairCom DB API C and C++ Programmer Reference Guides for complete and updated information on all of these great new features.

7.1 Batches for FairCom DB API

There are situations where a record fetch, a record insert or a record delete operation has to be applied to a group of related records. For example, if an invoice record is being displayed, it may be necessary to retrieve all invoice items for that invoice as well. FairCom DB has a batch feature which allows for the high performance transfer of large number of records at once. This greatly improves application performance. FairCom DB API has been enhanced with this batch support.

The following types of batch operations may be performed:

Retrieve records by partial key

All records with keys matching a partial target key are loaded into an internally maintained buffer region. If there are more records than fit in the buffer, those that fit are loaded, and subsequent calls will retrieve the remaining records.

Retrieve records by index range

Retrieve all records that match an index range expression. All matched records are loaded into an internally maintained buffer region. If there are more records than fit in the buffer, those that fit are loaded, and a subsequent call will retrieve the remaining records.

Retrieve records by physical order

All records of a table are loaded by physical order into an internally maintained buffer region. If the selected records do not fit in the buffer, those that fit are loaded, and subsequent calls will retrieve the remaining records.

Insert a group of records

A group of new records are loaded into an internally maintained buffer region and this group of records are inserted into a table.

Delete a group of records

All records with a key matching a partial target key are deleted.

Review the complete documentation in the c-treeDB C and C++ Developer Guides for complete information on this new c-treeDB feature.

7.2 Resources for FairCom DB API

It can be advantageous at times to attach auxiliary information to a particular table that does not conform to the record structure of that table. For example, features such as versioning or special flags relating to the status of the table. Generally, this information is not repeated in every record of the table. You could create a special record with a special key value that you do not process as you do your regular records. Ultimately, however, this forces exception handling routines for this special case and can impose a heavy maintenance cost on the life cycle of the application.

FairCom DB provides a unique feature created for exactly this purpose: c-tree Plus Resources. Resources are special variable-length records stored within your data file, whether you use fixed or variable-length records. A set of API functions provide access to create, update and delete these resource records. These records do not require a key in an index, therefore your program does not access them via routine data handling functions.

Resources provide critical support for many advanced c-tree Plus features. FairCom defined resources allow seamless functionality of many of the c-tree Plus Drivers, Incremental ISAM functionality, conditional index support, FairCom DB API and FairCom DB SQL. These resources continue to be important as new technology is added. Resources are added either automatically by some c-tree Plus features or manually by the developer. The use of resources requires the *RESOURCES* to be defined in the c-tree Plus library, which is the default.

FairCom DB API now offers support for adding, managing and deleting c-tree resources. Review the complete documentation in the FairCom DB API C and C++ Developer Guides for complete information on this new c-treeDB feature.

C API Functions

- **ctdbAddResource**
- **ctdbAllocResource**
- **ctdbDeleteResource**
- **ctdbFindResource**
- **ctdbFirstResource**
- **ctdbFreeResource**
- **ctdbGetResourceData**
- **ctdbGetResourceDataLength**
- **ctdbGetResourceName**
- **ctdbGetResourceNumber**
- **ctdbGetResourceType**
- **ctdbIsResourceLocked**
- **ctdbNextResource**
- **ctdbSetResourceData**
- **ctdbSetResourceName**
- **ctdbSetResourceNumber**
- **ctdbSetResourceType**
- **ctdbUnlockResource**

C++ CTResource Class Methods

- **Add**
- **Delete**
- **Find**
- **First**
- **GetData**
- **GetDataLength**
- **GetName**
- **GetNumber**
- **GetType**
- **IsLocked**
- **Next**
- **SetData**
- **SetName**
- **SetNumber**
- **SetType**
- **Unlock**
- **Update**

- **ctdbUpdateResource**

7.3 Unicode Support for FairCom DB API

Simply storing Unicode data has always been possible with FairCom DB API, provided the application treated the data as binary and performed any necessary translations. In this case using the stored Unicode data as a segment of an index was difficult since c-treeDB had no way of knowing how the underlying binary data was encoded: UTF-8. UTF-16, ASCII, etc.

FairCom DB API now provides support for Unicode fields. This support includes:

- Unicode UTF-16 field types
- UTF-8 compliant C/C++ API
- Indexing on UNICODE field data
- ICU library support

Review the complete documentation in the FairCom DB API C and C++ Developer Guides for complete information on this new FairCom DB API feature.

7.4 Callback Support in FairCom DB API

FairCom DB API represents a high-level, easy to use API on top of the popular FairCom DB ISAM and FairCom Low-Level APIs. FairCom DB API is intended as the new standard for FairCom DB programming. When compared to ISAM and low-level APIs, FairCom DB API introduced a more formal definition for the structure of data and index files, new concepts such as *ROWID* and *NULL* field support, and more specifically a formal definition for each field type supported.

Existing applications' data and index files, i.e. data created prior to the publication of the FairCom DB API API, may have implemented certain field types in ways that may be incompatible with FairCom DB API. Field types such as *CT_DATE*, *CT_TIME* and *CT_TIMES* are probably the most common examples of existing data that may be incompatible with FairCom DB API.

The FairCom DB API Callback feature was implemented to provide developers a means to intercept certain FairCom DB API operations and add custom code to manipulate record buffer layouts or change field data on the fly such that the record and field data are compatible with FairCom DB API.

To enable a given callback, perform the following:

1. Write your callback functions. Every callback function must have the callback function type. There is no need to implement every single callback type. Implement only the callback functions that are necessary for your logic.
2. Set your callbacks with the **ctdbSetCallback()** function. For example if you have implemented *CTDB_ON_LOGON* and *CTDB_ON_LOGOUT* callback functions, you call the **ctdbSetCallback()** function once for each callback to the session handle with the appropriate callback type.

You remove a callback from a FairCom DB API handle by calling the **ctdbClearCallback()** function. After removed, a callback for a particular handle will not be called again. You can remove all callbacks from a particular session, database, table or record handle by calling the **ctdbClearAllCallback()** function.

Review the complete documentation in the FairCom DB API C and C++ Developer Guides for complete information on this new FairCom DB API feature.

- c-treeDB C API Programmer's Reference Guide (https://docs.faircom.com/doc/ctreedb/)
- c-treeDB C++ API Programmer's Reference Guide (https://docs.faircom.com/doc/ctreepp/)

7.5 Row Level Permanent Filters

Unlike the standard call to **ctdbFilterRecord()** to establish a temporary, user specific filter for data record reads, a new type of permanent filter can be established that is system-wide (i.e., once established the filter applies to all users).

This powerful capability will enable FairCom DB API users to implement advanced customizations just before reading and/or writing records, such as customized record read and/or record write constraints or row level security. Refer to the FairCom DB section Row-Level Permanent Callback Filters for complete details regarding the callback implementations.

FairCom DB callbacks

After establishing the system-wide filter, you must add your custom code to the following callback functions in module *ctclbk.c*:

- **ctfiltercb_init()** - called every time a new filter is established or when a table with a permanent system-wide filter is opened.
- **ctfiltercb_uninit()** - called every time a new filter is deleted, or when a table with a permanent system-wide filter is closed.
- **ctfiltercb_rowl()** - called every time a record is added or updated, or when an existing record is read, depending on the mode passed to the **ctdbSystemFilterOn()** function.

FairCom DB API C API Methods

- **ctdbSystemFilterOn()** - Establish a row level filter for a table.
- **ctdbSystemFilterOff()** - Removes the row level filter for a table.

FairCom DB API C++ API methods

- **CTTable::SystemFilterOn()** - Establish a row level filter for a table.
- **CTTable::SystemFilterOff()** - Removes the row level filter for a table.

7.6 Key Counting Functions

FairCom DB provides a capability for estimating the number of keys between two key values and an actual count of keys contained in an index. These functionalities were added to FairCom DB API.

FairCom DB API C API Methods

- **ctdbEstimateSpan()** - returns an estimated number of records between two key values.
- **ctdbNumberOfKeyEntries()** - retrieves the number of key entries in an index file identified by its index number.

FairCom DB API C++ API methods

- **CTRecord::EstimateSpan()** - returns an estimated number of records between two key values.
- **CTRecord::NumberOfKeyEntries()** - retrieves the number of key entries in an index file identified by its index number.

7.7 Retrieve a Field that Partially Matches a Key Segment

FairCom DB API was previously able to find the matches between a key segment and a field only if the key segment started at the beginning of a field and exactly matched the entire field, as is required by FairCom DB SQL. It can be advantageous to retrieve a field on which the key segment starts at the beginning of a field, and does not exactly match the entire field. Partial field matching functions and methods have been added to FairCom DB API to handle this.

FairCom DB API C API Methods

- **ctdbGetSegmentPartialField()** - retrieves a field object that matches a key, even if the segment does not match the entire field length.

FairCom DB API C++ API

- **CTSegment::GetPartialField()** - retrieves a field object that matches a key, even if the segment does not match the entire field length.
- **CTSegment::GetPartialFieldName()** - retrieves the name of a field that matches a key segment, even if the segment does not match the entire field contents.

7.8 Retain Locks After Commit

By default, the FairCom DB API begin transaction function, **ctdbBegin()**, will begin a transaction by invoking c-tree's **TRANBEG()** function with the *ctTRNLOG* and *ctENABLE_BLK* modes set. If the **TRANBEG()** function succeeds, FairCom DB API's **ctdbBegin()** function automatically calls **LKISAM()** to suspend locks enabled by **TRANBEG()**, allowing users to enable any locks they wish: read locks, write locks, blocking or non-blocking. In this case, the FairCom DB API commit or abort transaction functions, **ctdbCommit()** and **ctdbAbort()**, call the appropriate c-tree ISAM functions to terminate the transaction and free all locks.

A new feature was added to allow customization in the way these locks are handled by FairCom DB API's begin and end transaction functions.

FairCom DB API C API

- **ctdbGetKeepLock()** - returns the FairCom DB Keep Lock mode.
- **ctdbSetKeepLock()** - sets the FairCom DB Keep Lock mode.

FairCom DB API C++ API

- **CTBase::GetKeepLock()** - returns the FairCom DB Keep Lock mode.
- **CTBase::SetKeepLock()** - sets the FairCom DB Keep Lock mode.

7.9 Filters are Now Record Based, Rather Than Table Based

Originally, FairCom DB filters were implemented as table based, such that once a filter was activated, all ISAM contexts would share the same condition. This situation caused some odd behavior and prevented FairCom DB API record handles from setting their own filters to operate on table records.

A modification was introduced to the filter logic to permit the setting of filters for each independent c-tree ISAM context and FairCom DB API was changed for the new filter condition. Each FairCom DB API record handle can have its own filter and operate independently on the table records.

Since the filters should now be defined on a record basis, table based filters are no longer supported and FairCom DB API returns **CTDBRET_NOTSUPPORTED**.

FairCom DB API C++ API

- **CTRecord::SetFilter()** - sets a record filter condition.
- **CTRecord::GetFilter()** - clears a record filter condition.
- **CTRecord::IsFiltered()** - returns if a record filter condition is set.

7.10 Rebuild Tables with FairCom DB API

ctdbRebuildTable() has been modified to call the c-tree Plus **CTRBLIFILX()** function to rebuild a c-tree table. When used in conjunction with the open modes *CTOPEN_CORRUPT* and *CTOPEN_DATAONLY*, the new **ctdbRebuildTable()** function can be used as a direct replacement for the FairCom DB **ctrbldif** rebuild utility. The complementary FairCom DB API C++ API, **CTTable::Rebuild(),** method was additionally added.

7.11 Default Values for Alter Table Operations

FairCom DB API's alter table function can be used to alter the schema of an existing table by adding new fields or modifying existing fields of the specified table. A new feature has been introduced to allow the alter table function to support a default field value specification. During an alter table operation, when a new field is added to the table, or when an existing field type is changed, an optional default field value can be specified for these fields.

The following functions have been added to the FairCom DB API APIs enabling this capability.

FairCom DB API C API

- **ctdbSetFieldDefaultValue()**
- **ctdbGetFieldDefaultValue()**

- **ctdbIsFieldDefaultValue()**
- **ctdbClearFieldDefaultValue()**
- **ctdbClearAllFieldDefaultValue()**
- **ctdbSetFieldDefaultDateTimeType()**
- **ctdbGetFieldDefaultDateType()**
- **ctdbGetFieldDefaultTimeType()**

FairCom DB API C++ API

- **CTField::SetFieldDefaultValue()**
- **CTField::GetDefaultfieldValue()**
- **CTField::IsFieldDefaultValueSet()**
- **CTField::ClearFieldDefaultValue()**
- **CTTable::ClearAllFieldDefaultValue()**
- **CTField::SetFieldDefaultDateTimeType()**
- **CTField::GetFieldDefaultDateType()**
- **CTField::GetFieldDefaultTimeType()**

7.12 Attach and Detach Existing Sessions to FairCom DB API

There are situations where an existing connection to c-tree already exists via a call to a low level or ISAM c-tree initialization function, however, FairCom DB API functionality is required without terminating the existing connection and starting a new FairCom DB API session. The following functions were added to the FairCom DB API API to permit a session handle to be attached and detached from an existing c-tree connection.

FairCom DB API C API

- **ctdbAttachSession()**
- **ctdbDetachSession()**
- **ctdbGetAttachMode()**

FairCom DB API C++ API

- **CTSession::Attach()**
- **CTSession::Detach()**
- **CTSession::GetAttachMode()**

7.13 Attach and Detach Open Tables to FairCom DB API

A FairCom DB API table handle or object can be attached and detached to an already open data file. Applications may need to open a table using FairCom DB ISAM and FairCom Low-Level functions and then attach the table to a FairCom DB API table handle to take advantage of full FairCom DB API functionality.

FairCom DB API C API

- **ctdbAttachTable()**
- **ctdbAttachTableXtd()**
- **ctdbDetachTable()**

FairCom DB API C++ API

- **CTTable::Attach()**
- **CTTable::AttachXtd()**
- **CTTable::Detach()**

7.14 Retrieve FairCom DB API Field and Segment Change Status

New functionality was added to the FairCom DB API APIs to support enhanced field and segment handling. The status of a field handle is a bit map describing one or more changes that have occurred with the field object.

FairCom DB API C API

- **ctdbGetFieldStatus()** - The status of a field handle is a bitmap describing one or more changes that have occurred with the field object.
- **ctdbGetIndexStatus()** - The status of the index handle is a bitmap describing one or more changes that have occurred to the index object.
- **ctdbGetSegmentStatus()** - The status of the segment handle is a bitmap describing one or more changes that have occurred to the segment object.

FairCom DB API C++ API

- **CTField::GetStatus()** - The status of a field handle is a bitmap describing one or more changes that have occurred with the field object.
- **CTIndex::GetStatus()** - The status of the index handle is a bitmap describing one or more changes that have occurred to the index object.
- **CTSegment::GetStatus()** - The status of the segment handle is a bitmap describing one or more changes that have occurred to the segment object.

7.15 Move Segments in an Index Definition with FairCom DB API

New functions and methods were added to FairCom DB API to allow a key segment to be moved within the index to allow for easy editing of index properties.

FairCom DB API C API

- **ctdbMoveSegment()**

FairCom DB API C++ API

- **CTTable::MoveSegment()**
- **CTIndex::MoveSegment()**
- **CTSegment::MoveSegment()**

7.16 Retrieve c-tree Configuration with FairCom DB API

New functions and methods were added to FairCom DB API to retrieve FairCom DB system configuration values as well as important dynamic aspects of the system, such as the memory usage and the number of files in use. To determine if a particular system configuration option is active, call **ctdbGetSystemConfig()**, passing the corresponding pre-defined constant for that option, and check if the value returned is non-zero.

FairCom DB API C API

- **ctdbGetSystemConfig()**

FairCom DB API C++ API

- **CTBase::GetSystemConfig()**.

7.17 Determine if FairCom DB API Records Sets are Active

New methods and functions were added to FairCom DB API to determine if record sets are active or not.

FairCom DB API C API

- **ctdbIsRecordSetOn()**

FairCom DB API C++ API

- **CTRecord::IsRecordSetOn()**

7.18 Get Table Status information From FairCom DB API

New methods and functions were added to FairCom DB API to retrieve the status of any changes performed on a FairCom DB API table. These return an indication of the actions the FairCom DB API alter table function will take on the table.

FairCom DB API C API

- **ctdbGetTableStatus()**

FairCom DB API C++ API

- **CTTable::GetStatus()**

7.19 Identifying Duplicate Key Index Errors

New FairCom DB API functions were added to determine which index caused an error after a failed call to **ctdbWriteRecord()**.

FairCom DB API C API

- **ctdbGetErrorIndex()**

FairCom DB API C++ API

- **CTRecord::GetErrorIndex()**

7.20 Exclusive Sessions and Databases in FairCom DB API

New functions and methods were added to a FairCom DB API session to allow the setting of the session exclusive flag and to permit the retrieval of the session exclusive flag. The session exclusive flag controls if the session dictionary file is opened in exclusive mode or not. When a session exclusive flag is set, a successful logon will only succeed if no other users are currently logged on to the same session dictionary. This behavior is an extension of an exclusive table open.

FairCom DB API C API

- **ctdbSetSessionExclusive()**
- **ctdbIsSessionExclusive()**
- **ctdbSetDatabaseExclusive()**
- **ctdbIsDatabaseExclusive()**

FairCom DB API C++ API

- **CTSession::SetExclusive()**
- **CTSession::IsExclusive()**
- **CTDatabase::SetExclusive()**
- **CTDatabase::IsExclusive()**

7.21 Get and Set FairCom DB API Table Owner Information

It is now possible to specify a table owner in FairCom DB API and have the **ctdbCreateTable()** function automatically invoke the FairCom DB **Security()** function to set the table owner. New FairCom DB API methods and functions are available to maintain the table owner information.

FairCom DB API C API

- **ctdbSetTableOwner()**
- **ctdbGetTableOwner()**

FairCom DB API C++ API

- **CTTable::SetOwner()**

- **CTTable::GetOwner()**

7.22 Set Integer Values with FairCom DB API CTMoney Class

A new method was added to the *CTMoney* class to enable the assignment of an integer value to a *CTMoney* object.

- **CTMoney::SetInt()**

7.23 New Mode for Table Rebuilds with Missing Index Files

A new open mode of *CTOPEN_DATAONLY* was introduced to the FairCom DB API API. When a **ctdbOpenTable()** is passed with the *CTOPEN_DATAONLY* mode, FairCom DB API can invoke a full alter table call and rebuild all the indices, if they are missing or not.

Previously, FairCom DB API could only rebuild a table with the table opened with the *CTOPEN_CORRUPT* mode and perform a full alter table by calling **ctdbRebuildTable()** or by implicitly calling **ctdbAlterTable(***CTDB_ALTER_FULL***)**. While this method works quite well when a table is corrupt, it does not work if at least one of the index files were missing.

7.24 Set Operations State in FairCom DB API

FairCom DB allows an application to change its operations state. Many advanced performance enhancing features can be accessed through the c-tree Plus function **SETOPS()**. FairCom DB Set Operations support was added to the FairCom DB API APIs. Additionally, specific support was added for setting automatic commit functionality within the FairCom DB API *CTBase* class with this **SETOPS()** implementation.

FairCom DB API C API

- **ctdbSetOperationState()**
- **ctdbGetOperationState()**

FairCom DB API C++ API

- **CTBase::SetOperation()**
- **CTBase::GetOperation()**

7.25 FairCom DB API Transaction Begin Modes

FairCom DB offers a rich array of data integrity options. Full transaction processing offers the safest and best performance of all the available options. There are times when other FairCom DB options, such as *PREIMG*, might be advantageous. FairCom DB API has been enhanced to accept the following transaction processing modes:

- *CTBEGIN_NONE* - No begin transaction mode set. This is the default mode.

- *CTBEGIN_PREIMG* - Transaction atomicity only. Auto-recovery is not available. Mutually exclusive with *CTBEGIN_TRNLOG*.
- *CTBEGIN_TRNLOG* - Full transaction processing functionality including auto-recovery. Mutually exclusive to *CTBEGIN_PREIMG*. This is the default begin transaction mode.
- *CTBEGIN_DEFER* - Defer begin transaction until update.
- *CTBEGIN_AUTOSAVE* - Automatically invokes savepoints after each successful record or resource update.

FairCom DB API C API

- **ctdbGetTransactionMode()**
- **ctdbSetTransactionMode()**

FairCom DB API C++ API

- **CTBase::GetTransactionMode()**
- **CTBase::SetTransactionMode()**

7.26 FairCom DB API Functions to Start and Stop the c-tree Server Engine

FairCom DB can be supplied in a dynamically loadable module, called only on demand. To fully utilize the dynamically loadable approach, FairCom DB API functions were added to start and stop the database engine.

FairCom DB API C API

- **ctdbStartDatabaseEngine()**
- **ctdbStopDatabaseEngine()**

FairCom DB API C++ API

- **CTSession::StartDatabaseEngine()**
- **CTSession::StopDatabaseEngine()**

8. Advanced Functions for the FairCom DB SDK

Introduction

The FairCom DB SDK is where you find the power and control you need to build the most advanced database applications. Providing direct calls into the core of the FairCom DB database engine provides a flexibility not found anywhere else.

The ability to block access to files, and quiesce FairCom DB are now available directly from your application. Partial Record Rewrites and advanced new caching strategies bring additional performance. New permanent callback filters provide a solid security feature for the most demanding of applications which need to prevent data retrieval at the lowest levels. Many great new FairCom DB features are described in this section and you are invited to explore them all in your next database project.

8.1 Block Access to Files

There are situations where it would be extremely advantageous to completely block access to an open data file by current users. For example, consider the case where an administrator requires immediate access to a data file to compact the file as the file system is near capacity.

Typically in this situation, the application is brought down, as the compact operation requires exclusive access to the data file and indices. A better interactive approach may be to simply hold users out of the file while the operation takes place and allow them to “re-open” the file afterward. The new File Block and Quiet FairCom DB API calls gives you this ability.

- **ctFILBLK**
- **QuietCtree**

8.2 Blocking Lock Timeouts

Timeouts, measured in seconds, can now be specified for blocking lock requests made on data files. There are various ways the timeout can be specified, and in all of them, a zero timeout indicates NO timeout, not a zero timeout. If a lock request fails because of a timeout, a new error code is returned: **UTIM_OUT** (827).

- **ctLOKTIMOUT**

8.3 Partial Record Rewrites

When updating records with the FairCom DB ISAM or FairCom Low-Level APIs, the entire record is passed back to c-tree when rewriting, even when only a few bytes at the beginning of the record are modified. When working with large records a considerable performance gain may be obtained if the update to the entire record buffer is avoided and instead, only a smaller portion of the record is rewritten.

- **ReWritePartialRecord**

8.4 Row-Level Permanent Callback Filters

The FairCom DB API function **SetDataFilter()** can be used to establish a temporary, user-specific, filter for data record access (see “Data Filters and Conditional Indexes” in the *FairCom DB Programmer’s Reference Guide*). Consider the case of requiring a customized security audit trail. Individual record reads and writes can be monitored and logged with these permanent filters in place. An application could also use this feature to restrict a user’s ability to read and/or write records based on the row from the table of interest, thereby enabling a row level security mechanism. The username, table, and operation attempted, including the record image, are all data available to your application. **SetDataFilter** now supports establishing a new type of permanent system-wide filter.

- **ctfilter_init**
- **ctfiltercb_rowl**
- **ctfilter_uninit**

8.5 Advanced Cache Alternative for Scanning Data Files

The data cache and index buffer pages are managed using a least-recently-used (LRU) scheme such that the page that has not been touched the longest will be the page selected for re-assignment when a page is required to satisfy a current request. For indices, this simple strategy works very well since traversing an index tree frequently requires access to the same, important high level nodes (such as the root node) and/or to the “current” leaf node.

For data files, the LRU scheme is not always effective. In particular, when data file access or updates involve many different records with little or no revisiting of records once they have been processed, the LRU scheme can result in many cache pages to be assigned to recently accessed records. But, at least from the user’s perspective, there is little chance of revisiting these pages.

To address these situations, FairCom has devised an alternative cache strategy that provides an efficient new method of accessing the cache. The *ctSCANCACHEhdr* mode allows this new caching strategy.

The new strategy uses all the standard cache routines except when requiring a cache page (that is, c-tree has not found the data block it needs already in cache), it uses one of two dedicated cache pages.

The significance of this strategy is that a large set of data record operations can be performed with only two cache pages with little or no decrease in performance compared to no restrictions

on cache usage. Moreover, such restricted use may help other aspects of system performance since the bulk of data cache can be used for other files and/or other users. A large traversal of the data file will not replace other cache pages.

The following call will enable the scanner mode of the file to use the new cache strategy:

```
PUTHDR(datno, YES, ctSCANCACHEhdr);
```

8.6 Recursive Locking Support

FairCom introduces an additional locking mode, recursive locks. By recursive lock, we mean a lock applied to a record by a user already owning the same lock on the same record, and such an additional lock request is “counted.” Normally for an unlock call, c-tree removes any record locks, regardless of the number of times it was locked.

Recursive locks are typically useful for an application that must make lock and unlock calls without knowledge of existing locks on the same record by the same application. Recursive locks must be requested explicitly through a lock mode. The following API calls support OR-ing in the new lock mode bit, *ctLK_RECR*:

- **Begin**
- **Commit**
- **LockISAM**
- **LockCtData**

8.7 Retrieve a List of File names from FairCom DB

A new c-tree API function was added to enable a means to retrieve a list of filenames from FairCom DB. These files can be any c-tree or non-tree file, including c-tree memory files.

- **ctFILWCD**
- **FindFileByName**

8.8 File-based Global Mutexes

A new feature called a File-based Global Mutex enables support for a new type of synchronization object. It is file-based as the mutex uses a system lock on the designated file to establish mutex ownership, and it is global because it synchronizes access across multiple processes. These processes may be stand-alone c-tree executables or c-tree servers.

Consider two independent c-tree applications in one “environment”. By environment, we mean multiple c-tree applications, either on one machine or multiple machines, but with a single available data source. A file based mutex could protect against multiple unsynchronized attempts to access to this data file. An application developer could prevent unintended environment configurations from causing data integrity problems with this global based mutex.

- **ctFILMTX**

8.9 Programmatically Import Tables into FairCom DB SQL

The c-treeDB API provides a function call to dynamically import an existing ISAM table into FairCom DB SQL programmatically. This same ability is now available in the FairCom DB ISAM API.

- **ctSQLImportTable**

8.10 Access a FairCom DB Connection Created by Another Thread

To enhance multi-user interaction between multiple threads an advanced FairCom DB function has been added. When using the c-tree multi-threaded client library, and one thread initializes c-tree and another thread attempts to use that connection, c-tree calls fail with an access violation. In the multi-threaded client operational model, a c-tree connection is associated with the thread that established the connection.

- **ctSetOWNER**

8.11 List Users Owning or Waiting for Record Locks

It is sometimes necessary to identify users who hold a record lock. For example, when profiling performance issues, record locks held longer than necessary are frequently identified as the culprit. It is useful to identify the owner of the record lock to determine the nature of these locks.

To provide additional diagnostic information to the application server administrator, a FairCom DB function was added to retrieve a list of users that own, and are waiting for, a data record lock at a specified byte offset of a data file.

- **LockList**

8.12 Extended Connection Information in a Lock Dump

The **ctLOKDMP()** function is used to log information about currently-held locks. This function now logs the task ID and node name for each connected client, such that the task ID shown in the lock information can be easily tracked to the corresponding node name for the connection.

8.13 TCP/IP Client Connect and Communication Timeout Options

The FairCom DB client library now supports a timeout on TCP/IP socket send and receive operations. The timeout is configured on a per-connection basis and is disabled by default. To set a socket timeout, after connecting to FairCom DB, set the FairCom DB connection-specific global variable *ctsocktimeout* to the desired timeout value in seconds. If a TCP/IP send or receive operation blocks for more than the specified number of seconds, the c-tree client returns a fatal

communication error **ARQS_ERR** (127) or **ARSP_ERR** (128). This timeout value can be changed at any time, and takes effect on the next socket call.

To facilitate detection of this condition, two new error codes have been introduced and are returned when a socket operation times out:

Symbolic Error Code	Error Code	Explanation
TRQS_ERR	808	Request timed out.
TRSP_ERR	809	Response timed out.

8.14 Additional Security Administrator Functions

A new Security Administrator (SA ADMIN) function was added to allow for filename searches using wildcards using the SA ADMIN API.

- **SA_WILDCARD**

A new function was added to the Security Administrator API to display the error message text for the specified SA API error code.

- **SA_ERRMSG**

8.15 Increased Page Size Now Available

The maximum page size available has been increased from 32768 to 65536. A page size of 8192 remains the default FairCom DB setting. It is also noted that page sizes less than 1024 are no longer supported by default.

The FairCom DB configuration keyword `PAGE_SIZE` is used to change this setting should you require this.

Contact your nearest FairCom office if you require support for page sizes less than 1024. For additional information, please refer to “Miscellaneous Control” in the *FairCom DB Administrator’s Guide*.

8.16 Extended Version of Compact Function Added

The extended header version of the compact function has been added as **CMPIFILX8()**

The existing **CMPIFILX()** API call does not facilitate an array of *XCREblk*s to be passed in. When the indices associated with the data file to be compacted exist, then compact generally does not need these *XCREblk*'s used when the indices were created. However, if one or more of the indices does not exist, then **CMPIFILX()** recreated a *XCREblk* from the data file. This does not account for differences between the original data file *XCREblk* and the original index file *XCREblk*.

Additionally, when either **CMPIFILX()** or **CMPIFILX8()** are run on the server (as opposed to stand-alone), unless the server configuration option `COMPATIBILITY 6BTRAN_NOT_DEFAULT` is specified, then the *ct6BTRAN* mode bit is turned on in the recreated *XCREblk* or the explicit *XCREblk*, respectively, to ensure 6-byte transaction number support.

This new API call declaration is:

```
ctCONV COUNT ctDECL CMPIFILX8(pIFIL ifilptr,pTEXT dataextn,pTEXT indxextn,LONG permmask,pTEXT groupid,
pTEXT fileword,pXCREblk pxcreblk)
```

Note: It should be noted that since **CMPIFILX()** reconstructs an *XCREblk* for the index from the data file, there may be attributes of the data file that were not part of the original index file definition. For example, if the data file supported a large extent size, then the index inherits this large extent size regardless of how it was originally created.

8.17 Compact Function Now Supports Duplicate Key Purge and Update IFIL Options

The FairCom DB file compact function **CompactIFile()** now supports two features that the file rebuild function **RebuildIFile()** supports:

- Purging of records having illegal duplicate keys.
- Updating the *IFIL* resource in the data file with the *IFIL*/*IIDX*/*ISEG* structure definitions passed to **CompactIFile()**.

These features are used in the same way they are used when calling **RebuildIFile()**.

8.18 Additional SETOPS Modes

Heterogeneous Platform String Conversion

An additional **SETOPS()** mode, *OPS_CONV_STRING*, was introduced that optionally enables the client logic to scan a c-tree data file's field definitions when the client code reads the *DODA* from the data file. This is useful in a heterogeneous platform environment.

This conversion is disabled by default. You can enable this behavior from your application with the following FairCom DB call:

```
SETOPS(OPS_CONV_STRING, OPS_STATE_ON);
```

Automatic Low-level, Blocking Read Locks

The set operations function, **SETOPS()**, has an additional mode to enable automatic, low-level, blocking read locks on each record access that does not already have a lock. The locks are released automatically as soon as the record is read. Blocking locks permit a c-tree application to be coded the same whether or not the automatic locks have been requested. But this means a read will block until the read lock is available.

```
SETOPS(OPS_READLOCK,opcode)
```

8.19 Default Temporary File Path in Standalone and LOCLIB Models

FairCom DB supports the ability to set a default path for c-tree temporary files in standalone or *LOCLIB* mode. This feature is useful for setting the path for temporary files created when rebuilding or compacting c-tree files. FairCom DB supports this ability with the configuration keyword `TMPNAME_PATH`.

To use this feature, first initialize FairCom DB and then set the c-tree global variable *ct_tmppth* to point to a buffer containing the name of the desired temporary path. Include the path separator at the end of the path name. The buffer may be dynamically allocated. If so, you must also free the buffer when it is no longer needed and set *ct_tmppth* to NULL.

8.20 Options to Retrieve a File's Unique ID

Three new modes have been added to permit **GetCtFileInfo()** (short name **GETFIL()**) to be called to return each of the three components comprising the 12-byte Unique file ID assigned to each file at creation. The ID is comprised of a FairCom DB ID, a time stamp and a sequence number, each 4-bytes long.

GETFIL() can now be called with these new modes:

- *SERVID* -- returns server ID
- *TIMEID* -- returns time stamp
- *FILEID* -- returns sequence number

8.21 Update Unicode Version

A new FairCom DB call has been implemented to avoid the restriction which occurs when a UNICODE library version does not match the version in effect when the file was created. The open file logic ensures the ICU versions match and the rebuild function has no way to override the version check.

- **ctUpdateICUversion**
- **ctUPDATICU**

8.22 Enforce Maximum Disk Read/Write Sizes on Windows

A large variable-length record write operation could fail with Windows error 1450 ("Insufficient system resources exist to complete the requested service") when the data file resided on a network drive. c-tree normally writes the record in a single call to the **WriteFile()** Win32 API function. To avoid this error, c-tree supports setting a maximum disk read and write size. Read or write operations that exceed this size are performed as a series of reads or writes of the specified maximum size.

To set a maximum disk read and write size, compile c-tree with #define ctMAX_IO_SIZE <maxbytes>, where *<maxbytes>* is the maximum number of bytes c-tree will read or write in a single system call.

Note: #define ctMAX_IO_SIZE is supported on Windows systems only and has no effect on other platforms.

8.23 Other New Functions Available

FairCom DB has more than a dozen new functions available for a wide range of new functionality. Some of these new functions are quite powerful, such as **ctFILBLK()** and **ctQUIET(),** giving developers advanced control over file maintenance. The Callback functions are available in the advanced FairCom DB Server SDK allowing you absolute control over your data.

You can find complete descriptions of all of these new functions in the updated and revised *FairCom DB Programmer's Reference Guide*, Appendix F "Function Reference Guide".

- **SetSystemConfigurationOption()** - Dynamically changes server configuration settings at runtime.
- **ctFileCLoseCallback()** - Custom close file callback stub for the c-tree Server SDK. (Located in the *ctuserx.c* module.)
- **ctFileCreateCallback()** - Custom create file callback stub for the c-tree Server SDK. (Located in the *ctuserx.c* module.)
- **ctFileOpenCallback()** - Custom open file callback stub for the c-tree Server SDK. (Located in the *ctuserx.c* module.)

9. More c-treeVCL Functionality

Introduction

While Borland has continued developing new frontiers with its CodeGear tools, FairCom DB API VCL remains available for this great application development environment. Bringing FairCom DB power to your Borland projects has never been easier. Drag-and-drop our components into your next CodeGear database project.

9.1 New c-treeVCL Features

Working with Resources in c-treeVCL

FairCom DB Resources are a convenient feature allowing extraneous data to be placed into a c-tree data file. Consider application state or version information. With a c-tree resource the application can easily retain information between instances directly in the main data file, without having to create a specialized record requiring exceptional handling. c-treeVCL has been enhanced with support for these c-tree resources. A new class, *TCtResource* has been created with all of the properties and methods necessary for resource handling. You can find all of the information in the updated *c-treeVCL Developer's Guide* (https://docs.faircom.com/doc/*ctreevcl/*).

Renaming Tables

A new method, **RenameTable()**, has been added to the *TCtDatabase* class to rename tables. This procedure takes as parameters the old and the new name as strings.

```
TCtDatabase.RenameTable(FOldName : string; FNewName : string);
```

The table must be closed and opened with exclusive access to rename the physical file resources.

Key Counting Methods

It is frequently helpful to return a periodic status to the user in the case of long searches, or estimations of how long a search will take. With c-tree Plus, you could periodically get the **CurrentISAMKey()** of the last retrieved record, then use **ESTKEY()** to determine its approximate ordinal positioning in the index and that, divided by **NbrOfKeyEntries()** provided a % complete. c-treeVCL now implements the c-treeDB corresponding functions, **ctdbNumberOfKeyEntries()** and **ctdbEstimateSpan()**.

NumberOfKeyEntries() and **EstimateSpan()** methods have been added to the *TCtRecord* class.

Index Key Length Information

A *KeyLength* property has been added to the *TCtIndex* class for convenience in obtaining the index key length for the index.

10. More Options with FairCom DB Utilities

Introduction

To take advantage of many of the new FairCom DB features, several existing FairCom DB utilities have been updated with new options. As always with the FairCom DB SDK, you have complete source code available to see how these new features were implemented and take advantage of them directly in your applications today.

10.1 Enhanced FairCom DB Statistics Utility Features

The FairCom DB Statistics Utility, **ctstat**, is a client utility used to display statistics collected by the FairCom DB. **ctstat** provides valuable real time monitoring of critical FairCom DB operations via the advanced *SNAPSHOT* feature. Many enhancements have been made to **ctstat** to allow for more robust gathering of vital FairCom DB information.

- **Show memory in use by memory files**
- **Previously established users now displayed**
- **Enable I/O Statistics Options per File**
- **Output I/O Statistics Options per File**
- **Report options for existing connections**
- **Enable c-tree Function Call Times by File**
- **Output c-tree Function Call Times by File**
- **Display Date and Time in the Header**
- **Initial delay is now avoided between intervals**

10.2 Additional FairCom DB SQL Table Import Utility Options

The FairCom DB SQL Table Import Utility, **ctsqlimp**, is a versatile utility to import existing c-tree data and index files for use with FairCom DB SQL. This makes it possible, for example, to continue to use your existing traditional ISAM applications with enhanced industry standard SQL backend reporting. Several new enhancements have been made to this utility to help developers import a broader range of c-tree data.

- **Specifying Primary Keys From Imported Indexes**
- **Mapping Unsigned Integers With FairCom DB SQL**
- **Piped Input Is Now Accepted**
- **NOT NULL Field Attribute Retained On Import**

10.3 Enhanced Security Administrator Usage and Options

The security administrator utility, **sa_admin**, now supports an alternative syntax for specifying options when creating user accounts, changing extended user settings, creating groups, and changing group memory settings. This new syntax is designed to be more intuitive and useful for security administrators. A user can choose to specify only the desired options, and they can be specified in any order. Prior to this change, all options had to be specified and they had to be specified in a particular order.

- **Improved Command Line Options Handling**
- **Support for Encrypted Password Files**
- **Retrieve a List of Filenames**
- **Use Wildcard Specifiers**
- **Improved File Permissions Usage**

10.4 c-tree Information Utility Enhancements

The FairCom DB Information Utility, **ctinfo**, is a valuable tool to examine many c-tree data and index file attributes. Several enhancements were applied to the c-tree *IFIL* and *DODA* extraction utility, **ctinfo**.

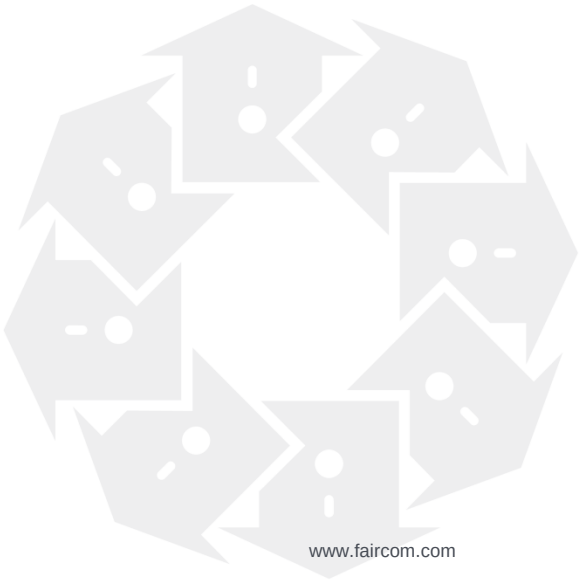
- **List all files open by FairCom DB.**
- **List all files open by a particular connection.**
- **List connections that have a particular file open.**
- **List locks held on a particular file.**
- **Display 64-bit logical and physical file sizes.**
- **Output individual extended file mode attributes.**
- **The -rdkeys option has been enhanced to properly handle HUGE files.**
- **Now opens the specified file in shared read-only mode.**
- **Display complete XCREblk information.**
- **Display details about a conditional index resource when it is present.**

11. FairCom Typographical Conventions

Before you begin using this guide, be sure to review the relevant terms and typographical conventions used in the documentation.

The following formatted items identify special information.

Formatting convention	Type of Information
Bold	Used to emphasize a point or for variable expressions such as parameters
CAPITALS	Names of keys on the keyboard. For example, SHIFT, CTRL, or ALT+F4
FairCom Terminology	FairCom technology term
FunctionName()	FairCom DB Function name
Parameter	FairCom DB Function Parameter
`Code Example`	Code example or Command line usage
utility	FairCom DB executable or utility
filename	FairCom DB file or path name
`CONFIGURATION KEYWORD`	FairCom DB Configuration Keyword
CTREE_ERR	FairCom DB Error Code



Copyright Notice

Trademarks

FairCom DB, FairCom EDGE, c-treeRTG, c-treeACE, c-treeAMS, c-treeEDGE, c-tree Plus, c-tree, r-tree, FairCom, and FairCom's circular disc logo are trademarks of FairCom USA, registered in the United States and other countries.

The following are third-party trademarks: Btrieve is a registered trademark of Actian Corporation. Amazon Web Services, the "Powered by AWS" logo, and AWS are trademarks of Amazon.com, Inc. or its affiliates in the United States and/or other countries. AMD and AMD Opteron are trademarks of Advanced Micro Devices, Inc. Macintosh, Mac, Mac OS, and Xcode are trademarks of Apple Inc., registered in the U.S. and other countries. Embarcadero, the Embarcadero Technologies logos and all other Embarcadero Technologies product or service names are trademarks, service marks, and/or registered trademarks of Embarcadero Technologies, Inc. and are protected by the laws of the United States and other countries. HP and HP-UX are registered trademarks of the Hewlett-Packard Company. AIX, IBM, POWER6, POWER7, POWER8, POWER9, POWER10 and pSeries are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. Intel, Intel Core, Itanium, Pentium and Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. ACUCOBOL-GT, Micro Focus, RM/COBOL, and Visual COBOL are trademarks or registered trademarks of Micro Focus (IP) Limited or its subsidiaries in the United Kingdom, United States and other countries. Microsoft, the .NET logo, the Windows logo, Access, Excel, SQL Server, Visual Basic, Visual C++, Visual C#, Visual Studio, Windows, Windows Server, and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Oracle and Java are registered trademarks of Oracle and/or its affiliates. QNX and Neutrino are registered trademarks of QNX Software Systems Ltd. in certain jurisdictions. CentOS, Red Hat, and the Shadow Man logo are registered trademarks of Red Hat, Inc. in the United States and other countries, used with permission. SAP® Business Objects, SAP® Crystal Reports and SAP® BusinessObjects™ Web Intelligence® as well as their respective logos are trademarks or registered trademarks of SAP. SUSE" and the SUSE logo are trademarks of SUSE LLC or its subsidiaries or affiliates. UNIX and UNIXWARE are registered trademarks of The Open Group in the United States and other countries. Linux is a trademark of Linus Torvalds in the United States, other countries, or both. Python and PyCon are trademarks or registered trademarks of the Python Software Foundation. isCOBOL and Veryant are trademarks or registered trademarks of Veryant in the United States and other countries. OpenServer is a trademark or registered trademark of Xinuos, Inc. in the U.S.A. and other countries. Unicode and the Unicode Logo are registered trademarks of Unicode, Inc. in the United States and other countries.

All other trademarks, trade names, company names, product names, and registered trademarks are the property of their respective holders.

Open Source Components

Like most software development companies, FairCom uses third-party components to provide some functionality within our technology. Often those third-party components are selected because they are a standard in the industry, they offer specific functionality that is easier to license than to develop and maintain in the long run, or they provide a proven and inexpensive solution to a particular business need. Examples of third-party software FairCom uses are the OpenSSL toolkit that provides Transport Layer Security (TLS) for secure communications and the ICU Unicode libraries to provide wide character support (think international characters and emojis).

Some of these third-party components are the subject to commercial licenses and others are subject to open source licenses. For open source solutions that we incorporate into our technology, we include the package name and associated license in a notice.txt file found in the same directory as the server.

The notice.txt file should always stay in the same directory as the server. This is particularly important in instances where your company has redistribution rights, such as an ISV who duplicates server binaries and (re)distributes those to an eventual end-user at a third-party company. Ensuring that the notice.txt file "travels with" the server binary is important to maintain third-party and FairCom license compliance.

1/30/2025

12. Index

V

W

X