test link

Update Guide

FairCom RTG V2 Update Guide

Audience

Developers

Subject

New features in FairCom FairCom RTG V2

FairCom®

Contents

1. FairCom RTG Version 2 Highlights

Field-proven, FairCom RTG is used day-after-day by major corporations to bring their legacy systems up to the latest standards of performance, stability, and data integrity.

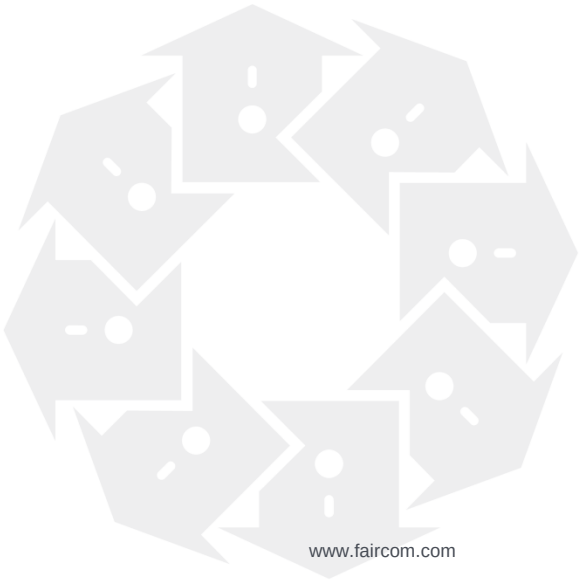
FairCom RTG V2 delivers enhancements inspired by real-world experience with applications like yours.

FairCom RTG Unleashes the Value of Your Legacy Applications

FairCom RTG replaces COBOL and Btrieve file handling with an enterprise-class data management system. With ACID transactions, real-time backups, and automatic recovery, it improves the performance, reliability, and data integrity of your application. In addition, industry-standard SQL interfaces unlock your data to modern applications.

FairCom created FairCom RTG with three main goals in mind:

- **SQL Access** - FairCom RTG unlocks your data with full read/write access through SQL and other relational interfaces—without affecting COBOL read/write access to the same data.
- **High Availability** - ACID transactions, real-time backups, and automatic recovery, improve data availability, scalability, and data integrity.
- **Robust Data Management** - FairCom RTG is a data-management system. It allows COBOL and Btrieve applications to leverage the acclaimed FairCom RTG database in a client/server configuration.



Connected Data. Countless Possibilities.

In the era of Big Data, integration of legacy systems has become an increasing pain point for companies. Many solutions help you collect and analyze new incoming data to get critical business insights. As important as it is, that data is only the tip of the iceberg. A valuable source of data lies beneath the surface.

Much of the world still runs systems built on COBOL and Btrieve. Over years of operation, these systems have amassed a wealth of data about your business. Data that could be turned into a competitive advantage if you could access it.

That data is isolated by the limited connectivity of those programming languages, making it extremely difficult for modern applications to access. SQL and other relational interfaces are needed to connect your data to modern analytical tools and business intelligence (BI) applications.

At FairCom, we understand this real-word scenario. The first version of FairCom RTG, in 2012, introduced technology that made it easy to get up-and-running and begin leveraging your data. It began as a revolutionary, “hands-off” approach that can be implemented with little or no changes to your application.

FairCom RTG V2: The Power of Connection

FairCom RTG V2 integrates with your legacy applications and connects them to the modern world. It empowers your applications in two ways:

- **Integration with Your Application**

  FairCom RTG seamlessly replaces your file system with robust FairCom client/server technology, which has been proven in mission-critical installations around the world.

  This release integrates with your applications more tightly than ever. It updates your legacy application with increased stability, high availability, and scalability by adding ACID transactions, real-time backups, and automatic restores.

- **Integration with Other Applications**

  FairCom RTG connects your data to the world of reporting and analytical applications through a set of relational SQL interfaces.

  This release provides greater SQL capabilities for more powerful integration with data. You can create SQL indices for COBOL data so you can better handle data that does not easily translate into the relational SQL model. And our BTRV product now makes SQL interfaces available for your Btrieve data.

This version of FairCom RTG delivers COBOL and BTRV solutions that are easy to get up-and-running and more powerful at delivering results. All while maintaining the core FairCom RTG commitment, by keeping it easy-to-use.

All your data is connected and brought into focus, available to return a 360-degree perspective for meaningful, actionable insight to business.

With connected data, the possibilities are countless.

1.1 SQL Leverages the Value of Your Data

Unlock the value of your COBOL and Btrieve data with relational access through SQL, ODBC, JDBC, ADO.NET, etc.

Customers cite many reasons for their loyalty to FairCom RTG, including its ability to bring new performance, reliability, and data integrity to their existing applications. Certainly the most valuable FairCom RTG feature is its ability to provide seamless SQL access to your COBOL and Btrieve data. With SQL access, FairCom RTG makes your data available to the industry's latest business intelligence, analytical, and reporting tools.

SQL Indexes on Your COBOL Tables Improve Performance

With Version 2, FairCom RTG makes another breakthrough in its SQL technology: Now you have the ability to execute CREATE INDEX on imported COBOL tables. Some COBOL indexes do not translate well to SQL—this new feature allow you to create SQL indexes on this data.

This much-requested new feature improves SQL performance. When a COBOL index has not been sqlized properly, the performance of SQL queries on those fields can be impacted. This feature allows you to simply create SQL indexes where needed to speed up queries.

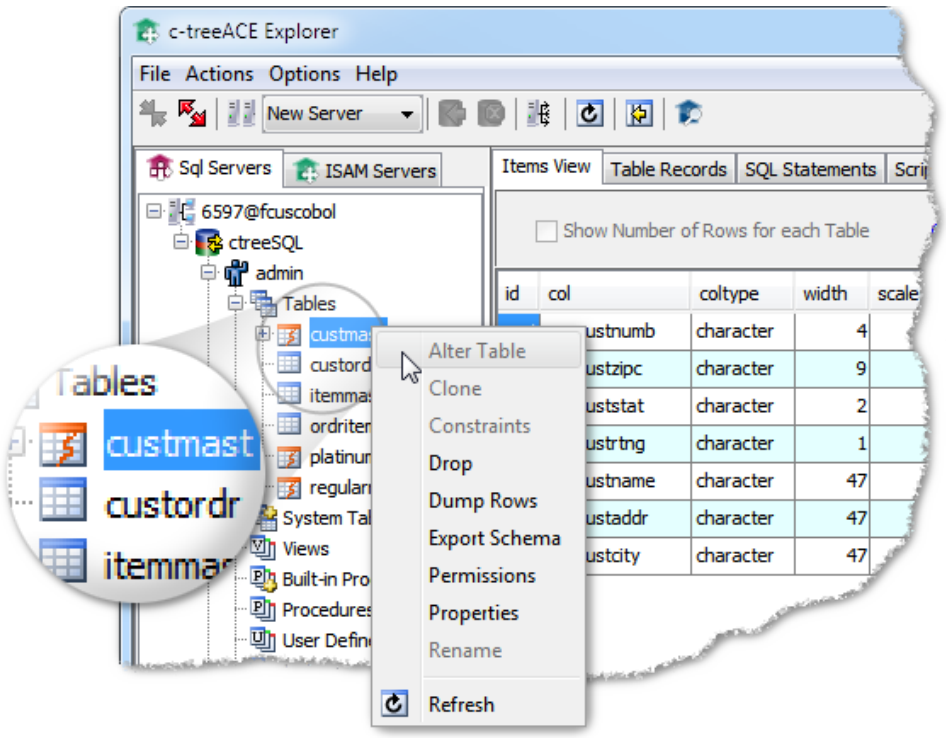
Learn more... (page 29)

Navigate Your SQL Data with SQL Explorer

FairCom RTG opens your FairCom product data to the world of SQL access. FairCom knows you will want to monitor and manage this data—and that means seeing it the way it appears to SQL applications. c-treeACE SQL Explorer and c-treeACE Explorer are graphical tools that allow you to view and manipulate your data, providing comprehensive management capabilities.

c-treeACE SQL Explorer and c-treeACE Explorer are client tools designed to interact with FairCom RTG Servers through the TCP/IP SQL port. Using the SQL language, they give you access to all the items in your c-treeACE database. They also provide the ability to execute single SQL statements and SQL scripts including options for viewing the execution plans.

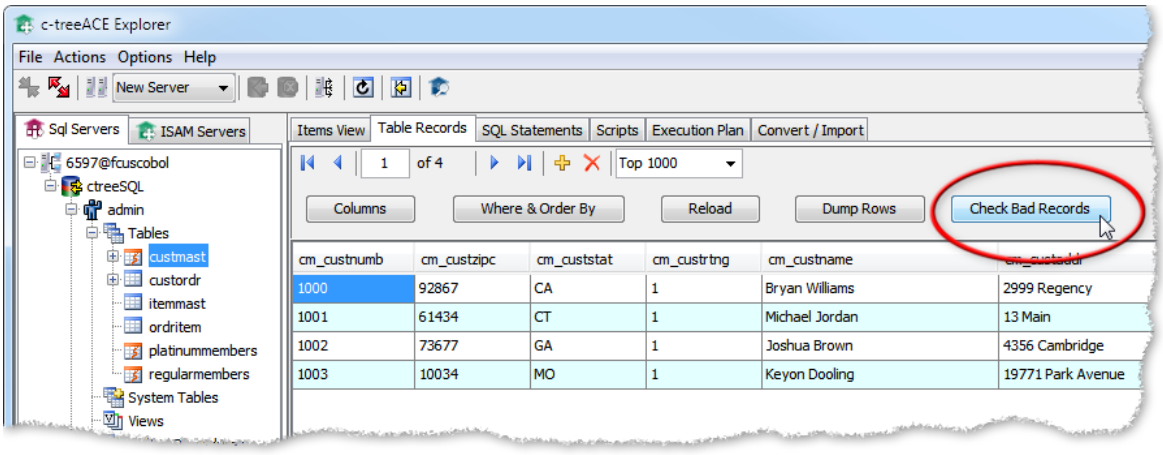
c-treeACE SQL Explorer and c-treeACE Explorer show linked tables in a different color (a reddish orange). Tables that were also sqlized have a jagged "**S**" (or lightning bolt) to indicate they were sqlized and linked. The **custmast** table in the image below is a sqlized and linked table:



When you right-click a table, the context menu will display only the options that are available to that table (all other options are dimmed), as shown in the image above. Some options available to regular tables, such as Alter Table, Clone, **and** Constraints**, are disabled because they are not available for sqlized tables.**

Checking for Bad Records

Some tables may contain records that have conversion problems, so you will want to check to see if the conversion was performed properly. FairCom RTG V2 makes this step in the conversion process easier: Simply select a sqlized table, click the Table Records tab, and click **Display Bad Records**. A test query will be performed and all records with data problems will be displayed.

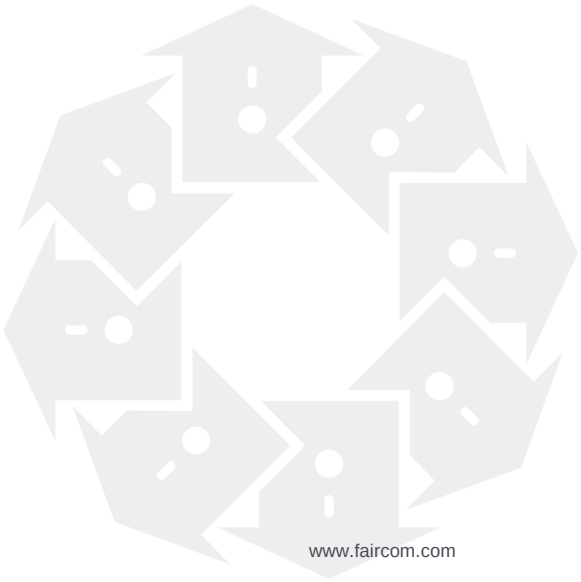


1.2 Tighter Integration with Your COBOL Applications

FairCom RTG V2 tightly integrates with your COBOL application for the best performance and data integrity ever.

This release of FairCom RTG implements many enhancements aimed at seamlessly integrating with your COBOL application. The list of new and improved features in FairCom RTG V2 goes on and on. Here is a brief summary:

- **Improved updates during scans on duplicate index** (page 93)
  When FairCom RTG scans a file with an index that allows duplicates, a record that was changed without updating the key used for scanning is now updated only once.
- **File organization specification allows you to fine-tune COBOL integration** (page 37)
  A new <file> configuration attribute makes it possible to specify the file organization, such as indexed file, relative file, line sequential file, record sequential file, etc.
- **File locking behavior conforms to ACUCOBOL's locking** (page 32)
  Integration with ACUCOBOL is fine-tuned like a Swiss watch. The <runitlockdetect> setting now enforces record locks as well as file locks so it is possible, for example, to exclusively open a file twice from the same run unit.
- **Better handling of index re-organization errors due to redefined schema** (page 32)
  Improvements in V2 better handle the situation when a directive redefines the schema and there is no valid combination of fields to compose a key.

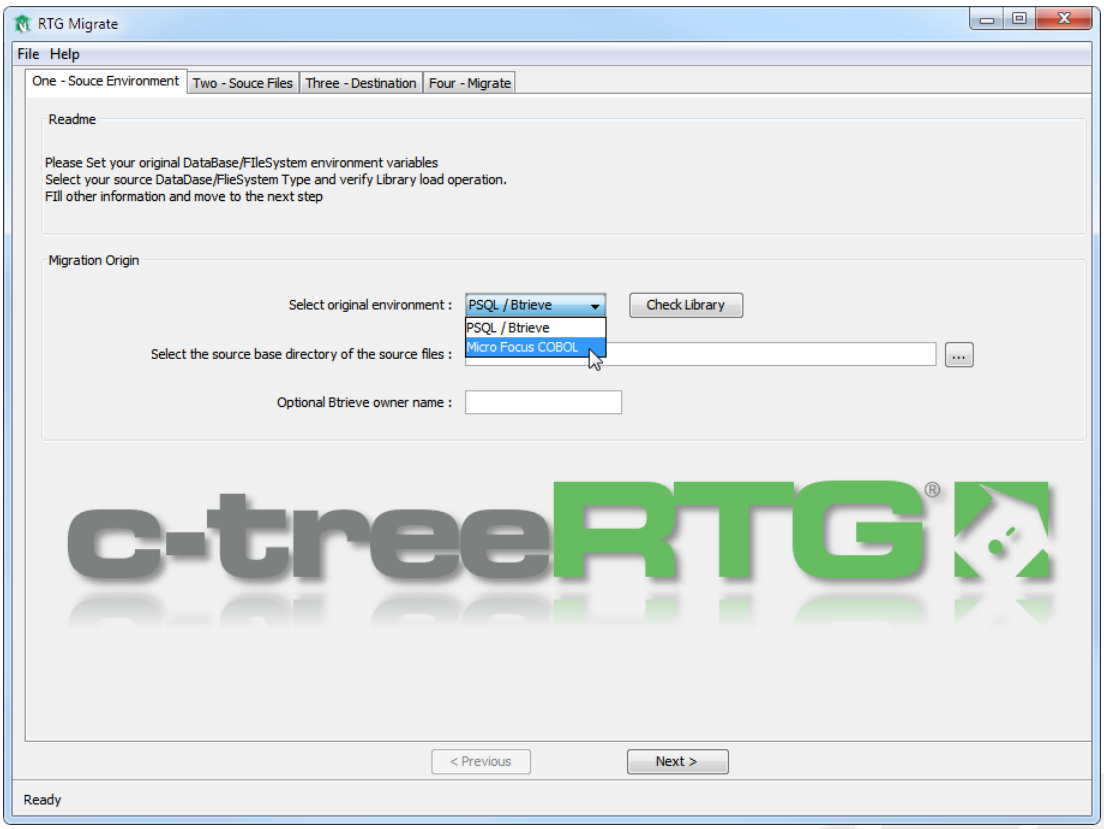


1.3 Ease of Use

FairCom RTG includes a set of productivity tools to help you get the most out of your applications.

RTG Migrate Helps You Move into FairCom RTG

The recently introduced RTG Migrate tool provides a graphical interface for migrating data into FairCom RTG applications. In Version 2, this tool has been updated to provide increased usability and improved features—and guide you through the process of "moving-in" to FairCom RTG.



Let RTG Migrate Guide the Way

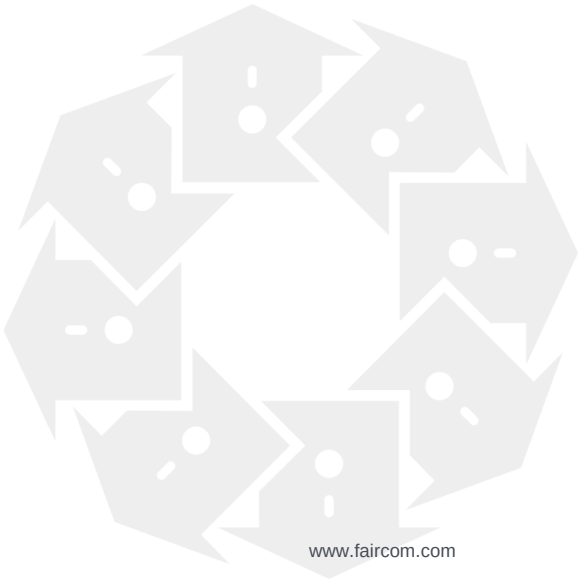
This tool can facilitate data migration from an existing file system into a format suitable for FairCom RTG in two different ways:

1. It can perform the data migration on one or multiple files.
2. It can create a script that calls the **ctmigra** command-line utility for one or multiple files.

Changes to the RTG Migrate tool include features designed to make migration easier:

- **Creates the ctree.conf file to help get you started** (page 46)
- **New options: encryption, data compression, and transactions** (page 46)

- **New "Append" option** (page 46)
- **Test Connection button** (page 46)
- **New casesensitive keyword and textfield to set the configuration file name** (page 46)



We Didn't Forget the ctmigra Command-Line Utility

The **ctmigra** command-line utility can be called by RTG Migrate or used stand-alone from the command-line. This utility is the engine that does the work of migrating your files to the format used by FairCom RTG. It can be called from a command-line in situations where this type of operation is more efficient than the RTG Migrate GUI tool. The GUI tool can be used to generate a script for running this utility. The changes to this utility add to the functionality of both tools.

Changes to **ctmigra** include:

- **Options to set encryption, data compression, and transaction processing** (page 46)
- **Detection of missing or invalid library** (page 49)
- **Options to replace existing files and disable batchaddition** (page 48)
- **Support for the ExtFH interface** (page 48)
- **Improved error messages** (page 49)
- **More ctmigra enhancements...** (page 50)

RTG Config Tool Gets You Going in a Hurry

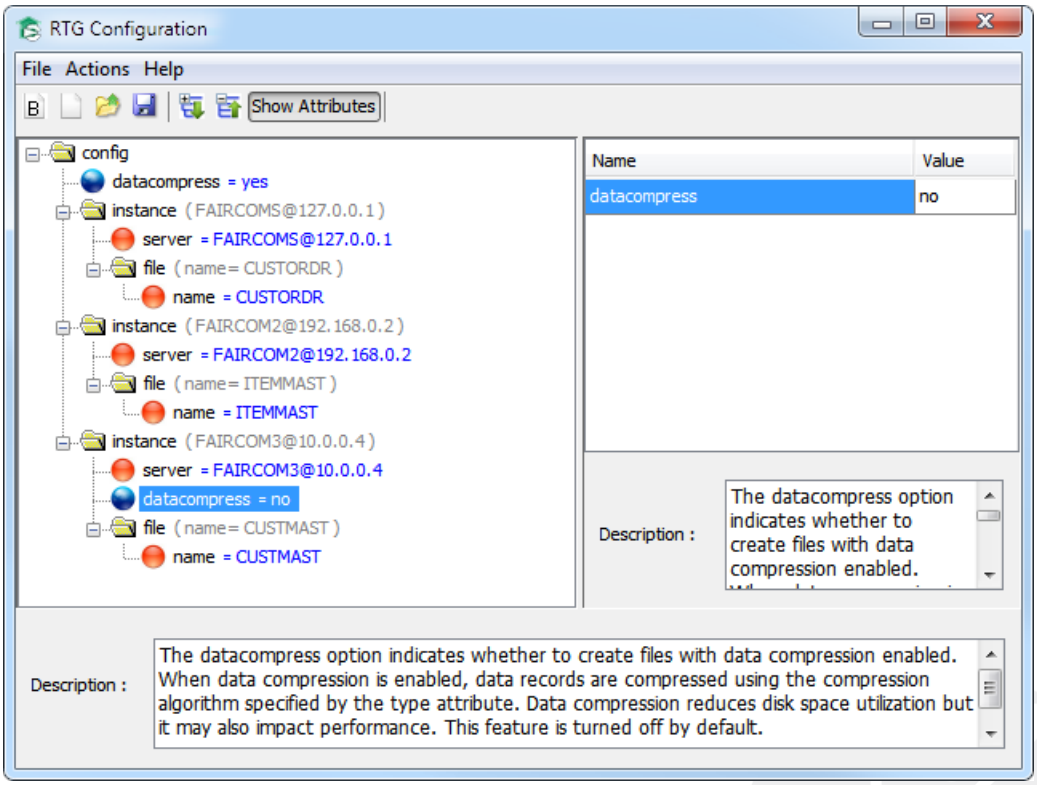
Configuration is essential to getting any system up-and-running. A proper configuration can mean the difference between acceptable performance and outstanding performance. In the extreme case, configuration can make the difference between running and not running.

The FairCom RTG configuration tool is designed to help you though the process of configuring your system for maximum performance. Considerable effort has been put into the RTG Config tool to simplify the task of setting up your system.

Client Configuration Tool

With FairCom RTG V2, FairCom puts a new device in your tool chest: The FairCom RTG Client Configuration Tool. This tool provides an intuitive user interface, prompting you for the best options to configure your FairCom RTG environment.

The FairCom RTG Client Configuration Tool is your way to establish the configuration necessary for the client to connect to FairCom RTG. The RTG Config tool simplifies the process of defining connection strings and file locations.



A new, **Basic Configuration** wizard has been created to help you get going in a hurry.

The list of updates to the RTG Config tool includes the features below:

- **Configuration Tool now recognizes "priority" and "casesensitivity" attributes** (page 36)
- **Option to dump configuration in XML format** (page 56)

New Configuration Options for the Perfect “Tune-Up”

Version 2 of FairCom RTG offers greater configuration flexibility to conform to the requirements of your environment. These options are all available when you are using the latest version of the RTG Config tool.

- **<locktimeout> option to set blocking lock timeout** (page 38)
- **<temporary> element to create files with reduced disk I/O** (page 43)
- **<trxholdslocks> option sets transaction lock behavior** (page 43)
- **<log><debug><transaction> option to debug transactions** (page 40)
- **<normalize> option to normalize file paths** (page 41)
- **<filecopy> uses server-side file copy capability to make a physical copy of data and index files** (page 39)
- **CTREE_CONF_DUMP environment variable to specify configuration dump file name** (page 44)

New POWER in the Command-Line Utilities

Whether you are a developer migrating an application to FairCom RTG or an administrator maintaining a system, FairCom has placed a powerful set of utilities at your fingertips. These utilities have a single focus: helping you to get the job done.

With Version 2, enhancements in these utilities make them even more powerful. Improvements appear in a variety of utilities, including those shown in the lists below.

ctutil:

- **Filecopy and configuration options** (page 39)
- **Option to dump configuration in XML format** (page 56)
- **-check option (-x) to scan a file for integrity issues** (page 56)
- **-upgrade switch** (page 59)
- **-clone to create empty copy of existing file** (page 56)
- **-test command to check configuration and connections to servers** (page 58)

ctstat:

- **ctstat reports sub-function timings** (page 63)
- **Show sub-function in last function** (page 64)

xddgen:

- **Support for names larger than 31 chars** (page 65)
- **Specify configuration files directory** (page 66)
- **Suppress dash or replace with underscore** (page 65)
- **Map OCCURS DEPENDING ON into longvarchar** (page 66)
- **Parsing improvements** (page 66)
- **Syntax for WITH DUPLICATES on RECORD KEY** (page 67)
- **Error/warning messages improved to be clearer** (page 67)
- **More xddgen enhancements...** (page 67)

File Copy Between Servers Simplifies Administration

Some legacy systems maintain large numbers of data files which must be accessed for reporting or auditing. In many cases, these files were scattered across multiple systems, further complicating the process of consolidating their data, especially when different operating systems were involved. System administrators may be accustomed to running a long list of command-line procedures to copy these file to a central location.

FairCom RTG allows you to automate file coping between servers to eliminate lengthy and tedious administrative procedures.

The FairCom RTG file copy API supports physically copying data and index files when the source and destination environments match. In the case of heterogeneous environments, such as different operating systems or endianness, file copy recreates the files record-by-record so operating system and endianness issues are dealt with for you. Administration is greatly simplified when large numbers of files must be moved for backups or reporting.

Improved Tutorials Lend a Helping Hand

One reason FairCom's database technology stands out above commodity databases is that FairCom was founded by developers, it is run by developers, and it caters to developers. We understand the importance of training people to get the most out of this technology. Even our "ready-to-go" FairCom RTG product will be easier to implement with proper training. Toward that goal, we have improved the FairCom RTG tutorials to make the process of moving to this powerful new environment easier-to-understand and implement.

Sqlize Tutorial

FairCom RTG provides many new opportunities for your COBOL applications. One of the most exciting is opening your data to SQL access. To take advantage of direct SQL access, you perform a process called "sqlizing," which prepares your data for full read/write SQL access without affecting access from your existing COBOL applications. FairCom RTG avoids the tedious task of normalizing existing tables.

A new sqlize tutorial guides you through the process of making your data ready for SQL access. The step-by-step instructions provided in the *Tips for Advanced Sqlizing* chapter of the **FairCom RTG User's Guide** (https://docs.faircom.com/doc**/ctcobol/**) explain the entire process. Depending on the structure of your data, you may need to use some of those techniques to sqlize your files.

The xddgen utility is used to map your COBOL data to SQL. This utility extracts file definitions directly from your source and creates a schema required for SQL access. Using information provided by xddgen, FairCom RTG directly maps COBOL tables to SQL. Your COBOL application continues to access your same data with no changes required.

The xddgen utility may be able to sqlize your data without any modifications to your copybook. In certain situations, your data will be structured in such a way that will require you to add compiler directives to your copybook to tell xddgen explicitly how to handle your data types. You will not need to restructure your data or rewrite your application. Simply add COBOL compiler directives to the source file as comments, which avoids all compilation issues.

Look for the sqlize tutorial in the *\Driver\ctree.cobol\tutorials* path of your FairCom RTG installation.

See also:

- *Tips for Advanced Sqlizing* in the **FairCom RTG User's Guide** (https://docs.faircom.com/doc**/ctcobol/**)

1.4 Server Engine Serves You Better

FairCom RTG Version 2 features the latest updates to the core engine. These advanced features debuted in the current release of c-treeACE, V11. These powerful features are now available to FairCom RTG users.

A recap of these features demonstrates why FairCom RTG Version 2 has the most stability, power, and performance of any release of this ground-breaking product.

Great Performance News for OLTP Applications

Performance remains a significant goal of every database system. FairCom continues a tradition of advanced performance solutions balanced with solid data integrity. Our two most powerful OLTP operating models now have additional performance boosting controls:

- A new transaction log mode for Delayed Durability
- New Background Flush controls for safety of non-transaction updates

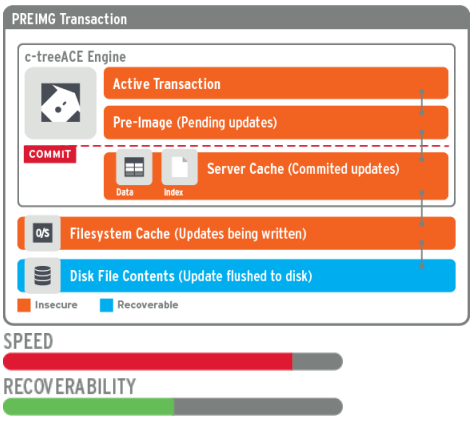
Delayed Transaction Durability

With full transaction control for complete ACID compliance, transaction logs are synced to disk with each commit operation, ensuring absolute data integrity with complete recoverability. Full, durable ACID transaction control enables many powerful features not available without recoverable log data:

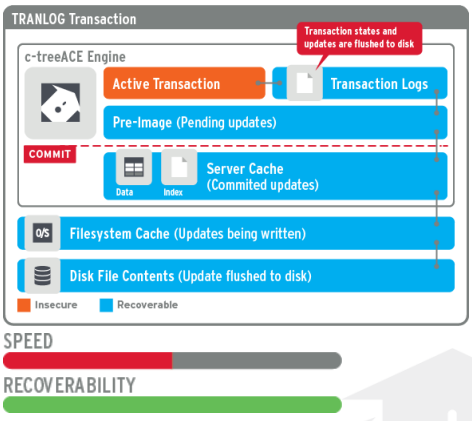
- Automatic database recovery
- Live database backups without rebuild on restore
- Replication
- Transaction auditing

The most critical of these is automatic recovery in case of system failure. Additionally, full transaction control remains a critical area of database performance tuning. Database updates must be secured in write-ahead logs for guaranteed recoverability. This comes with a performance impact due to the synchronous I/O requirements ensuring data is safely persisted.

Preimage (Atomicity Only)



Full Recoverability

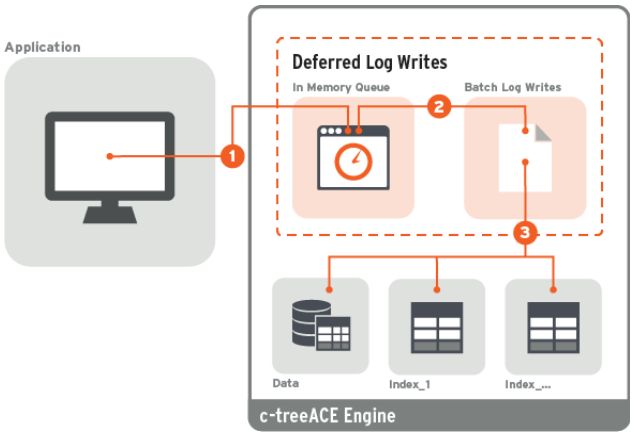


Many applications could benefit from a “relaxed” mode of transaction log writes. With today’s hardware stabilities and power redundancies, it is conceivable to slightly relax full durability constraints and still maintain acceptable risk tolerance. The balance becomes how much loss of recoverability these systems can tolerate.

Allowing database administrators to balance their window of vulnerability against on-line performance, c-treeACE provides a new Delayed Durability (page 70) feature for transaction logs. Regardless of your durability setting, you will always recover to a known state in time, with full referential integrity. The balancing decision becomes how current must you recover? If you are willing to accept guaranteed recovery, say up to one second ago, you can gain a significant amount of performance.

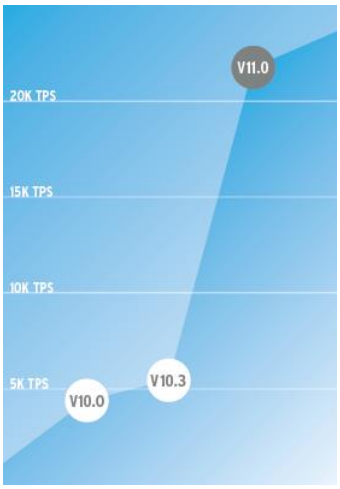
This feature is enabled with the following configuration entry and takes as an argument the maximum number seconds, *N*, that will elapse between log syncs. This ensures that, after a commit, the transaction log will be synced *in no more* than *N* seconds, thereby allowing you to define your window of vulnerability.

```
DELAYED_DURABILITY <N>
```



The end result can approach non-transaction performance while ensuring committed transactions are persisted to storage within less than *N* seconds of vulnerability. In selected test cases, **up to 300% faster transaction throughput** has been observed when configured with 1 second of delayed durability.

Windows Performance



Legend:
V10 - c-treeACE Database Server, Full Transaction Processing, 100 users
V10.3/V1 - c-treeACE / FairCom RTG Database Server, Full Transaction Processing, 100 users
V11.0/V2 - c-treeACE / FairCom RTG Database Server, Full Transaction Processing, 100 users, DELAYED_DURABILITY1

Test Environment:
Dell PowerEdge R710 machine with 2 Xeon 3.6 GHZ Xeon Processors with a total of 32 logical cores, 32 GB RAM, 600G 15000 RPM Drive, Windows Server 2012.

Test Utility:
FairCom's cttctx load test utility simulating a record add / read / delete sequence on 23 files with 4 indexes per file. 100 threads x 10,000 iterations = 1,000,000 transactions per file for a total of 69,000,000 transactions.

Background Flush of PreImage/Non-transaction Data Updates

c-treeACE offers multiple levels of transaction protection for your data. Some applications do not require the recoverability full transaction support provides for performance reasons. However, these applications become quite vulnerable to data loss should system failure occur. If FairCom Server terminates abnormally, say from a power failure, updates to data and index files that are not under full transaction control are lost if those updates have not yet been written from c-tree's in-memory data and index caches to the file system. The following factors typically reduce the number of dirty pages that exist:

1. When an updated cache page is being reused, the updated page is written to the file system cache.
2. When all connections close a c-tree file, FairCom Server writes the updated pages to the file system cache before closing the file.
3. An internal thread periodically checks if FairCom Server is idle, and if so it writes updated pages to the file system cache.

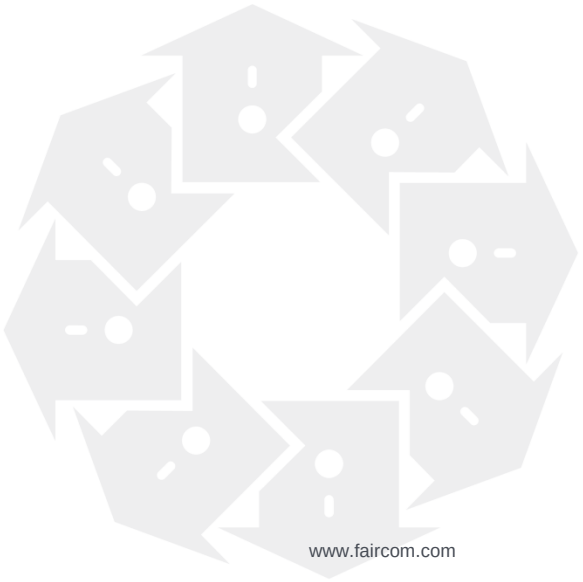
However, the combination of using very large data and index caches, keeping files open for extended periods of time, and constant OLTP activity increase your likelihood that more dirty cache pages exist.

This data vulnerability is now greatly reduced in such a way that allows YOU to define your window of vulnerability. Database administrators can balance performance needs with data safety guarantees to within seconds—or they can select immediate safety.

To set a limit on potential loss of updates for non-transaction data and index files, FairCom Server now supports background flush (page 76) options to write dirty pages to the file system within a specified time period, limiting potential data loss for non-transaction data and index files.

Performance tests have shown that with only a few seconds of data vulnerability, you can obtain the same great performance you are already accustomed to, but now with an assurance that much more of your data is secured to disk thereby substantially reducing your data vulnerability.

See **Delayed Durability Transaction Log Mode for Performance** (page 70).



Coordinate Application Recovery with Transaction Restore Points

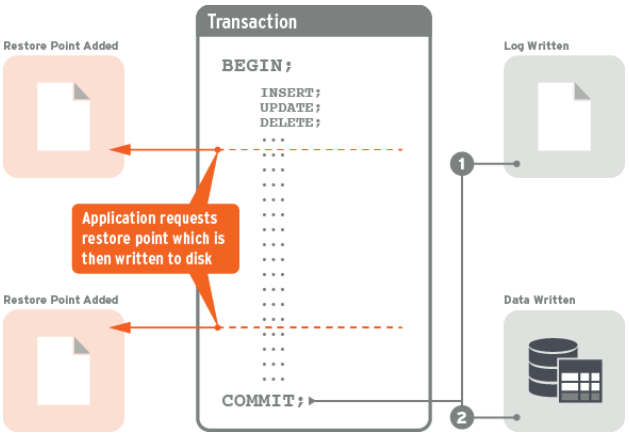
Transaction control by definition requires logging all database changes to persisted storage. Of course, transaction control is highly desirable for recovery of the database in the event of system failure. It also makes available other valuable features such as replication. However, this comes with a direct and measurable performance cost.

The new Delayed Durability feature can greatly enhance performance; however, should a system fail, database recovery could take considerable time if the transaction logs held substantial volumes of data not yet flushed to data and index files.

This leads to an additional challenge of how to balance data integrity using transactions against performance **and** recovery. Certain classes of applications can restore to known points in time. Consider the simple case of a bulk load. Should anything fail during loading, the load process can usually be restarted. If this known position could be coordinated with a c-treeACE server transaction log position, one could architect a very effective recovery process.

To this end, a new Restore Point feature has been introduced. A Restore Point is a position in a transaction log to which FairCom Server's automatic recovery can roll back the system in event of system failure.

The Restore Point is used for restoring a system to a point in time coordinated with the application as a known good state. After restoring the system to a Restore Point, the application can be architected to perform work that was not recovered. For example, those applications maintaining their own journal of operations. By coordinating a known good state in time between the application and the server's transaction logs, these applications can quickly recover back to a point in time when absolutely needed.



See **Transaction Restore Points for Application Recovery** (page 88).

More V11 Features in FairCom RTG V2

Millisecond Timestamp Resolution

We now introduce millisecond timestamp resolution for c-tree time types. This support is available across multiple c-tree APIs supported by FairCom RTG. See **Millisecond Timestamp Resolution Support** (page 82).

Table Locks

Table Locks prevent other connections from reading and/or writing to the file. These can greatly reduce resource usage and contention with other connected users. See **Table Lock Support** (page 84, **/doc/ctreeplus/table-lock.htm**).

TCP IPv6 Support

Internet Protocol IPv6 format is available for FairCom RTG servers on Windows and Linux (Java and ADO.NET SQL clients currently support only IPv4). IPv6 greatly expands the number of available IP addresses over the previous standard. See **IPv6 Support** (page 87).

Memory Files

True in-memory files for extremely fast data access performance.

Quiesce and File Block

Block physical access to files while not stopping your server process.

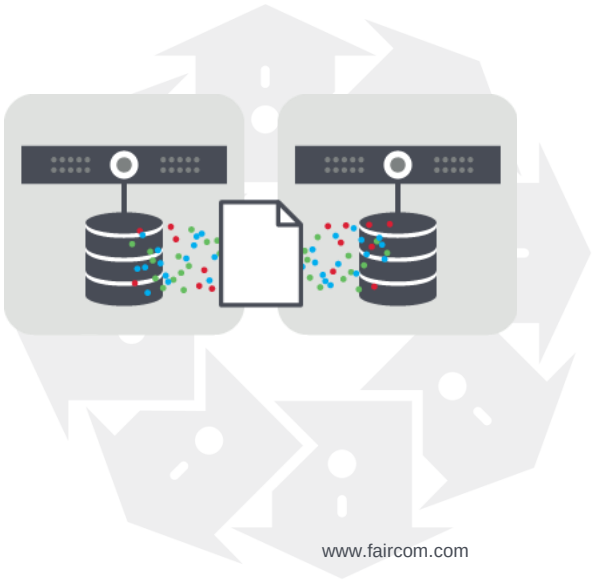
- **Quiesce** is designed for quick SAN snaps and other times when a complete quiet server state is required. Multiple options are available for full or crash consistent data states.
- **File Block** is the perfect solution when you simply need to replace or move a file on an active server. Block the file, replace and unblock. c-treeACE coordinates all client interactions.

Hot Backups

Our Dynamic Dump backup is a real-time hot backup solution. Completely transparent and automatic backups can take place while users stay connected and working. Full featured restore includes roll forward for the ultimate in data safety.

Copy Files Between Servers

Easily moving and copying files between servers is a consistent need for our customers. A new function makes this a seamless effort directly from the application layer. File transfer is now done server-to-server in addition to the client-server transfer feature introduced earlier. See **FairCom RTG - ctutil filecopy and configuration options** (page 39).



NODE_DELAY default value changed from 75 to 0

When many keys are deleted from an index, causing many nodes to become empty, searching for a key value can require walking over many of these empty nodes. This can slow performance. In particular, we noticed that the space management index of a variable-length data file experienced this issue. A simple case that demonstrates this issue is to add records to a variable-length data file, then delete all or many of the records, and add more records. The second set of adds takes longer than the first due to the presence of the empty nodes in the index tree.

The default `NODE_DELAY` settings for FairCom RTG have been changed from 75 to 0. A `NODE_DELAY` value of 0 causes the delete node thread to wait 1 second before pruning a node from the index tree.

Expect Faster Application Throughput from Automatic Transaction Optimization

Transaction control is one of the most important data integrity benefits FairCom RTG brings to existing COBOL applications. Adding transaction support with FairCom RTG is transparent to the application with easy local configurations. Data integrity is provided by security data to persisted storage as soon as it is committed to the database. That persistence introduces a slight performance hit. Due to the nature of the automatic transaction support, this performance hit can be significant in various use cases. FairCom RTG V3 has improved this area of automatic transaction performance.

FairCom RTG creates tables as transaction processing capable by default; however, FairCom RTG applications seldom complete taking advantage of transaction processing control. COBOL, by default, opens tables in such a way that explicit transactions do not apply to them. In these cases, FairCom RTG now includes logic to detect and force automatic transaction processing.

When a FairCom RTG client sends a request for an update to the server, the following happens:

1. The server begins the transaction.
2. The server performs the update.
3. The server commits the transaction.
4. The server replies to client with the result.
5. The server waits for the next client request.

To enable this improved transaction optimization, include the following in your local FairCom RTG configuration options (*ctree.conf*). This is disabled by default.

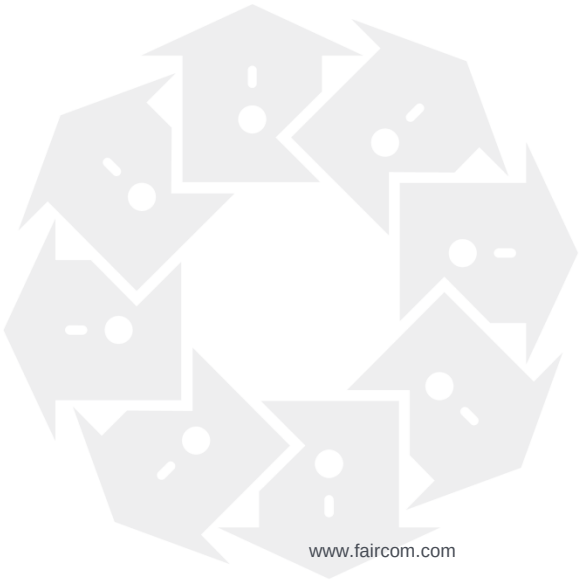
This optimization reduces the amount of time before the server answers back to the client after an update. The server replies to the client while it is committing the transaction (the commit does not need to complete before returning back to the client). Compare the following sequence of events when enabled:

1. The server begins the transaction.
2. The server performs the update.
3. **The server replies to client with the result**
4. **The server commits the transaction.**
5. The server waits for the next client request.

By reducing the amount of time *before* returning the answer to the client, this greatly improves application throughput.

This has the effect that the record is not exactly committed or aborted by the time the client operation instigating the automatic transaction to the server (add, delete, update) returns. In case of disaster recovery, this may cause the last record update to be lost if the "disaster" occurred after the answer has been received by the client and before the commit is executed by the server, an expectedly very rare event.

This optimization is similar to the `DELAYED_DURABILITY` configuration (set on the Server side in *ctsrvr.cfg*) as both guarantee transaction atomicity and consistency and not durability, meaning the last transaction may be lost in rare cases of system failure. Typical envisioned failure scenarios include out of storage space, out of memory conditions, and storage system integrity issues. In case of transaction failure, the FairCom RTG server logs a *CTSTATUS.FCS* message and shuts down in a controlled coordinated effort.



Latest c-treeACE SQL Features

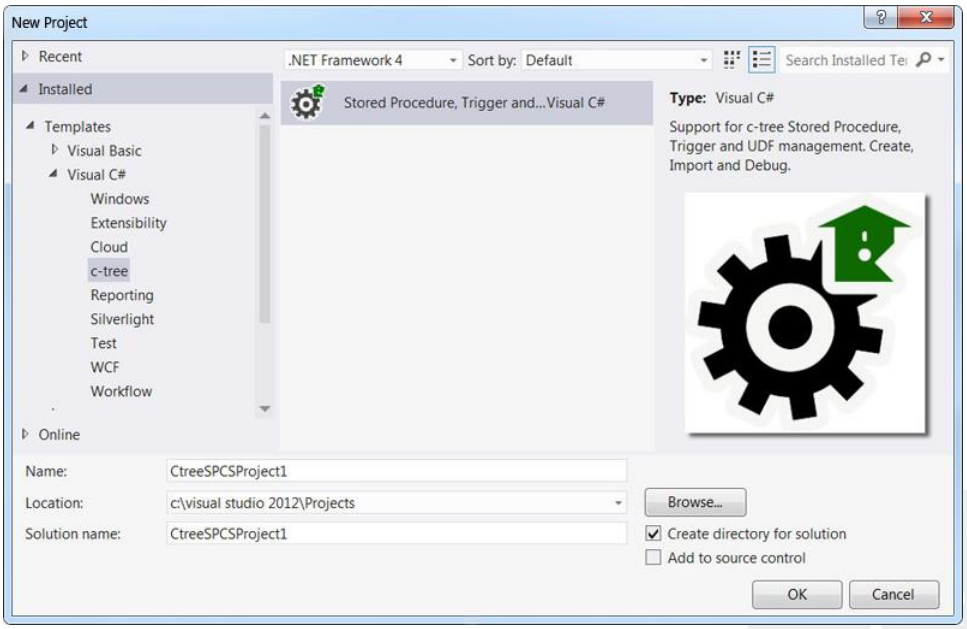
c-treeACE SQL brings standards-compliant SQL to even your most non-SQL data. You've invested years in your successful c-tree applications. Now you can enjoy fresh new interfaces and capabilities while maintaining strong and proven c-tree foundations. Many applications are already reaping the benefits. And with that, many requests for extended SQL capabilities have been suggested. c-treeACE SQL V11 brings a list of great new features.

Stored Procedure Development in the .NET Framework - C#

SQL stored procedures and triggers are an easy and powerful way to move advanced logic into your server core, protecting investments in performance-sensitive processing while maintaining a centrally managed approach for long-term maintenance. Stored procedures fully encapsulate business logic for business rule enforcement. Triggers maintain database integrity directly at the database server level.

c-treeACE SQL has long supported stored procedures, triggers, and user defined functions with Java for cross-platform development ease.

Stored procedures are now available for your Microsoft Windows .NET development environment. Program your procedure in C# or any other .NET supported language. Seamlessly remain in your .NET environment while developing advanced server-side processing logic. With Visual Studio integration, it is easy to create, debug, and deploy your C# stored procedure code remotely. Your assemblies are deployed alongside c-treeACE SQL with a click of a mouse.



See **New Stored Procedure Development Frameworks** (page 81).

NetBeans Plugin for Java Stored Procedure Development

c-treeACE SQL now includes a Java NetBeans plugin for advanced development potential. Take advantage of the complete
NetBeans IDE in creating and maintaining new and existing Java stored procedures. Existing Java stored procedure investments can continue to be used with much easier maintenance and deployment capabilities. See

NetBeans Plugin for Java.

More SQL Enhancements

c-treeACE SQL includes a list of many other big new additions for easier and enhanced SQL capabilities with your existing applications:

- **Entity Framework 6** Support with ADO.NET
- **SQL Groups** for much easier user role management
- **Table Valued Functions** for dynamic calculated tables in queries based on stored procedure logic
- **Extended ALTER VIEW, ALTER TABLE, and ALTER INDEX Flexibility** for views, tables, and index management
- **Table Locks** for exclusive user access
- **Recursive Query** support for hierarchal “tree”-based queries
- **Scrollable Cursors**
- And continued query optimizer improvements for ever faster performance

Collectively, these great new SQL features bring enhanced administrative capabilities to new **and** existing applications.

2. FairCom RTG V2 Quick Guide to Upgrade Procedures

Your FairCom RTG system can be upgraded to the latest version with very few changes. It is important to follow the procedures in this section to upgrade in a safe manner.

Upgrade to the latest FairCom RTG V2 Database Engine

While FairCom attempts to maintain backward compatibility when at all possible, transaction logs from earlier versions are not always compatible with newer FairCom RTG formats. For example, FairCom RTG V2 is built upon the latest FairCom Server (V11), which introduced changes in the transaction log to accommodate new capabilities. When backward compatibility is affected, then all of the steps in the following upgrade procedure are required. In cases where no backward compatibility exists, then steps 3 through 6 can be considered optional.

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

It is easy to install and use FairCom RTG with your existing files by removing prior transaction logs in a safe manner. Follow these easy steps, which are appropriate any time you are upgrading a c-tree installation:

1. Have all clients cleanly exit from the existing the c-tree Server.
2. Perform a controlled shutdown of the c-tree Server with your normal shutdown sequence. Verify the c-tree Server shutdown was successful by looking for the following "Server shutdown completed" in *CTSTATUS.FCS*:

   ```
   - User# 00023 Server shutdown completed
   ```

3. Block the ability of any clients to attach to the c-tree Server.
4. Restart your existing c-tree 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 your existing c-tree Server.
6. Remove existing transaction logs and associated files (*L*.FCS*, *S*.FCS*, *D*.FCS*, and *I*.FCS*).
7. Copy your new FairCom Server executable (**ctsrvr.exe** or **ctreesql.exe**) into the existing c-tree Server directory.

> **Note**: Client compatibility can prevent connections to the new FairCom RTG database engine. It is always advised to use the most recent matching client version with your FairCom RTG server version. Version 11 of the FairCom Server introduced backward compatibility changes that affect FairCom RTG.

8. Unblock the ability of any clients to attach.
9. Start c-treeACE in your usual manner and begin accessing your existing data.
10. **Optional - for SQL users**: Sqlize your tables using the **sqlize** utility or **ctutil** *-link*. The V2/V11 version of c-treeACE SQL Explorer provides new features, such as highlighting sqlized tables in different colors, checking for bad records, and adding SQL indexes (see *c-treeACE SQL Explorer* (page 51)). If you want to use these features, you will need to sqlize after upgrading to FairCom RTG V2 **even if you have sqlized in an earlier version of FairCom RTG**.

For more information, see the *Knowledgebase* (https://docs.faircom.com/doc/*knowledgebase/ace_upgradeace.htm*) in the online documentation.

FairCom RTG Component Upgrade

You may need to recompile existing runtimes. Although recompiling it is not strictly necessary, it is recommended. See the setup section for your compiler in the *FairCom RTG User's Guide* (https://docs.faircom.com/doc/*ctcobol/*) for details.

SQL Driver Upgrades

FairCom RTG includes a collection of SQL drivers to allow relational access through a variety of interfaces:

- JDBC drivers
- ODBC drivers
- PHP modules
- ADO.NET drivers
- Stored Procedure frameworks

To upgrade the drivers, simply run the FairCom RTG installation program again. You do not need to uninstall the existing drivers before upgrading.

3. FairCom RTG V2 Update Details

FairCom has implemented many enhancements and changes in the latest version of FairCom RTG. These updates make this the best release of FairCom RTG ever.

This document lists enhancements and changes you will experience when you upgrade to FairCom RTG V2.

3.1 New SQL Enhancements

SQL Indexes on COBOL Files

FairCom RTG allows you to create a SQL index on your FairCom product tables. You can execute CREATE INDEX on imported tables or you can use the graphical tools, c-treeACE SQL Explorer and c-treeACE Explorer.

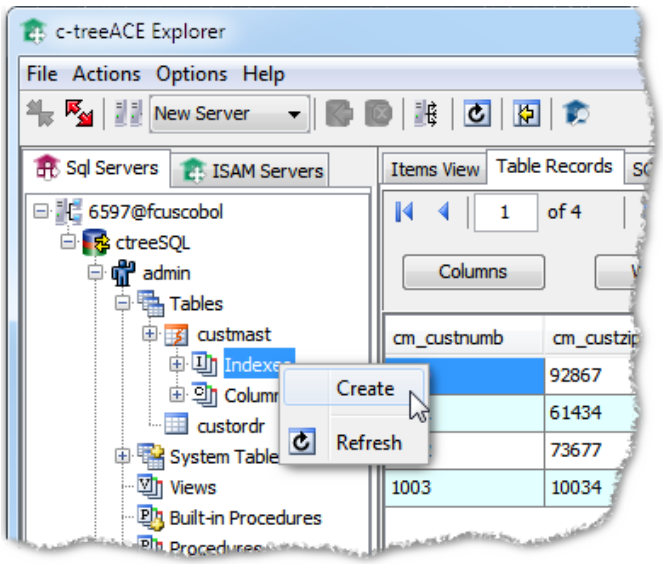
Performance of SQL Queries

A practical use of this feature is in handling FairCom product indexes that do not translate well to SQL. In some cases, the index is imported into SQL by sqlize with limited functionality, so it can be used only to perform searches using the "=" and the "<>" (not equal) operators. In a small number of cases, the index cannot be sqlized at all.

These cases may impact the performance of SQL queries on those fields. This issue is now easily addressed: simply create SQL indexes where needed to speed up queries. It is not necessary to replace every FairCom product index, simply replace those that are needed to speed up your SQL queries.

Note: A SQL index cannot be created on FairCom product "bit" data types (Boolean).
Date/time types (**date**, **time**, **datetime** / **timestamp**) can have additional indexes created if their type definitions are in collation order (YYYYMMDD for dates).

The presence of a SQL index will better optimize some queries. As a side effect, the index returns rows sorted in logical order as opposed to binary order as is the case for FairCom product indexes.



SQL create index logic on date/time fields

FairCom RTG V2 introduced support for creating indexes in SQL over data that sorts properly for SQL. That support did not include fields using a numeric field type that was mapped to date, time, and timestamp SQL fields.

This support includes indexing SQL fields mapped to data with two constraints:

1. Indexing of the underlying actual data type must be supported.
2. The numeric value of the type and its date, time, or timestamp interpretation must sort identically (i.e., dates having a YYYYMMDD format can be sorted but not dates with a MMDDYYYY format).

Create and Alter Table Support for COBOL Files

The release provides full DML (ALTER TABLE, CREATE INDEX) on tables created in COBOL (containing XDD information) and imported into SQL per the same index creation restrictions as noted above.

Right-justified strings (JustAN field type) mapped to SQL

The COBOL-to-SQL type mapping did not support JustAN (right-justified strings) COBOL types. JustAN types are now supported by mapping them into VARCHAR field types.

This mapping best mimics the behavior of these fields in COBOL. In particular, varchar is the only SQL type that considers trailing spaces as significant.

Due to the nature of JustAN types, left-side spaces are considered padding. On reading from the COBOL buffer, they are trimmed so they do not appear in SQL, where they would be significant data.

Variable-length fields mapped into LONGVAR* SQL field

Variable length fields map into SQL LONGVAR* fields.

The XDD structure allows the following elements and attributes for variable length field support.

1. dbtype values:
   - BLOB: Indicates a variable-length binary object with length depending on a field value.
   - CLOB: Indicates a variable-length text object with length depending on a field value.
2. <field> attribute *sizefield* used in conjunction with dbtype BLOB or CLOB and having size = "0"

For an XDD field mapped into a LONG VARCHAR or LONG VARBINARY, the following conditions must be met:

1. The field definition must have:

   `size="0"`

   `sizefield="X"` where "X" is a valid field containing the number of bytes (we suggest this field to be hidden, but this is not mandatory)

   `dbtype="clob"` or `dbtype="blob"`

1. At maximum, one and only one field mapped to a BLOB or CLOB type.
2. *It must be the last field in the record buffer*.

If one (or more) of the above condition is not met, error **CTDBRET_CALLBACK_11** ("Unsupported clob/blob definition") is returned.

3.2 FairCom RTG V2: More Power to You

File copy between heterogeneous servers

The FairCom RTG file copy API now supports copying between two heterogeneous servers. See also*FairCom RTG - ctutil filecopy and configuration options* (page 39).

Relaxed index re-organization errors on redefined schema affecting index definition (WHENFULL)

Enhancements have been made to the logic to better handle the situation when the "`*>XDD WHENFULL`" directive has been used in **xddgen** generating the XDD and there is no valid combination of fields to compose a key. Now the index will be ignored by SQL instead of returning an error (**CTDBRET_CALLBACK_17**). In this situation, SQL will not be able to use the index to optimize queries. The index information, however, will be properly maintained on updates/inserts performed through SQL.

Conform file locking behavior to ACUCOBOL's V_INTERNAL_LOCKS when <runitlockdetect> set to 'no'

When the ACUCOBOL configuration setting `V_INTERNAL_LOCKS` is set to "0" (off, false, no), Vision tracks locks but does not enforce internal record or file locking. The FairCom RTG configuration setting `<runitlockdetect>`, which aims to simulate the `V_INTERNAL_LOCKS`, did not enforce record locks but still enforced file locks. This modification corrects the file locking behavior so that it is now possible, for example, to exclusively open a file twice from the same run unit.

Improved isCOBOL compatibility for UNLOCK operation

FairCom RTG has been modified to address an unexpected behavior when used with isCOBOL:

In an isCOBOL application the UNLOCK operation releases locks when it is called within an active transaction, while the expected behavior would be to not release any lock.

The logic has been modified to conform to this behavior.

Improved log output with file names on ERROR entries

The output of the FairCom RTG log includes a file name with the log messages. Warning and error log messages are formatted with the file name appended at the end of the message. The file name is preceded by the instance name, when specified in *ctree.conf* with the `<instance name>` attribute, or by the instance number and file number to uniquely identify the file as follows:

```
instance_number#file_number:file_name.
```

The file name is logged for START, READ, NEXT, PREVIOUS, WRITE, REWRITE, DELETE, and UNLOCK operations.

In addition, every entry in the log is pre-pended with a thread ID to uniquely identify the process/thread that made the log entry. The thread ID is made of 8 hexadecimal digits followed by a “>” sign.

Example:

```
00002D40> 20140915T141358 core:3494:cts_rewdel ERROR 5:42:0 record is locked 0#0:arc1.dat
```

Substitution Specifiers

The following substitution specifiers can be used in the name attribute of the `<instance>` and its variant elements:

- `%t` to print the thread ID in unsigned decimal format
- `%p` to print the process ID in unsigned decimal format
- `%i` to print the instance number in unsigned decimal format

These specifiers can optionally contain embedded format specifiers with the following prototype:

```
%[0][width][x|X]specifier
```

- `[0]` left-pads the number with zeroes (0) instead of spaces when width is specified
- `[width]` minimum number of characters to be printed. If the value to be printed is shorter than this number, the result is padded with blank spaces. The value is not truncated even if the result is larger.
- `[x]` (lower case) number is as unsigned hexadecimal integer
- `[X]` (upper case) number is as unsigned hexadecimal integer

Generic debug log messages

This modification to FairCom RTG introduces a new `<log><debug>` option `<generic>` to disable generic debug log messages which are enabled by default when `<log><debug>` is not specifically set to `no`.

FairCom RTG introduces support for ExtFH File Information operation

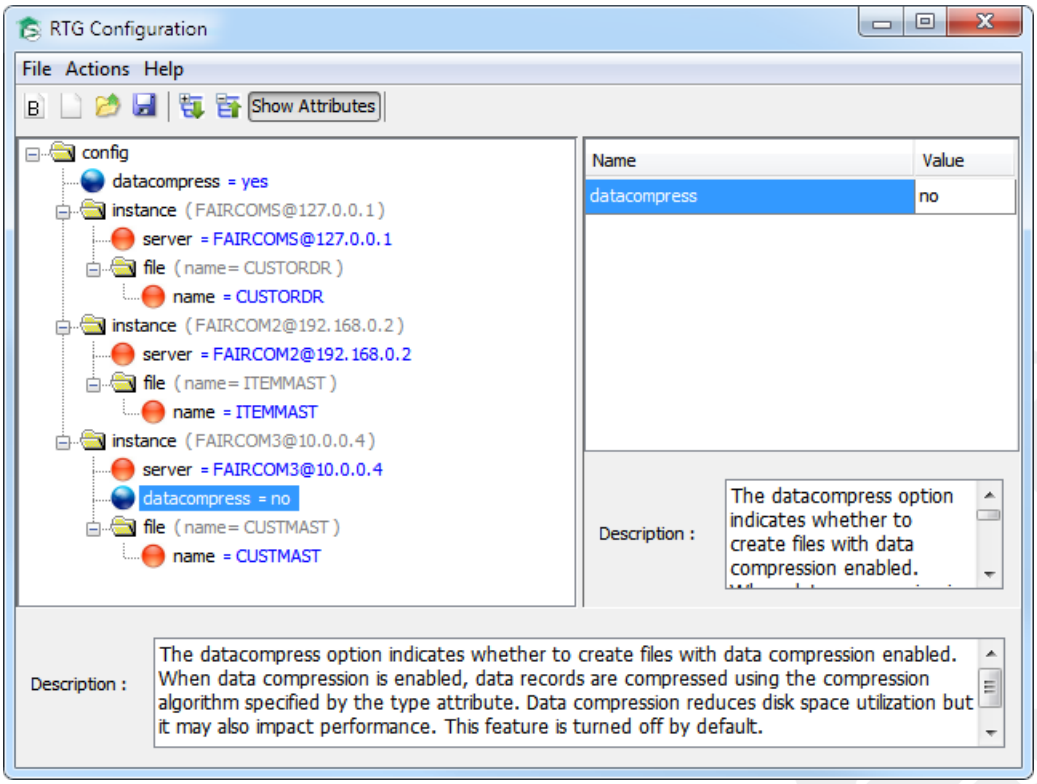
Another advance has been made in the integration between FairCom RTG COBOL and your application. FairCom RTG now offers support for the ExtFH File Information operation. It is now possible to call **CTEXTFH()** with opcode 06 to retrieve key definitions and other file information.

4. FairCom RTG Configuration

The FairCom RTG Configuration Tool, RTG Config, is new and improved in this release. This section highlights the changes and the options provided for configuring your system.

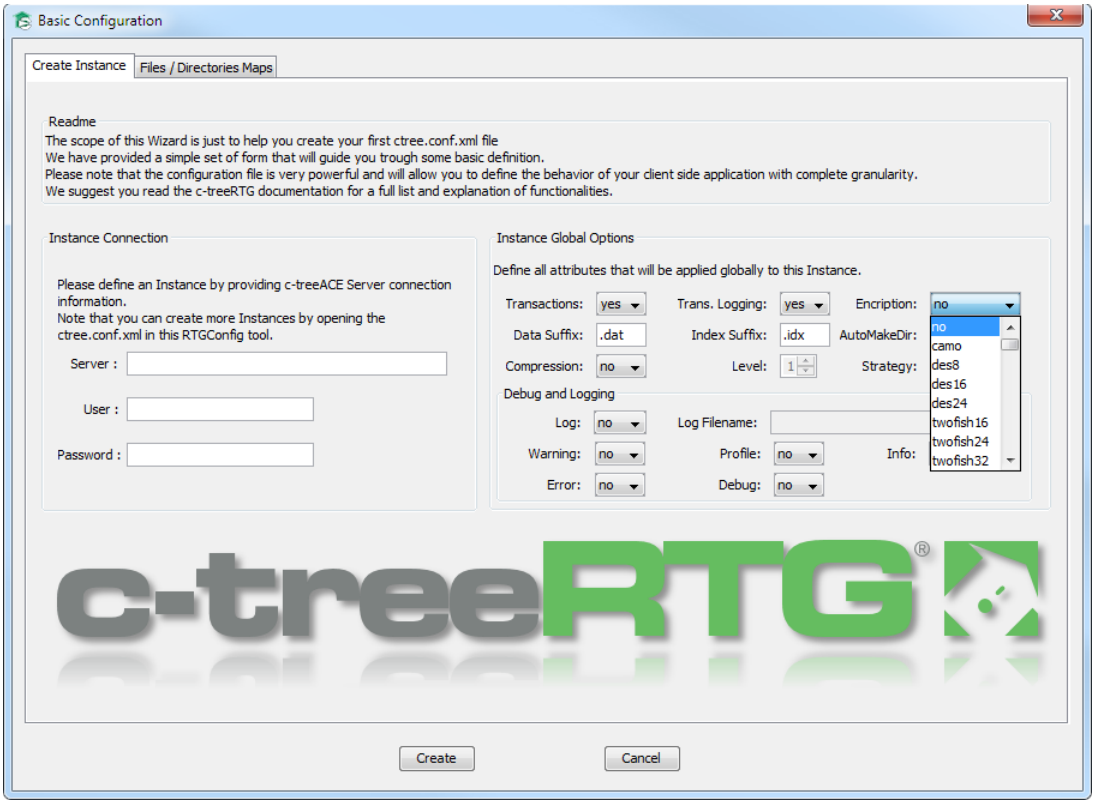
4.1 Easily Achieve the Perfect Setup with the New Configuration Tool

The FairCom RTG Client Configuration Tool is your way to establish the configuration necessary for the client to connect to FairCom RTG. The newly updated RTG Config tool simplifies the process of defining connection strings and file locations.



4.2 New Basic Configuration dialog

A Basic Configuration wizard has been added to the RTG Config tool. This window allows a simple setup of a basic configuration.



4.3 Configuration Tool now recognizes "priority" and "casesensitivity" attributes

The FairCom RTG RTG Config Configuration Tool has been updated to recognize the `priority` and `casesensitivity` attributes on a <file> tag.

4.4 Added configuration elements to the Configuration Tool

The Configuration Tool for the FairCom RTG products has been updated to include all the new configuration elements.

Logic was added to signal and automatically discard unknown elements and attributes so that the Configuration Tool does not fail opening configuration files having unknown information.

4.5 File organization specification in configuration file

It is now possible to specify the file organization using the "type" attribute in the `<file>` tag of the FairCom RTG client configuration.

Valid values of the "type" attribute are:

- *I* - Indexed file
- *R* - Relative file.
- *L* - Line sequential file
- *S* - Record sequential file
- * - Any file type (this is the default if nothing is specified)

At this time FairCom RTG only supports indexed files, so any file organization different than "*I*" and "*" should be used only with `redirinstance` pointing to a different file system.

The file organization is taken into account when matching the file rules.

Matching All File Rules

Particular attention is required for the "matching all file rules" (a rule matching all files in all directories): Now FairCom RTG needs a "matching all file rule" for *each* file type.

When a "matching all file rule" does not specify the file type, it will be used if there is no specific "matching all file rule" for a particular file type.

If a file rule does not specify the file type, the missing "type" is interpreted as matching all file types of files regardless of whether a name and/or dir is specified.

4.6 <config filematch> attribute to specify file matching precedence algorithm

A new attribute has been added to the `<config>` element: `<config filematch>`. The value of `<config filematch>` determines which `<file>` rule takes precedence when more than one rule applies to a file.

- Setting `<config filematch="0">` results in the FairCom RTG configuration using the original algorithm to provide backward compatibility for existing FairCom RTG configuration files.
- Setting `<config filematch="1">` results in the FairCom RTG configuration using the new and simplified file matching algorithm.

The differences between the two algorithms are listed as follows:

- In case of multiple wildcard matches: The rule with the most matching characters (name and dir combined) takes precedence. For example, file name "custmast" matches `<file name="cust*">` before matching `<file name="*ast">`.

- In case the combined name and dir lengths are equal: The order of appearance in the configuration file is used. For example, file name "custmast" matches whichever comes first in *ctree.conf* between `<file name="cust*" dir=".">` and `<file name="*mast" dir=".">`.

4.7 <forcedelete> Configuration Option to Force Deletion of Orphan Files

The COBOL operation DELETE FILE fails with COBOL error **98** (corrupted file) on orphan files that are data files that are missing their relative index file. The DELETE FILE operation on Micro Focus instead, deletes the orphan files and returns successful.

The new configuration option `<forcedelete>` forces DELETE FILE operations to delete any orphan file left when set to "yes". When it is set to "no", it causes DELETE FILE to return an error and leave the orphan file on disk.

By default, `<forcedelete>` is set to "no" except for ExtFH drivers where it is set to "yes" to match Micro Focus COBOL behavior.

Before the introduction of <forcedelete>, the default behavior for FairCom RTG ExtFh driver was to fail DELETE FILE on orphan data files (missing its index file) and succeed on orphan index files (missing its data file) to match the behavior of old Micro focus COBOL products. More recent Micro Focus product behavior is to remove any orphan file and return success.

4.8 <locktimeout> configuration option to set blocking lock timeout

In V2 and later, a new configuration option, `<locktimeout>`, sets the blocking lock timeout. For example: `<locktimeout>15</locktimeout>` sets the timeout for blocking locks to 15 seconds.

- If `<locktimeout>` is set to a value greater than 0, an operation attempting to acquire a blocking lock on a locked record returns after `<locktimeout>` seconds with a locked error.
- If `<locktimeout>` is set to 0 (default), an operation attempting to acquire a blocking lock on a locked record waits until the record becomes available.

4.9 <filecopy>

The `<filecopy>` configuration setting forces the use of the server-side file copy capability to carry on copy operations. The `<filecopy>` option is not enabled by default so a user must explicitly enable it in the configuration file in order to use it. When `<filecopy>` is enabled and conditions do not allow it to be used, the copy operation fails rather than falling back to other copy methods.

The `<filecopy>` option forces a physical copy of the data and index files. The other copy mechanisms available extract the structure information for the source file, create a new file accordingly with the current *ctree.conf* settings, and then copy records from the source to the destination (the outcome of this process can be a file which is not identical to the source file, for example it might be compressed when the source was not).

Accepted Values

Value	Effect	Synonyms
yes	Forces the use of the server-side file copy (a physical copy of the data and index files).	y, true, on, 1
no	Extract the structure of the source file, create a new file (with current *ctree.conf* settings), and copy records from the source to the destination.	n, false, off, 0

Attributes

Attribute	Description	Synonyms
overwrite="y"	Allow existing files to be overwritten (disabled by default).	

See also:

- *ctutil -filecopy*

4.10 <forcedelete> - File delete correctly handles missing file with <forcedelete> enabled

A `DELETE FILE` COBOL operation failed with COBOL error **9D** if the file was missing and the `<forcedelete>` option was enabled. The logic has been corrected.

4.11 <log><error> types added: <locked>, <missingfile>, and <undefined>

This modification introduces <log><error> configuration keywords to mask three new error types in addition to the existing <atend>, <duplicate>, and <notfound>. This can be useful if you encounter a large number of error **9:0** (undefined record position) in your FairCom RTG logs and need a way to mask them. The following new types are now supported:

- <locked> to mask record locked errors
- <missingfile> to mask file not found errors
- <undefined> to mask no current record position errors

In the FairCom RTG log file, these errors can be identified by the first of the three colon-separated error values:

```
RTG Error : Underlying c-tree API Error : Operating System Error
```

For example, a record locked error will display the following (notice **42** is the c-tree error for record locked):

```
ERROR 5:42:0
```

The complete list of FairCom RTG errors that can be masked now contains the following:

```
Keyword        RTG Error

<atend>        25

<duplicate>     7

<locked>        5

<missingfile> 15

<notfound>      8

<undefined>     9
```

4.12 <log><debug><transaction> configuration to debug transactions

This modification introduces a new <log><debug> option to debug transaction calls in FairCom RTG. To enable this feature, add the following to *ctree.conf*:

```
<log><debug><transaction>yes</transaction></debug></log>
```

4.13 <log><debug><transaction> additions

This modification is an addition to the new <log><debug><transaction> configuration to add more locations where debug information about transactions is logged.

4.14 <log file> attribute now supports substitution specifiers

Substitution specifiers can be used in the <log file> configuration attribute. The <log file> attribute can contain substitution specifiers to build the log file name using (for example) an environmental variable.

Example:

The following setting creates a different log file for each Windows user:

```
<log file="%(USERNAME).log">yes</log>
```

4.15 <normalize> configuration option to normalize file paths

The <normalize> configuration option helps normalizing file paths passed to FairCom RTG. This option is global-only and can be set only as a child of <config>. It is not valid if specified as child of <*instance> or <file>. When this option is in use the file path gets normalized before any file configuration matching.

The <normalize> section can perform two types of normalization:

1. <sep> - Specifies the path separator, such as forward slash (/) or backslash (\).
2. <drive> - Specifies a drive letter to be used if the path does not include a drive letter.
3. If the file path to be normalized already includes a drive letter, <drive> accepts a force= attribute to force it to change the drive to the one specified, for example <drive force="y">.

Examples

The following `<normalize>` section replaces all path separators with a backslash (`\`) and prefixes all the file paths with a `C` drive unless the path to be normalized already includes a drive letter:

```
<normalize><sep>\</sep><drive>C</drive></normalize>
```

If the file path to be normalized already has a drive it is not changed by default. It is possible to force the drive to be changed to the one specified by setting the `force=` attribute to `yes`. For example:

```
<normalize><drive force="y">C</drive></normalize>
```

It is also possible to remove an existing drive by specifying an empty drive. For example:

```
<normalize><drive></drive></normalize>
```

4.16 <normalize> options to handle relative paths

This modification introduces two new options to the <normalize> configuration elements:

- The `<normalize><drive relative="yes"/><normalize>` configuration attribute specifies if the drive should be added to relative file paths in addition to full paths. By default the option is set to "no" which means that the drive is not added to relative file paths.
- The `<normalize><relative>yes</relative></normalize>` configuration element specifies if the relative file paths should be normalized to always begin with a '`./`' or '`.\`'. By default the option is set to "no" which means that no `./` or `.\` is added to relative file paths.

4.17 <sqlize> attributes substitution specifier support

The `<sqlize>` option in *ctree.conf* now accepts substitution specifier support for its attributes: `database`, `password`, `symbolic`, `prefix`, `owner`, and `rules`. For example:

```
<sqlize prefix="%(NOME_START)_"/>
```

This support is similar to the support for substitution specifiers provided by the XFD attribute, as described in the *Substitution Specifiers /doc/ctcobol/SubstitutionSpecifiers.htm* topic in the FairCom RTG user guide.

4.18 <temporary> configuration element to create files with reduced disk I/O

In V2 and later, a FairCom RTG configuration element, `<temporary>` can be used to suppress the disk flush of certain operations to increase performance.

Note: Use this option only on files that are not required to persist in the event of a system crash or power failure.

4.19 <trxholdslocks> config option sets transaction lock behavior

A new configuration keyword, `<trxholdslocks>`, has been introduced to simulate ACUCOBOL's `TRX_HOLDS_LOCKS=1` option, which does not free locks at transaction commit unless the locked records are updated during the transaction.

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

Accepted Values

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

Example

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

4.20 CTREE_CONF_DUMP environment variable to specify configuration dump file

The `CTREE_CONF_DUMP` environment variable allows the FairCom RTG configuration to be saved to a specified file in XML format. If the value of this environment variable is a file name, FairCom RTG dumps its configuration in XML format to the specified file after FairCom RTG has been initialized with a *ctree.conf* (or *iscobol.properties*) configuration.

IsCOBOL Tip: This feature can be used as a way to migrate the isCOBOL configuration options related to FairCom RTG from *iscobol.properties* format to *ctree.conf* format.

5. FairCom RTG Data Migration

This release of FairCom RTG includes a more robust set of tools to help you get your system up and running. The RTG Migrate tool and the graphical sqlize tool get your data ready for your FairCom RTG application.

5.1 RTG Migrate Improvements

The RTG Migrate tool has undergone improvements to its user interface. The interface has been rearranged to remove unused fields. Improvements have been made to the descriptions and selections in the combobox that allows selecting the type of COBOL environment.

As a Java-based tool, it automatically detects a 32-bit vs a 64-bit JRE at runtime so it has no need for specific configurations.

ctree.conf creation

The RTG Migrate tool now supports creating the *ctree.conf* configuration file.

The default for "Keep directory structure..." has been changed to Yes.

Test Connection button

The **Test Connection** button in the FairCom RTG RTG Migrate tool allows you to check the configuration by pinging the server.

"Append" option

The **Append** and **Replace Existing Files** options are available in the RTG Migrate tool.

The **Resulting Directory** layout shows indexes with their names and it highlights conflicts.

Added casesensitive keyword and textfield to set configuration file

New options were added to the RTG Migrate tool:

- Added the `casesensitive` keyword specifies if the file name and/or file dir attributes should be matched against the file path considering case sensitivity (default = yes, file and directory names are case-sensitive).
- Added a text field to set the file name of the configuration file

Options to set encryption, data compression, and transaction processing

New options have been added to the **ctmigra** command-line utility and the RTG Migrate tool. These options allow you to set encryption, data compression, and transaction support in the destination c-tree files.

ctmigra Usage:

- *-e*, *--encrypt=CIPHER* - Encrypt destination file with CIPHER algorithm.
- *-z*, *--datacompress=TYPE[;LEV][;STR]* - Compress destination data file with TYPE algorithm.

- *-t*, *--transaction=no|yes|logging* - Create destination data file with or without transaction support.

These three options enable the following configuration options on the destination file (corresponding to *-e*, *-z*, and *-t* respectively):

```
<encrypt type="CIPHER">
<datacompress type="TYPE"= level="LEV" strategy="STR">
<transaction logging="no|yes">
```

Added "Select All" and removed Check Box from directories

The user interface of the RTG Migrate has been improved as follows:

1. A **Select All** button and an **Invert Selection** check box were added.
2. A right-click context menu was added to the source files tree with **Select All** and **Unselect All** options.

RTG Migrate path separators

The handling of path separators in the paths for the source and target files has been improved in the RTG Migrate tool.

Shell Script to run RTG Migrate with proper environment

A Shell Script has been created to run RTG Migrate with the proper environment. Two different shell scripts are available:

- *RTGMigrate.sh* is for all the *NIX except AIX
- *RTGMigrate.aix.sh* is a special version for AIX

5.2 ctmigra Updates

Options to replace existing file and disable batchaddition

This modification introduces a new FairCom RTG **ctmigra** option to replace the existing destination file. The option can be specified on the command line using either the single-dash or double-dash syntax:

```
--replace
-r
```

A 0 value for the *--batch-size* / *-b* option disables <batchaddition> / <bulkaddition> logic in FairCom RTG.

If both *-c* and *-s* / *-d* options are used, an error will be returned as they are mutually exclusive.

ctmigra --quiet and --verbose options to select output information

These options for the **ctmigra** command-line utility suppress all output (*--quiet* or *-q*) or select the information to be sent to stdout (*--verbose=LEVEL* or *-v*). The *--verbose* parameter is a bitmask which can combine the following values:

1 - show message about final result

2 - show percentage progress

4 - show time spent in migration phases (read, write, finalize)

For example, to display everything, use *--verbose=7*, which is 1+2+4. If not specified, the *--verbose* level is *3* (1+2).

The *--quiet* option is identical to *--verbose=0*.

Support for ExtFH

ExtFH support has been improved in the latest version of **ctmigra**. With this support, **ctmigra** provides an easy way to migrate data from compilers that use ExtFH into an application that uses FairCom RTG.

Updates include:

- support for mapped names in ExtFH redirinstances
- check for source file not found
- display of error message when library cannot be loaded

ctmigra displays progress

The **ctmigra** utility now displays the progress of the migration process whle it works.

ExtFH now returns number of records on open

This modification changes the ExtFH interface to return the number of records when a file is open. The number of record is stored in the fcd-relative-key member of the FCD structure following the *ExtFH specifications* *http://documentation.microfocus.com/help/index.jsp?topic=%2FGUID-0E0191D8-C39A-44D1-BA4C-D67107BAF784%2FHRFLRHEXFH01.html*.

ctmigra: Progress display for ExtFH

This modification introduces support for displaying progress of migration process for ExtFH file types.

Detects missing or invalid library

This modification enhances **ctmigra** error messages displayed when the FairCom RTG library or specified external library is missing or invalid (for example, due to 32/64-bit incompatibility).

Error messages have been improved

In FairCom RTG V2, error messages displayed by the **ctmigra** utility have been enhanced as follows:

- It now displays actual file names instead of $SOURCE$ and $DEST$.
- It now displays an error message if the source or destination c-tree Servers are not running (c-tree error 133).
- It now displays an error message if the user/password is not correct (c-tree error 450/451) or GUEST logon is disabled (c-tree error 470).
- It now displays an error message if c-tree Server is not valid due to OEM version incompatibility (c-tree error 530).
- It now displays an error message if the specified external library cannot be loaded.

Behavior Change: This modification changes the behavior of the **ctmigra** tool.

ctmigra added to FairCom RTG cmdline and guitools.java directories - Windows

The **ctmigra.exe** utility has been added to the FairCom RTG *cmdline* and *guitools.java* directories. The documentation reflects the *cmdline* location. It is not meant to be used directly from the *guitools.java* location because it is there only so it can be used by the RTG Migrate tool.

More ctmigra enhancements

These enhancements were made to the **ctmigra** tool:

1. **ctmigra** now automatically creates a destination directory if it does not already exist.
2. A new command-line option *-a* / *--append-ext* to append index extension instead of replacing data extension.

6. FairCom RTG Graphical Tools

With a focus on usability, this release of FairCom RTG includes enhancements to the rich set of graphical tools.

Important improvements have been made to RTG Migrate, RTG Config, and c-treeACE SQL Explorer.

6.1 RTG Migrate

RTG Migrate has a new wizard to guide your way into FairCom RTG. For details, see the **RTG Migrate** (page 46) section.

6.2 RTG Config

RTG Config simplifies configuration with a new Basic Configuration wizard.

See the **RTG Config** (page 35) topic for details.

6.3 c-treeACE SQL Explorer

c-treeACE SQL Explorer has been updated to handle the specialized requirements of sqlized tables.

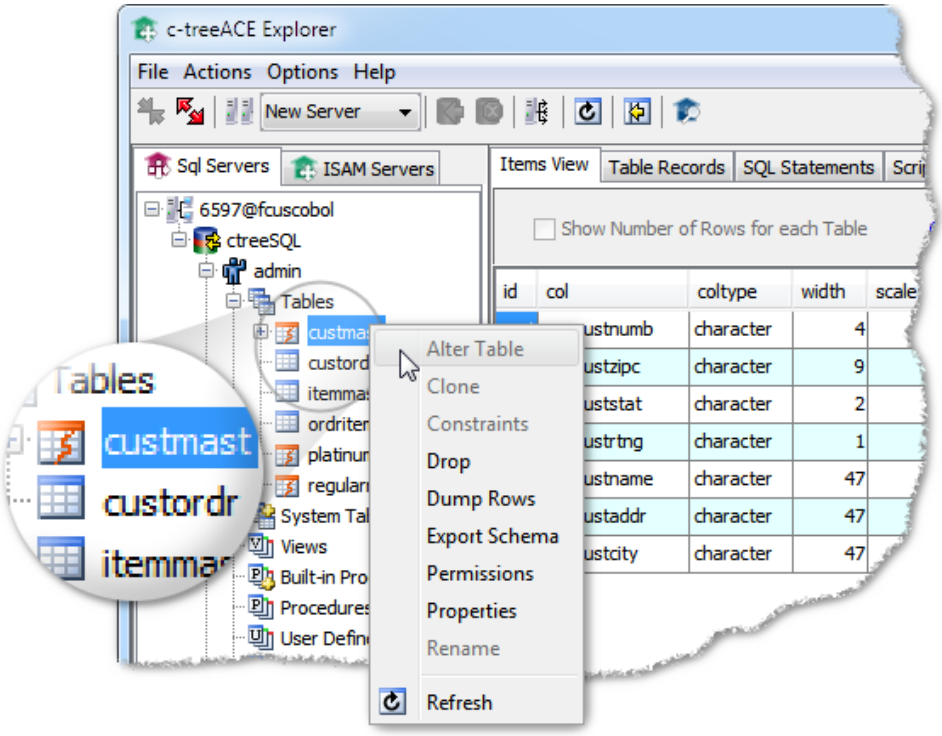
For more information and a peek at the screens, see the topic titled **Navigate Your SQL Data with SQL Explorer** (page 5).

More new features are now available in c-treeACE SQL Explorer allow you to:

- *Create indexes on sqlized tables* (page 53)
- *Check for Bad Records after migration* (page 53)

Viewing Sqlized Tables

c-treeACE SQL Explorer and c-treeACE Explorer show linked tables in a different color (a reddish orange). Tables that were also sqlized have a jagged "**S**" (or lightning bolt) to indicate they were sqlized and linked. The **custmast** table in the image below is a sqlized and linked table:



When you right-click a table, the context menu will display only the options that are available to that table (all other options are dimmed), as shown in the image above. Some options available to regular tables, such as **Alter Table**, **Clone**, and **Constraints**, are disabled because they are not available for sqlized tables.

Create indexes on sqlized tables with c-treeACE SQL Explorer (.NET & Java) context menu

As part of the FairCom RTG support for creating SQL indexes in COBOL tables, the .NET and Java versions of the c-treeACE SQL Explorer now offers a **Create** option in the context menu. This option allows creating SQL indexes on tables that have been sqlized from COBOL or BTRV sources.

Check Bad Records button in c-treeACE SQL Explorer (.NET & Java)

A new **Check Bad Records** button was added to the Table Records panel. This button is visible for FairCom RTG tables only.

7. FairCom RTG Command-Line Utilities

This release of FairCom RTG includes a more robust set of command-line utilities to help you get your system up and running. The following fixes/enhancements have been made to these tools.

7.1 ctutil

Create permanent index on-the-fly

A new function has been added to **ctutil** to create a permanent index.

```
ctutil -idximg
```

Option to dump configuration in XML format

A new option, *-l* (lower case "L") is available to **ctutil** to dump the configuration in XML format before the actual **ctutil** output.

Scan all records of a table to find conversion errors easily

This addition to the **ctutil** gives you a command to scan all records of a table to find conversion errors easily.

ctutil -check option (-x) to scan a file for integrity issues

The **ctutil** *-check* command was implemented to verify only the integrity of the index structure and did not perform any check on data and key consistency. The functionality has now been expanded to optionally perform extended verifications. It now has the following optional flags:

- *-x* - Perform extended tests matching records with keys on all indexes
- *-k=index* - Perform extended tests matching records with keys only on specified index

ctutil -clone to create empty copy of existing file

This modification introduces a new **ctutil** *-clone* command to create an empty copy of an existing file.

Usage:

```
ctutil -clone  source_file dest_file
```

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

ctutil -run to execute multiple ctutil commands

This modification introduces new **ctutil** command *-run* to execute multiple **ctutil** commands in a single **ctutil** run. The *-run* command takes one parameter which is a command list: a text file containing a list of **ctutil** commands.

This command provides the ability to execute multiple **ctutil** commands in a single run to avoid program initialization overhead.

Usage

```
ctutil -run  command_list_file
```

where:

- *command_list_file* is a text file that contains one **ctutil** command on each line followed by all its required parameters. The **ctutil** commands start with a dash character, therefore each line that does not begin with a dash (excluding any leading space character) is considered a comment and is ignored.

Example

```
ctutil -run  commands
```

The text file *commands* contains the following commands to create a file called *custmast* using definitions found in *custmast.xdd*, add records to it from *custmast.txt*, and link it to SQL. The final line in the file is a comment because it does not begin with a dash:

```
-make custmast custmast.xdd
-load custmast custmast.txt
-sqlize custmast custmast.xdd ADMIN ctreeSQL
These commands create a file, add records to it, and link it to SQL
```

Return

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

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

ctutil -sqlcheck enhanced to return all problems in the table

The **ctutil** *-sqlcheck* facility verifies if a table, given an XDD, has records causing conversion errors in SQL. In its implementation, it stops on the first error it encounters. This functionality has been expanded to print out all errors in all records or to print out all errors on the first failing record. The default value is to stop on the first error.

Usage:

```
ctutil -sqlcheck  file xdd_file [-conv=convention_ID] [-show=show_type]
```

- *file* - File name without extension
- *xdd_file* - Path to a data definition file in XML format
- *convention_ID* - Optional numeric storage convention ID:
  A - ACUCOBOL-GT (default)
  B - MBP COBOL

D - Data General
I - IBM
M - Micro Focus
N - NCR COBOL
R - Realia COBOL
V - VAX COBOL

- *show_type* - level of errors to show:
  all - show all errors in the table
  first - show the first error in the table
  firstrec - show all errors in the first problematic record in the table

ctutil -sqlcheck enhanced to print on stderr errors during XDD parsing and interpretation

The **ctutil** *-sqlcheck* command shows the errors that may occur converting data. If the XDD file contains an error, the utility reports an error with no information of the error type. The logic has been modified so this command will show an error message indicating the source of the problem.

-test

Check the configuration and connection to servers. The syntaxes are:

```
ctutil -test config [file]
ctutil -test connect
ctutil -test filerules
```

- Running **ctutil** *-test config* option checks the configuration. If file is specified, it also checks which configuration instance matches the specified file.
- Running **ctutil** *-test connect* checks that all servers defined in the configuration (with the `<instance server>` attribute) are reachable.
- Running **ctutil** *-test filerules* will print the file rules in the order they will be considered by FairCom RTG at runtime when matching a filename:

```
ctutil -test filerules
Initialized from (ctree.conf)

<file name="*" dir="mydir" casesensitive="yes" type="*"/>
<file name="myfil" dir="*" casesensitive="yes" type="*"/>
<file name="*" dir="*" casesensitive="yes" priority="-32767" type="*"/>

Operation completed successfully.
```

Note: Running the -test command with no additional specifications will run the -test config command by default.

ctutil -test filerules option to print the file rule sequence for file matching

This useful new **ctutil** option prints out the list of file rules in the sequence they are considered when matching file names. This feature is useful for debugging and to understand why a file gets matched to an unexpected rule.

The new **ctutil** *-test filerules* option will print the file rules in the order they will be considered by FairCom RTG at runtime when matching a filename:

```
ctutil -test filerules
Initialized from (ctree.conf)

<file name="*" dir="mydir" casesensitive="yes" type="*"/>
<file name="myfil" dir="*" casesensitive="yes" type="*"/>
<file name="*" dir="*" casesensitive="yes" priority="-32767" type="*"/>

Operation completed successfully.
```

ctutil -upgrade switch

Note: If you are upgrading to FairCom RTG V2 from prior releases, you will need to run the **ctutil** *-upgrade* command.

ctutil *-upgrade*

This option upgrades the file to the current configured format. It basically copies the file using the definitions in the configuration file. With this switch it is possible to take an existing data file and upgrade it to the latest format.

Usage:

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

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

This capability gives the customer a tool to upgrade an existing file to match the current settings in *ctree.conf* and/or the current FairCom RTG file format. This switch makes it possible to upgrade an existing data file to the latest format. For example, it would be used if a revision changed the file's physical layout (e.g., altering the header). FairCom RTG logic attempts to handle files in older formats, however it may be necessary to upgrade files after installing a new revision to take advantage of new features or to correct bad behavior.

If no destination file is included, the file will be upgraded in place to include the ctFlexRec attribute.

Upgrade specific attributes

- *-ctattr* - This option was introduced in RTG V5, and allows a table to participate in hot alter table operations and other features requiring complete schema information. This attribute forces an existing RTGI resource to be upgraded to the latest RTGI v2, which is required for this advanced support. When the -ctattr option is specified, the file is not re-created to match

the current configuration. Instead, only its RTG resource is converted to the <ctattr> format. This is faster than re-creating the file, especially if the file has large amount of data records. Existing RTG tables from prior versions remain compatible with current RTG versions. However, they will lack certain capabilities until their internal resources (such as RTGI) are updated.

ctutil -unload option -k to read/write records in index order

The **ctutil** *-unload* option exports data to a sequential file. This modification introduces a new option, *-k*, to the **ctutil** *-unload* command to read (and in turn write) records in the specified index order. To read records in primary key order, specify *-k=1*. If not specified, records are read using default key (primary key).

Usage

The new usage for **ctutil** *-unload* command is as follows:

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

- *file* - File name without extension
- *seq_file* - The destination file name including the path
- *-b* - Destination file is in binary sequential format (default)
- *-t* - Destination file is in line sequential format
- *-p* - Destination file is in BTRV ASCII format
- *-v* - Destination file has variable-length records
  - *[2|4|8]* - record length is stored in 2 or 4 (default) or 8 bytes
  - *[n|x]* - record length is stored in COMP-N (default) or COMP-X format
- *-k=index* - Reads and writes records in specified index order (*-k=1* for the primary key)

-sqlcheck

This **ctutil** option verifies that the data is correct in sqlized tables. The *-sqlcheck* option is used to verify that data in a file is compatible with SQL definitions. The command scans all records of the given file using the primary key index and stops at the first conversion error encountered showing an error message, the record number, schema and field name of the incorrect data.

Usage:

```
ctutil -sqlcheck  file xdd_file [-conv=convention_ID] [-show=show_type]
```

The *-sqlcheck* option checks SQL definitions against data in a file. Valid parameters are:

- *file* - File name without extension
- *xdd_file* - Path to a data definition file in XML format
- *convention_ID* - **COBOL Users Only**: Optional numeric storage convention ID; see *convention_ID values.*
- *show_type* - level of errors to show:

all - show all errors in the table

first - show the first error in the table

firstrec - show all errors in the first problematic record in the table

Example:

The usage and output of the command is as follows:

```
C:\>ctutil -sqlcheck filecompa filecompa.xdd -conv=A
ctutil Version 11.2.0.5893-160315
Initialized from (ctree.conf)

Checking filecompa...

ascii value is not a digit. Record#: 8 Schema#: 0 Field: NS

Operation completed successfully.
```

Example:

This example reveals numeric sign issues

```
>ctutil  -sqlcheck  CUSTOMER.DAT  CUSTOMER.xdd   -show=all
ctutil
Version 6.0.0.324-250123 - ACUCOBOL Edition
Initialized from 'ctree.conf'

Checking CUSTOMER.DAT...

Record#: 0000002 Schema#: 005 Field: keycharcnt Error: unexpected value for sign [8c] valid values [0d|0f]
(last nibble only)
Record#: 0000002 Schema#: 005 Field: elecharcnt Error: unexpected value for sign [0c] valid values [0d|0f]
(last nibble only)
Record#: 0000002 Schema#: 005 Field: maxelecnt Error: unexpected value for sign [0c] valid values [0d|0f]
(last nibble only)
Record#: 0000008 Schema#: 005 Field: keycharcnt Error: unexpected value for sign [0c] valid values [0d|0f]
(last nibble only)
Record#: 0000008 Schema#: 005 Field: elecharcnt Error: unexpected value for sign [0c] valid values [0d|0f]
(last nibble only)
Record#: 0000008 Schema#: 005 Field: maxelecnt Error: unexpected value for sign [0c] valid values [0d|0f]
(last nibble only)
Record#: 0000019 Schema#: 005 Field: keycharcnt Error: unexpected value for sign [0c] valid values [0d|0f]
(last nibble only)
Record#: 0000019 Schema#: 005 Field: elecharcnt Error: unexpected value for sign [0c] valid values [0d|0f]
(last nibble only)
Record#: 0000019 Schema#: 005 Field: maxelecnt Error: unexpected value for sign [0c] valid values [0d|0f]
(last nibble only)

Operation completed successfully.
```

ctutil -load -r2 option

The *-r2* option for the **ctutil** *-load* command specifies that the binary sequential input file was created with the **Recover2** utility.

ctutil sqlize options password requirement removed

The **ctutil** *-sqllink*, *-sqlunlink*, and *-sqlize* commands do not require the admin password: this is now an optional parameter. When the password is not specified, the user in *ctree.conf* for the targeted server needs to be DBA or have RESOURCE permissions on the database.

7.2 ctstat reports more detailed timing information for performance profiling

The FairCom Server function monitor now shows specific FairCom RTG function timings (such as RTG_OPEN, RTG_READ, etc.). In previous releases, the function timings display grouped all of these functions into a single FairCom RTG function time and call count. This information about critical FairCom RTG operations provides useful information for performance profiling.

In this revision, the FairCom RTG functions have been separated into their own function timing entries so we can know which functions are being called and taking the most time when profiling performance. In the example below, note the separate entries for RTG_OPEN and RTG_VERSION:

```
function              counter       secs
CLISAM                     10      0.004
FRSVSET                    25      0.003
NXTVSET                    25      0.001
INTISAM                    10      0.003
SETOPS                     20      0.000
CTSQLCTL                   10      0.000
GETDODAX                    4      0.000
COMMBUF                    12      0.001
SYSCFG                     10      0.000
RTG_OPEN                   10      0.176
RTG_VERSION                10      0.001
ctSNAPSHOT                 26      0.003
```

If an old version of **ctstat** is used with a new server, the FairCom RTG function names will show up as UNKNOWN.

7.3 ctadmn shows detail in last function

When a FairCom RTG function is called, FairCom Server now provides more specific information about the last function called for the connection. This allows monitoring utilities such as **ctadmn** to show the FairCom RTG function name rather than a generic name. For example, previously **ctadmn** would show:

```
UserID: ADMIN                           NodeName:
Task 19                                 Communications: FSHAREMM
        Memory:  51K       Open Files: 0      Logon Time:   0:07
        Tran Time:   --    Rqst Time:   0:07  NoRequest  Rqst# 198 RTG
```

Now it shows:

```
UserID: ADMIN                           NodeName:
Task 19                                 Communications: FSHAREMM
        Memory:  51K       Open Files: 0      Logon Time:   0:07
        Tran Time:   --    Rqst Time:   0:07  NoRequest  Rqst# 248 RTG_VERSION
```

7.4 xddgen

If you want to provide SQL access to your data, you will need to define the record schema through an XDD (eXternal Data Definition) file. The XDD is an external XML file that is stored as a special resource within the data file created through the processes described in this chapter. If your COBOL compiler can provide an XFD, you can generate the XDD from it using **ctutil** *-xfd2xdd* or **ctutil** *-sqlize*. If you do not have an XFD, the **xddgen** utility can generate the XDD from your COBOL source code.

This command-line utility analyzes your COBOL program to create the XDD. You will specify the COBOL program file as input. A variety of other parameters allow you to specify the dialect of COBOL (ACCUCOBOL, Micro Focus, IBM, etc.), source format (free or fixed), directories, and other options.

This section explains enhancements to this utility.

xddgen now allows names larger than 31 chars

The **xddgen** utility is able to generate the XDD if a field name is larger than 31 characters to allow more flexibility in generating the XDD. Two configuration entries accommodate this situation:

1. `user-defined-names-len` (defaults to 31) - Can be any number between 31 and 64. It determines the number of significant chars in fields and table names.
2. `check-redefinition` (defaults to yes) - If yes, check for field names redefinitions that would cause having multiple SQL fields with the same name and if so, error out.

The behavior when names larger than `user-defined-names-len` is encountered has been changed so that, instead of generating an error, it produces a warning and truncates the name to the maximum allowed length.

Suppress Dash or Replace with Underscore

A boolean **xddgen** configuration behavior allows suppressing the dash character ("-") when it is present instead of substituting it with an underscore ("_"). The default is `no`, which preserves the past behavior in which a dash is substituted with an underscore.

```
suppress-dash-in-field-names
```

- *no* (default) - Dashes are substituted with underscores.
- *yes* - Dashes are eliminated from the file name rather than being substituted.

Example:

You can create a new configuration file and set this feature as desired. When you run **xddgen**, the *-std=* switch can be used to point to the new configuration file:

1. Copy the configuration file specified by the **xddgen** *-std=<dialect>* configuration switch. For example, if *-std=mf*, make a copy of the configuration file *mf.conf* found in the *conf* directory.
2. Rename the configuration file copied in step 1.
3. Add (or change) for setting for `suppress-dash-in-field-names: yes`

4. Run **xddgen** *-std=<new_name>* replacing *<new_name>* with the new configuration file name assigned in step 2.

Configuration Files Directory

The **xddgen** configuration files can be loaded from the *config* directory under the **xddgen** executable directory. This allows calling **xddgen** from any directory without worrying about copying the *config* sub-directory.

A number of **xddgen** behaviors are controlled by configuration files. The default configuration file, *default.conf*, can be modified using either the *-std* or the *-conf* command-line switches:

- When *-std=AAA* is used, the configuration file name is *AAA.conf*.
- When *-conf* is used, the configuration file name is the exact name passed in with no changes. If no path or a relative path is specified, it will be based on the current working directory.

When *-conf* is not used, the file is loaded from a directory using the following criteria:

1. If the environment variable COB_CONFIG_DIR is specified, that is the only directory that will be considered.

   else
2. The search for the configuration file will include the *config* directory under the current working directory (i.e. the directory where **xddgen** is launched).
3. If #2 fails, the search for the configuration file will include the *config* directory under the **xddgen** invocation path (i.e., the directory from which **xddgen** was executed).

The above criteria are also used to locate "included" configuration files (one configuration file can include another and overwrite some of the values specified).

Map OCCURS DEPENDING ON into LONG VARCHAR

Logic has been implemented in **xddgen** to detect eligible `OCCURS DEPENDING ON` (ODO) fields and generate the XDD syntax to map them into LONG VARCHAR. An ODO is eligible when:

1. it is the last record element
2. the size of the `OCCURS` field is 1 byte

The **xddgen** utility can map into a LONG VARCHAR by simply using the `*>>XDD BINARY` directive on the `OCCURS`.

The field that the `OCCURS` depends on is automatically marked hidden.

Parsing improvements

This modification is part of a continuous effort to improve the ability of **xddgen** to properly parse COBOL source code.

Syntax for WITH DUPLICATES on RECORD KEY

In most COBOL compilers it is possible to specify WITH DUPLICATES on the RECORD KEY, but **xddgen** did not allow this (it failed with a syntax error). The **xddgen** syntax has been expanded to support duplicates on the record key in addition to the support for duplicates on alternate keys. A record key that allows duplicates cannot be marked as primary key in SQL.

Improved and clearer error/warning messages

When the **xddgen** utility encountered a WHEN directive that referenced a file name that did not exist, it failed with a very generic message "1813: Error: field not found in field list during filter generation" pointing to the last line of the file. The logic has been updated to track the code position of this type of error and to print a clearer message providing more information.

More xddgen enhancements

Several enhancements have been made to the latest version of **xddgen**:

Generate an XDD for EXTERNAL files

Support was added **xddgen** to generate an XDD for files defined as EXTERNAL. By default, **xddgen** does not generate XDD files for external files, but it is now possible to change the **xddgen** configuration file follows to instruct **xddgen** to generate XDD files for EXTERNAL files:

```
# If yes, generates xdd file also for "external files"
# Value: 'yes', 'no'
consider-external-file: yes
```

Now when an external file is ignored, **xddgen** generates a warning message to inform about it.

Key based on a redefine

The **xddgen** utility used to return an error if a key was based on a redefine of a field or, in case of split keys, if one of the specified fields was a redefine. Now the logic has been changed to allow these conditions and produce a warning indicating the situation.

It is possible to change the behavior from the default of generating a warning to the old behavior of getting an error (or even to allow it without any warning) by changing the **xddgen** configuration entry:

```
# behaviour when a key is based on a redefine of a field.
# Value: 'ok', 'error', 'warning'
key-on-redefines: warning
```

Rationalized xddgen behavior on error

If an error occurs while generating an XDD, the XDD will not be generated (although other XDDs may be generated). The **xddgen** utility will always return a value different than 0 on the occurrence of an error. This implies it is sufficient to check the return code to see if something failed during **xddgen** execution.

7.5 Additional FairCom RTG Command-Line Tools

These command-line tools are available in FairCom RTG V3:

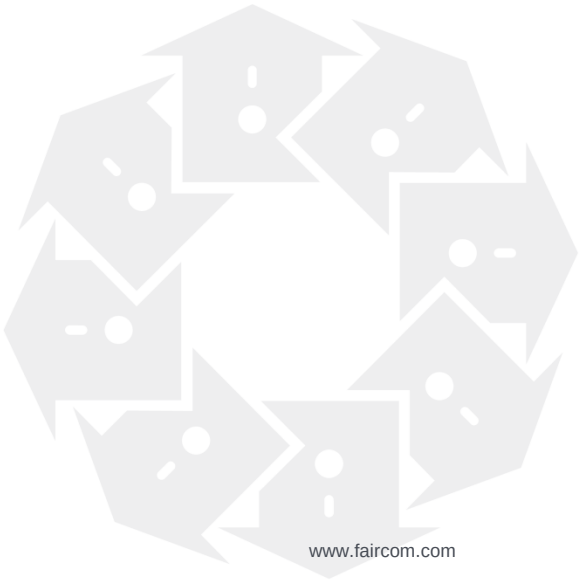
- **ctidmp** - Lists the contents of a dynamic dump file or a specific extent of a dump broken into multiple files.
- **ctpass** - Allows users to change their password.
- **ctsysm** - Monitors error, warning, and informational messages logged to the server status log, *CTSTATUS.FCS*. Allows monitoring by an automated external process, such as the Tivoli monitoring system from IBM.
- **cttrap** - Plays back a TRAP_COMM log file.
- **ctclntrn** - “Cleans” the high-water transactions marks within a c-treeACE index.
- **cthghtrn** - Displays the high-water transaction marks within an index file. This utility would typically be used when the FairCom Server transaction mark number gets too large.
- The **ctinfo** utility is now included in the FairCom RTG package. This utility retrieves *IFIL* and *DODA* structures from a c-treeACE file (and other header information).

For detailed information, see the **Command-Line Tools** (https://docs.faircom.com/doc**/cmdline/**) guide on the FairCom website.

8. Core Server Enhancements from c-treeACE

FairCom RTG Version 2 features the latest updates to the core engine. These advanced features debuted in current release of c-treeACE, V11. They are now available to FairCom RTG users.

A recap of these features demonstrates why FairCom RTG V2 gives you unprecedented stability, power, and performance.



8.1 Delayed Durability Transaction Log Mode for Performance

Delayed Durability Transaction Processing

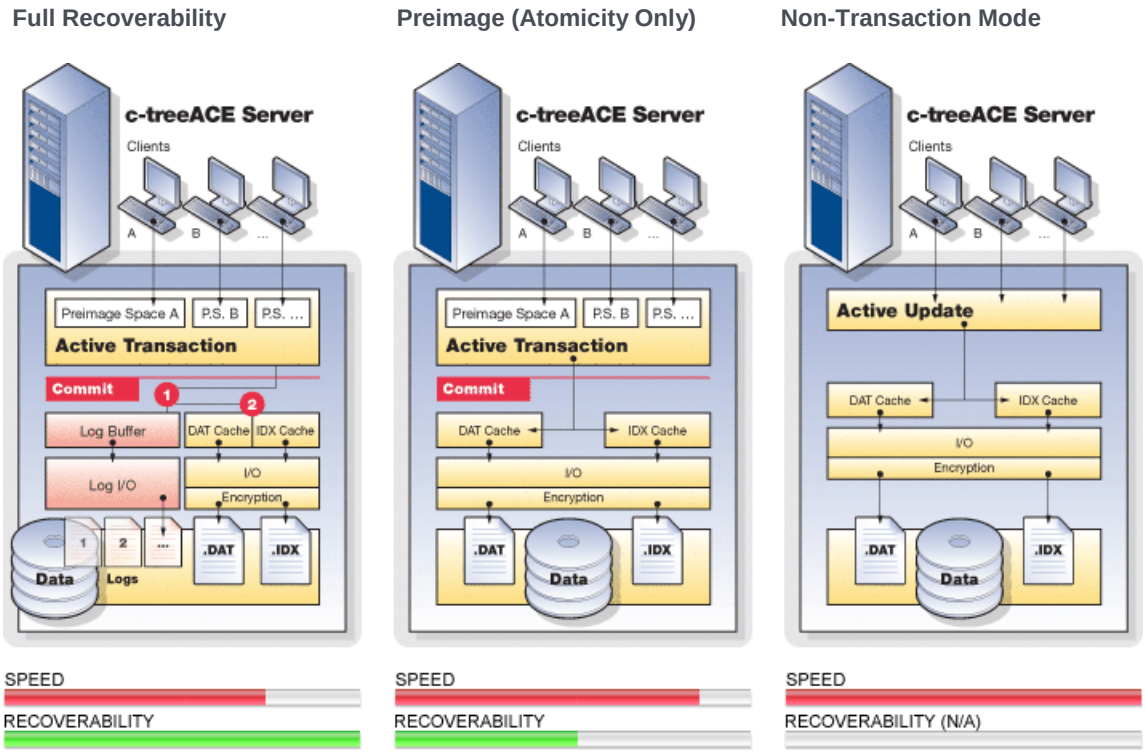
With full transaction control for complete ACID compliance, transaction logs are synced to disk with each commit operation, ensuring absolute data integrity with complete recoverability. Full durable ACID transaction control enables many powerful features not available without the availability of recoverable log data:

- Automatic database recovery
- Live database backups without rebuild on restore
- Replication
- Transaction auditing

The most critical of these is automatic recovery in case of system failure. However, full transaction control remains a critical area of database performance tuning. Database updates must be secured in write-ahead logs for guaranteed recoverability. This comes with a performance impact due to the synchronous I/O requirements ensuring data is safely persisted.

Many applications could benefit from a "relaxed" mode of transaction log writes. With today's hardware and power redundancies, it is conceivable to slightly relax full durability constraints and still maintain an acceptable data risk tolerance. The balance becomes "how much loss of recoverability can these systems tolerate?"

Allowing database administrators to balance their window of vulnerability against online performance, c-treeACE provides a new Delayed Durability feature for transaction logs.



This new transaction operation mode allows its c-treeACE transaction log updates to remain cached in its in-memory transaction log buffer as well as in file system cache, even after a transaction has committed. The challenge is to avoid index and disk updates from reaching disk before the transaction entries do. FairCom has managed to delay transaction log writes to persisted storage while guaranteeing these transaction log entries for a given transaction write to disk before any data file updates associated with that transaction are written to file system cache or to persistent storage with a modified log sync strategy. In addition, a background thread guarantees an upper bound on the total amount of time any transaction remains in a non-synced state.

This feature is enabled with the following configuration entry and takes as an argument the maximum number seconds, *N*, that will elapse between log syncs. This ensures that, after a commit, the transaction log will be synced *in no more* than *N* seconds, thereby allowing you to define your window of vulnerability risk.

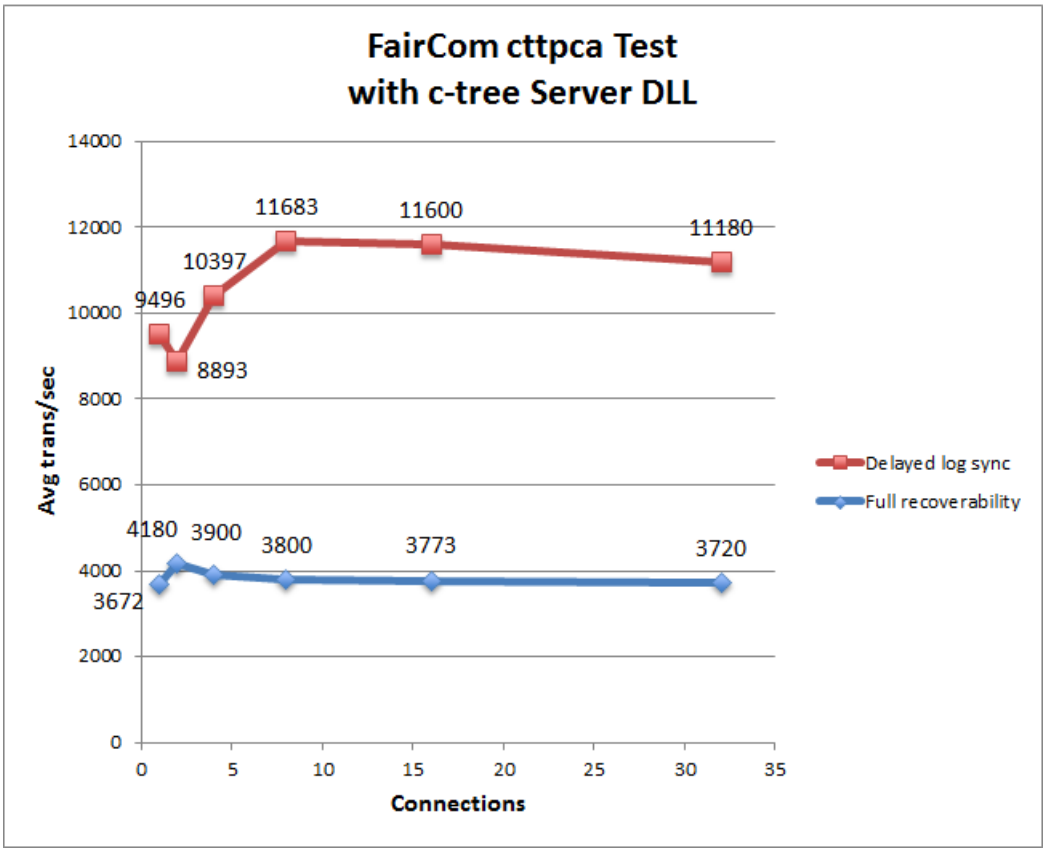
```
DELAYED_DURABILITY <N>
```

The end result can approach non-transaction performance while ensuring committed transactions to persisted storage to within less than N seconds of vulnerability. Values as low as one (1) second are shown to provide the best performance. In selected test cases, **up to 300% faster transaction throughput** has been observed. Higher values have been found to offer little, if any, additional benefit.

Performance Gains

Tests were run to compare the new Delayed Durability transaction mode with the existing transaction processing mode using `DELAYED_DURABILITY 1`, allowing the log sync to be deferred for 1 second. The results indicated that the new Delayed Durability achieved performance gains of a factor of 2 to 3 over the previous transaction processing. A setting of 1 second is recommended because it is enough for a good performance gain and higher values offer very little additional benefit.

The chart below shows performance of the new Deferred Durability transaction mode (red) and the existing standard transaction processing mode (blue):



The data was acquired using the FairCom **cttpca** test. This test was run for 30 seconds using server DLL with varying numbers of connections. New files were created before each test run. For this test, c-tree data and index caches were large enough to hold all records and index nodes.

The existing transaction processing is referred to as "full recoverability" because all transactions can be recovered. The new Deferred Durability technique can be used with *Restore Points* to allow the data to be rolled back to a known good state.

Modified Log Sync Strategy

Enabling the Delayed Durability feature activates a modified log sync strategy. Ordinarily, when a transaction commits, c-treeACE does not return from a **TRANEND()** call until log entries associated with the transaction have been written to persisted storage (e.g., your disk drive). With the new modified log sync strategy enabled only the following is ensured:

Before any data or index image is written to disk, all associated log entries are on disk.

With this modified log sync, in an extreme case, a transaction could be committed without any associated log entries on disk. Or a **TRANBEG** may be on disk, but not a corresponding **TRANEND**. This means c-tree cannot guarantee recovery of all transactions that have returned a successful commit.

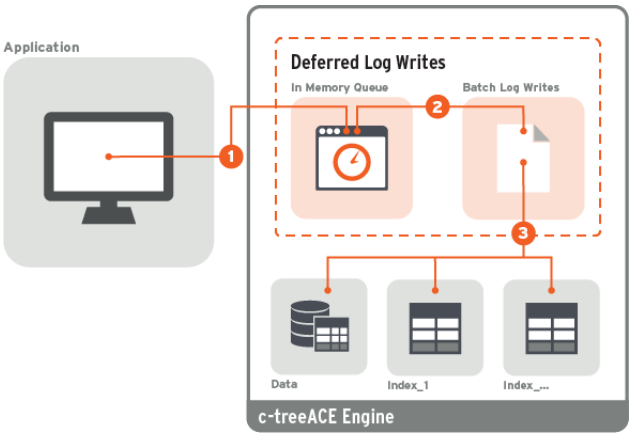
When Delayed Durability is active, the link between committing a transaction and syncing the transaction log to disk is de-coupled, and the number of log syncs is greatly reduced. Additional logic ensures that before c-tree data or index contents are written to disk, all necessary log entries are on disk. If log information is cached after a commit, the log will not necessarily be synced to persisted storage, *leaving a small vulnerable window for data loss potential in case of system failure*.

The Delayed Durability time window configuration is an added c-treeACE internal operations thread ensuring transaction log data is periodically synced on a regular basis. This log sync thread coordinates the transaction log sync to disk and guarantees that no more than N seconds can pass without a log sync to persisted storage. This thread wakes periodically and syncs accumulated transaction log data before the `DELAYED_DURABILITY` value is exceeded. A setting of 1 second is recommended as it provides excellent performance gains with minimal window of data loss in event of system failure.

Note: The transaction log buffer is always flushed when full. This buffer is defined as a 64K buffer. Any number of updates exceeding 64K will always trigger a log buffer sync to disk.

Delayed Durability Behavior

When the configuration option `DELAYED_DURABILITY` is specified in *ctsrvr.cfg* with a value greater than 0, FairCom Server does not sync its log buffer simply because a transaction commits. Instead, it syncs the log buffer if it becomes full, or if a write to the data cache requires the log buffer to be flushed to disk, or if the maximum defer time in seconds specified by the keyword is exhausted.



When this feature is in effect, a transaction commit writes to the in-memory transaction log buffer and data files in the following order:

1. **Write transaction log entries to the in-memory transaction log buffer.** The log entries are copied to the in-memory transaction log buffer and remain in the buffer until the log buffer becomes full or until a write to a transaction-controlled data or index file requires the log buffer to be written to disk. When the transaction commit call returns to the caller, there is no guarantee that the transaction log entries are on disk, and so the transaction is not guaranteed to be recoverable.
2. **Write data file updates to c-tree's data cache and/or to disk.** Depending on the options that are in effect for the file or for the database engine, this step will either write the data file updates to c-tree's data cache or write the changes to file system cache or to disk. It is not required that the changes to the data files are on disk when the transaction commits, because the transaction log entries can be used to redo the transaction if c-tree terminates abnormally before the updates to the data files are known to be on disk (which can be sometime after the transaction commits).
3. **Mark the transaction as completed.** This step causes other connections to view the key level locks as committed rather than pending commit.
4. **Unlock records.** This step allows other connections to lock and update the records that the transaction touched, if desired.

c-treeACE tracks the highest log position assigned to the contents of a data cache page for those items in preimage space that correspond to a data record. When a data-cache page is about to be written to the file system, c-tree checks if the last log byte synced to disk includes the highest required log position for the data cache page. If not, c-tree flushes the log buffer to the file system cache and syncs the file system cache to the transaction log file on disk.

Configuration Entries for Delayed Durability

`DELAYED_DURABILITY <N>` (default 0) controls whether or not to use a modified log syncing strategy.

With delayed durability enabled transaction logs are no longer sync'd to persisted storage on each commit (or other internal log buffer flush events) and instead, transaction log data is allowed to be written to the filesystem cache, and a background thread then periodically and consistently syncs transaction log contents to disk.

By allowing committed transaction entries to be written to filesystem cache and deferring the file flush can result in a very large performance gain in many cases. However, there is a trade off as a window of potential data loss vulnerability is then introduced. The period of time that transaction log contents are present in volatile filesystem cache before the flush could mean transactions already reported to the application as committed, in rare cases, might not make it to persisted storage.

With many modern storage devices there is a limited presumption that available capacitance on the system and storage device hardware that data is usually persisted even in a power outage situation, though this is not guaranteed. Thus alternate recovery strategies should be considered.

One strategy to coordinate the known state of committed database transactions with a known application state can be achieved with restore points. Restore points can be triggered by an application to create a known point in time where the application and the database are in sync. A restore point creates a special database transaction log checkpoint that can be later referenced by the application as a known good start point.

With delayed durability enabled, it is recommended to consider the use of restore points for a robust recoverable solution.

- When `DELAYED_DURABILITY` set to 0 disables delayed durability/
- When `DELAYED_DURABILITY` is set to a positive value, <*N*>, the new strategy is in use and the log sync is guaranteed to occur within <*N*> seconds. A setting of 1 second is recommended because it results in a good performance gain (higher values offer very little additional benefit). The following configuration options are set as shown below:

`SUPPRESS_LOG_FLUSH`	YES	(no idle flush of transaction files)
`SUPPRESS_LOG_SYNC`	YES	
`IDLE_TRANFLUSH`	-1	
`COMMIT_DELAY`	-1	(no commit delay)
`FORCE_LOGIDX`	ON	(all transaction indices use *ctLOGIDX*)
`COMPATIBILITY LOG_WRITETHRU`	Disabled	

Note: If the configuration file has one or more of these configuration entries set inconsistently after the `DELAYED_DURABILITY` entry, the server logs a message to *CTSTATUS.FCS* and continues to start, disabling any incompatible options after processing the configuration file.

Warning

When `DELAYED_DURABILITY` is enabled, recently committed transactions could be lost if FairCom Server terminates abnormally. For automatic recovery to succeed after FairCom Server terminates abnormally, either of the following should be considered.

1. The application must write a restore point to the log (using the **ctflush** utility or calling **ctQUIET()** with mode of *ctQTlog_restorepoint*) such that a restore point exists prior to the time the server terminated abnormally. In this case, automatic recovery recovers to that restore point.

   or

2. *ctsrvr.cfg* must contain the option `RECOVER_TO_RESTORE_POINT NO`, indicating that no restore point is needed. In this case, automatic recovery recovers to the last data that was written to the log on disk. This is the default configuration.

See Also

- *Transaction Processing Keywords /doc/ctserver/65411.htm* in the *FairCom Server Administrator's Guide* (https://docs.faircom.com/doc/*ctserver/*)

Controls for Performance AND Safety of Non-Transaction Updates

(In this discussion, a cache page that has been updated and has not yet been written to the file system is called a "dirty page.")

c-treeACE offers multiple levels of transaction protection for your data. Some applications do not require the recoverability full transaction provides for performance reasons. However, these applications may be vulnerable to data loss should system failure occur. If FairCom Server terminates abnormally, updates to data and index files that are not under full transaction control are lost if those updates have not yet been written from c-tree's in-memory data and index caches to the file system. The following factors typically reduce the number of dirty pages that exist:

1. When an updated cache page is being reused, the updated page is written to the file system cache.
2. When all connections close a c-tree file, FairCom Server writes the updated pages to the file system cache before closing the file.
3. An internal thread periodically checks if FairCom Server is idle, and if so it writes updated pages to the file system cache.

However, the combination of using very large data and index caches, keeping files open for long periods of time, and having constant activity on the system increases likelihood that more dirty cache pages exist.

It is possible to define a vulnerability window limiting potential loss of updates for your non-transaction data and index files. FairCom Server supports options to write dirty cache pages to the file system within a specified time period. This means that no more than a set amount of time can pass where data is not flushed to disk.

The following FairCom Server configuration options set the time limit (in seconds) that a data cache page or index buffer can remain dirty before it is written to the file system cache. The default time limit is `IMMEDIATE` (flush updates for non-tran files to the file system as soon as

possible). Specify a value to set a time limit in seconds. Specify `OFF` to disable time limit-based flushing.

```
NONTRAN_DATA_FLUSH_SEC    <time_limit_in_seconds>
NONTRAN_INDEX_FLUSH_SEC   <time_limit_in_seconds>
```

These options can also be changed using the **ctSETCFG()** API function and using the **ctadmn** utility.

Monitoring Non-Transaction Data Flush

Fields have been added to the system snapshot structure (*ctGSMS*) to hold the non-tran flush settings and statistics. See *Time limit on flushing updated data and index cache pages for TRNLOG files in the c-treeACE Programmer's Reference* (https://docs.faircom.com/doc/*ctreeplus/*).

Tuning Non-Transaction Data Flush

These FairCom Server configuration options set the number of counter buckets for the dirty data page and index buffer lists:

```
NONTRAN_DATA_FLUSH_BUCKETS    <number_of_buckets>
NONTRAN_INDEX_FLUSH_BUCKETS   <number_of_buckets>
```

The default number of counter buckets is 10. Setting the option to zero disables the use of the counter buckets.

Non-Transaction Flush Diagnostics

The configuration option `DIAGNOSTICS BACKGROUND_FLUSH` can be used to enable logging of flush thread operations to the file *NTFLS.FCS*.

The configuration option `DIAGNOSTICS BACKGROUND_FLUSH_BUCKETS` can be used to enable logging of flush counter bucket statistics to the file *NTFLSBKT.FCS*. Each time a text snapshot is written to the file *SNAPSHOT.FCS* file, the bucket statistics are written to the *NTFLSBKT.FCS* file.

Monitoring Delayed Durability Performance

SNAPSHOT Statistics

SNAPSHOT statistics data collection related to transaction log flushing have been updated as follows:

1. The `LOG FLUSH REQUESTS` includes a “data cache” value which is the number of log flushes instigated by the Delayed Durability requirement that no data image will go to disk until the log entries associated with the image have been flushed to disk. It includes a “# checks if ...” that shows the number of times a data cache write operation checked to see if a log flush was required.
2. A new section, `DATA CACHE LOG FLUSH REQUEST DETAILS`, breaks down the data cache flush requests into the operations that trigger the flush.

Below are excerpts from three different *SNAPSHOT.FCS* files:

1. The first using `DELAYED_DURABILITY` **On**, which sets `FORCE_LOGIDX` **ON**
2. The second with `DELAYED_DURABILITY` **OFF** and `FORCE_LOGIDX` **ON**
3. The third without `DELAYED_DURABILITY` **OFF** and `FORCE_LOGIDX` **OFF**

The same single-threaded application program was run that adds 100,000 ISAM records, each with three indices, and each add is in a transaction.

Recall that when an index has a file mode with *ctLOGIDX* enabled, additional transaction log entries permit automatic recovery to repair a damaged index at the site of the damage instead of rebuilding the entire index. *ctLOGIDX* affects the log flushing dynamics.

SNAPSHOT statistics can be output with the **ctstat** Statistics Utility *-text* option.

DELAYED_DURABILITY, which sets FORCE_LOGIDX ON

```
LOG FLUSH REQUESTS
     checkpoint/endlog/commit/abort tran: 127
                               begin tran: 518
                          LOGIDX option: 20909
                        file 1st update: 20
                            replication: 0
                             data cache: 7
                                  TOTAL: 21581
```

← New statistic

```
# checks if cache write forces log flush: 1609
```

← New statistic

```
    DATA CACHE LOG FLUSH REQUEST DETAILS
                     cache aging: 0
 updated page to be re-assigned: 0
            flush on file close: 3
                      checkpoint: 0
                         CTFLUSH: 0
                     ctrbktfls(): 0
                           other: 4
```

← New section breaks down data cache value above

DELAYED_DURABILITY OFF and FORCE_LOGIDX ON

```
LOG FLUSH REQUESTS
     checkpoint/endlog/commit/abort tran: 100137
                               begin tran: 561
                          LOGIDX option: 20846
                        file 1st update: 20
                            replication: 0
                             data cache: 0
                                  TOTAL: 121564

# checks if cache write forces log flush: 0

    DATA CACHE LOG FLUSH REQUEST DETAILS
                     cache aging: 0
 updated page to be re-assigned: 0
            flush on file close: 0
                      checkpoint: 0
                         CTFLUSH: 0
                     ctrbktfls(): 0
                           other: 0
```

DELAYED_DURABILITY OFF and FORCE_LOGIDX OFF

```
LOG FLUSH REQUESTS
     checkpoint/endlog/commit/abort tran: 100135
                               begin tran: 17207
                          LOGIDX option: 0
                         file 1st update: 20
                             replication: 0
                              data cache: 0
                                   TOTAL: 117362

# checks if cache write forces log flush: 0

    DATA CACHE LOG FLUSH REQUEST DETAILS
                      cache aging: 0
 updated page to be re-assigned: 0
            flush on file close: 0
                       checkpoint: 0
                          CTFLUSH: 0
                      ctrbktfls(): 0
                            other: 0
```

Comparison

Comparing the three different *SNAPSHOT* files reveals the following:

1. The total number of log flushes is significantly lower with Delayed Durability on.
2. The reduction in total log flushes is primarily from eliminating the requirement to ensure log entries are on disk before a transaction commit can return.
3. *LOGIDX* causes a significant number of log flushes. However, when *LOGIDX* is not used, other flushing requirements imposed by indices still generate a comparable number of log flushes related to *TRANBEG* entries getting flushed.
4. The data cache impact on the number log flushes is quite small.

8.2 New Stored Procedure Development Frameworks

c-treeACE SQL stored procedures (SP), user defined functions (UDF), and triggers provide powerful extensions to standard SQL operations allowing centralized business rules and enforcement of those rules.

The popularity of this feature has prompted several big new FairCom developments greatly extending development ease and platform extensibility:

- **NetBeans Plugin** - Cross-platform Java is the current framework used to develop SP, UDF, and triggers for c-treeACE SQL. To ease development, a NetBeans/Eclipse plug-in has been created to provide a complete IDE for Java procedure development. This is immediately available to users of Java stored procedures.
- **.NET Support** - Extending platform flexibility for Windows developers, c-treeACE SQL now supports stored procedures, user-defined functions, and triggers for the .NET framework. New Visual Studio integration provides simplified development of SP, UDF, and triggers in any .NET language that compiles to the CLI. .NET assembly dynamic link libraries are now easily deployed to remote servers from a convenient development platform. Templates for C# and Visual Basic are provided.
- **Dump and Deploy Utilities** - To simplify the process of deploying SP, UDF, and triggers to multiple servers, scriptable command-line utilities can now be used to dump existing SPs, UDFs, or triggers and deploy to remote servers. These are provided to assist application deployments.

8.3 NetBeans

c-treeACE SQL has supported basic procedural development leveraging Java due to its language flexibility and cross-platform feature. While very complex procedure development was possible, a full-featured development environment was not easily supported.

NetBeans is a complete and easy-to-use IDE for Java developers, which provides a natural framework for Java stored procedure development. FairCom now provides a NetBeans plug-in enhancing stored procedure development and testing. This includes expected modern features such as syntax checking and in-line debugging.

This plug-in allows developers to compile and test Java procedure code via a client-side interface and deploy final procedure code to the server. Debugging within the IDE is possible by retrieving procedure source code from the server. It is not necessary to provide a full JDK on the server as development can now be centralized at the developer workstation.

8.4 Utilities to Dump and Deploy SP, UDF & Triggers

To simplify the process of deploying SP, UDF, and triggers to multiple servers, these command-line utilities can be used to dump and deploy:

- **dbschema** - Dump utility option for exporting compiled SP, UDF, and triggers.
- **dbdeploy** (https://docs.faircom.com/doc**/cmdline/dbdeploy-util.htm**) - Deploy utility for loading exported SP, UDF, and triggers on other servers.

8.5 Millisecond Timestamp Resolution

Data acquisition has become much faster and along with it, the necessity for higher time resolution. Due to popular request, we now introduce millisecond timestamp resolution for c-tree time types. This support is available across multiple c-tree APIs, including core c-tree support, FairCom DB API and c-treeACE SQL. This support required addition of extended data types, which you should understand may impact backward compatibility with some applications.

This support includes the following changes:

- A new data type, *CT_TIME_MS,* was added to store time with millisecond support.
- A new base c-tree type, *CTTIMEMS*, was added to hold a time with millisecond format.

Searching an index based on *CT_TIME* is problematic when milliseconds are significant since the index information does not contain milliseconds. SQL will error out when performing index searches on *CT_TIME* columns.

Timestamps with milliseconds are supported on existing timestamp data with no conversion. A minor behavior change should be noted:

Prior to this release, a query similar to the example shown below would have ignored the milliseconds portion, returning records matching `'11/11/2015 12:00:15.000'`:

```
SELECT * FROM mytable WHERE mytimestamp = '11/11/2015 12:00:15.998'
```

With the changes described above, the milliseconds are now significant.

8.6 Table Lock Support

In V11 and later, c-treeACE supports file-level c-tree data file locks, or more commonly referred to as "table locks." An application can request a file lock to prevent other connections from reading from and/or writing to the file.

- A file read lock (table read lock or shared table lock) allows other connections to acquire read locks on records in the file but not write locks.
- A file write lock (table write lock or exclusive table lock) prevents other connections from acquiring read and write locks on records in the file.

These locks can be invoked using standard SQL commands or from any of the supported c-treeACE interfaces using commands similar to those shown later in this section.

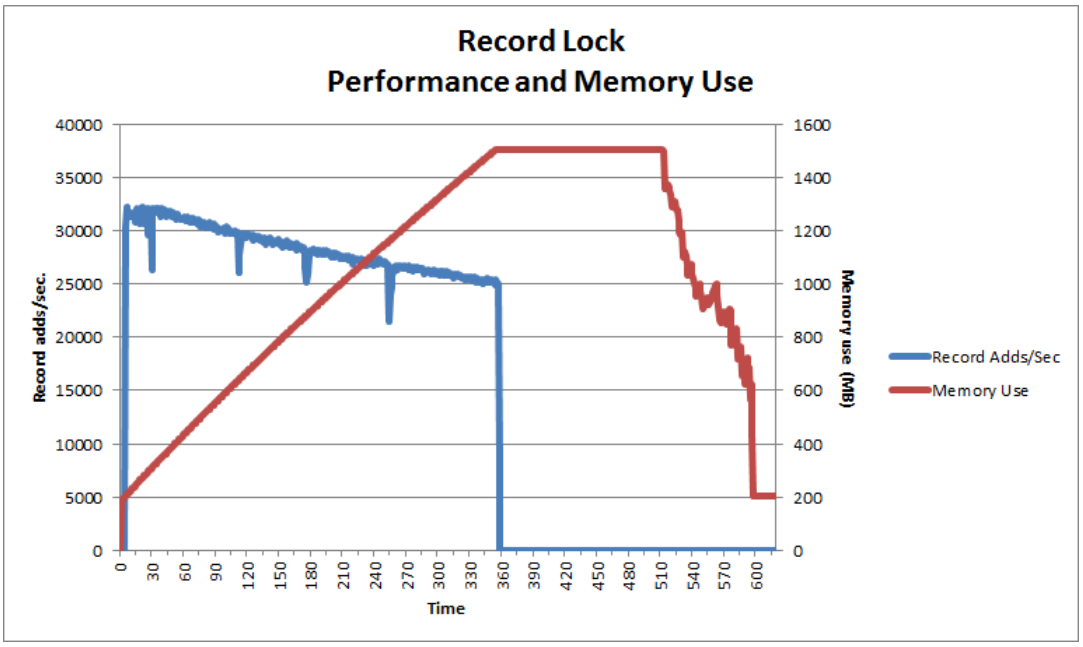
Table locks reduce resource usage with large sequences of actions on a table. A bulk add operation, for example, generates many lock requests. With a table lock the table is in an exclusive open state, thereby avoiding the necessity of managing a large number of new lock requests. This benefits both memory resource usage, as well as great gains in performance as overhead of lock management is greatly reduced.

Performance

The chart below shows the results of a test of the table lock. A C# stored procedure is adding 10 million records to a c-tree data file. The data file is a non-transaction file and has no index (to help isolate the effect of record locking).

The test was run twice: first with individual record locks and then with a table lock. In the chart below, the blue line indicates performance (records added per second) and the red line indicates memory use.

The chart below shows the results for the individual record lock case: the record insert performance (blue line) decreases over time, and memory use (red line) increases over time. And when the adds are finished and the commit happens, a noticeable time is spent freeing those locks. Insert time: **356 sec**.



The chart below shows the results for the table lock case: *the record insert performance (blue line) is higher than with individual record locks and stays at the same level of performance throughout*. The memory use (red line) initially increases by a small amount and then stays at that level. Insert time: **284 sec** *(20% faster)*.

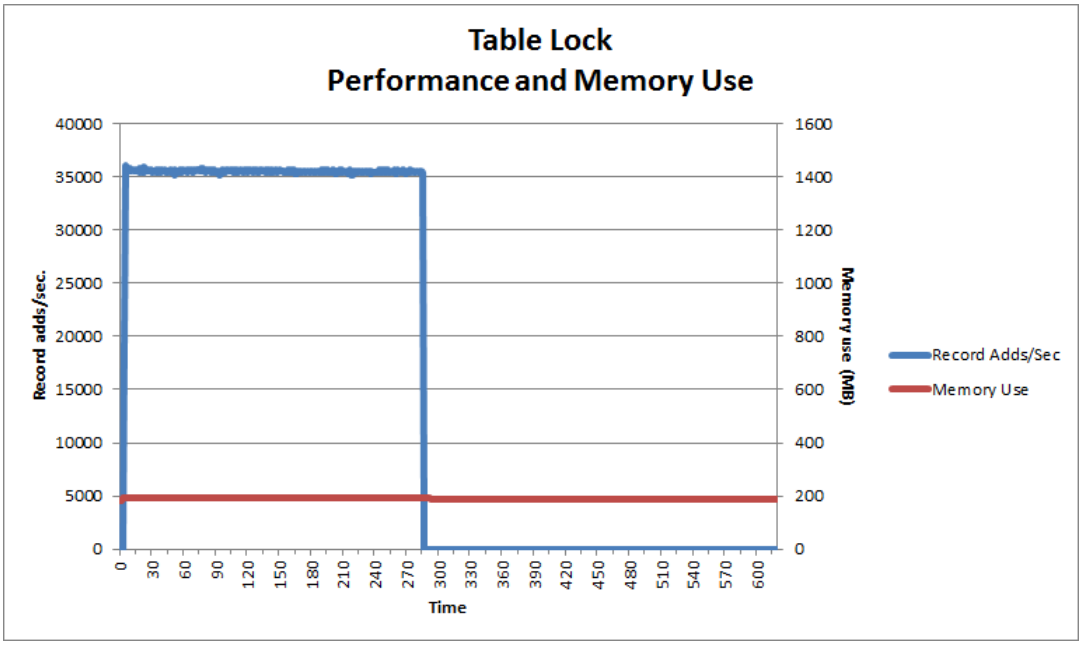


Table Locks with c-treeACE SQL

In applications where a large number of rows will be accessed for either reading or modifying, SQL provides an explicit locking construct for locking all the rows of a table. The `LOCK TABLE` statement explicitly locks a table in either `SHARE` or `EXCLUSIVE` mode.

The following example shows how to acquire a lock in the `EXCLUSIVE` mode for a table called `customer` from an ESQL command line:

```
EXEC SQL
     LOCK TABLE customer IN EXCLUSIVE MODE;
```

The above statement will prevent other transactions from either reading or modifying the table customer until the transaction either commits or performs a rollback.

The following example shows acquiring lock in the `SHARE` mode for a table called `orders`:

```
EXEC SQL
     LOCK TABLE orders IN SHARE MODE;
```

The above statement will prevent other transactions from modifying the `orders` table until the transaction either commits or performs a rollback.

Table Locks with c-tree API Functions

To acquire a file (table) lock using c-treeACE, call **LOKREC()** (https://docs.faircom.com/doc**/ctreeplus/lockctdata.htm**) with a new *ctLOCK_FILE* mode as follows:

```
LOKREC(datno, ctLOCK_FILE, lockmode);
```

See **Table Lock Mode for LOKREC** for details.

Table Lock Behavior with Transactions

If a table write lock is in effect when a record is updated in a transaction, the table write lock cannot be removed until the transaction commits or aborts. Note that the call to free the table write lock returns success (NO_ERROR) and *sysiocod* is set to **UDLK_TRN (-3)** in this situation, indicating that the table write lock was not released.

8.7 IPv6 Support

Internet Protocol (IP) IPv6 format is available for V11 and later c-treeACE servers.

IPv6 greatly expands the number of available IP addresses over the previous IPv4 standard, which used the familiar quad dotted format (four decimal numbers, from 0 through 255, separated by dots), allowing 4.3 billion addresses. IPv6 uses eight groups of four hex digits each, for 2^{128} addresses. Although proposed in 1998, only a small percentage of machines are currently using IPv6. However, IPv6 adoption is quickly expected as the number of IPv4 addresses is at, or very near, exhaustion, especially for external internet addressing. Although, IPv4 remains efficient for internal networks.

- IPv6 support is not available for QNX and SCO.

Server Configuration Options

This support is affected by placing the following configuration option in *ctsrvr.cfg*:

- `COMM_PROTOCOL F_TCPIPV6` - Starts a new listener thread on an IPv6 socket.

To support both IPv4 and IPv6 in the same c-treeACE Server, list the following keywords:

- `COMM_PROTOCOL F_TCPIP`
- `COMM_PROTOCOL F_TCPIPV6`

Server keyword `SQL_OPTION NO_IPV6` can be added to *ctsrvr.cfg* to accept only IPv4 connections. This is not generally recommended and is more likely to cause connection issues than to solve them.

Note: The `COMM_PROTOCOL F_TCPIPV6` keyword is commented out in the default *ctsrvr.cfg* file. Be sure to remove the comment symbol (the semicolon - `;`) before trying to use this new support.

An environment variable can be used for native clients to request only IPv4 address: *CTSQL_IPV4_ONLY*. Setting this variable to any value in the environment will effectively disable IPv6 connection attempts from the client. This may be needed on networks where both IPv4 and IPv6 are enabled, but the c-tree SQL server does not accept IPv6 connections.

Native SQL clients on Windows will attempt to connect to the first address (IPv4 or IPv6) resolved for the host.

Notes:

Server keyword `SQL_OPTION NO_IPV6` can be added to *ctsrvr.cfg* to accept only IPv4 connections. This is not generally recommended and is more likely to cause connection issues than to solve them.

Native SQL clients on Windows will attempt to connect to the first address (IPv4 or IPv6) resolved for the host.

Java and ADO.NET SQL clients currently support only IPv4. Explicit IPv6 addresses in client connection strings are not currently supported.

Default Protocol Override

An application can override the default protocol by specifying the protocol in the connection string with the following syntax:

```
FAIRCOMS@myhost^TCPIP
```

or

```
FAIRCOMS@myhost^TCPIPV6
```

Building IPV6 Client Applications

By default, c-treeACE libraries and utilities are built with IPV4 support only. For IPV6 support, build your own c-treeACE libraries and utilities using **mtmake.exe** (**mtmake** on Linux) or **mtpro.exe** (the Windows-only GUI version) located in the "*pro*" directory (by default: *\FairCom\V*\win32\pro*). On the client side, when IPv6 is selected in **mtmake**, both V4 and V6 stacks are included.

Note: As IPv6 adoption grows, default mtclient libraries will likely be enabled with this support in the near future.

If a numeric address is used, or the protocol is specified with the `^` syntax, we use the desired protocol. Otherwise we first attempt an IPv6 name resolution and connection. If that fails we attempt an IPv4 name resolution and connection.

Note: On Windows, IPv6 requires *winsock2.h* and linking with *ws2_32.lib*. For applications, this will mean compilation errors if *windows.h* is included before *ctreep.h* or *winsock2.h*.

GetServerBroadcast Function

IPv6-enabled clients can use the new function:

```
NINT GetServerBroadcast(pTEXT buffer, NINT bufsiz, UCOUNT port, LONG sec, NINT protocol)
```

This function provides functionality equivalent to the IPv4-only function **GetServerInfoXtd()**.

The protocol must be either of the constants `lcyNTREE_TCP` or `lcyNTREE_TCPIPV6`.

Note: With node-based licensing, if a single client machine (besides a local connection) connects to the server using both IPv4 and IPv6 addresses, it will count as 2 nodes.

8.8 Transaction Restore Points for Application Recovery

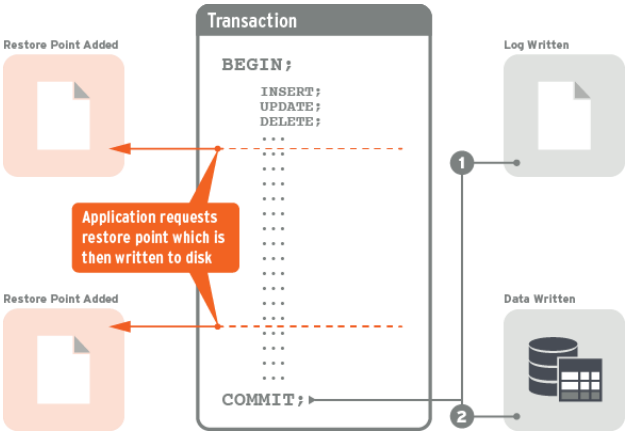
The feature from c-treeACE V11 gives you greater control in balancing data integrity against performance and recovery. A Restore Point marks a position in a transaction log to which FairCom Server's automatic recovery can roll back the system in event of system failure.

Transaction Restore Points

The new Delayed Durability feature can greatly enhance performance; however, should a system fail, database recovery could take considerable time if the transaction logs held substantial volumes of data not yet flushed to data and index files.

To this end, a new Restore Point feature has been introduced. A Restore Point is a position in a transaction log to which FairCom Server's automatic recovery can roll back the system in event of system failure.

The Restore Point is used for restoring a system to a point in time coordinated with the application as a known good state. After restoring the system to a Restore Point, the application can be designed to perform work that was not recovered. For example, those applications maintaining their own journal of operations. By coordinating a known good state in time between the application and the server's transaction logs, these applications can quickly recover back to a point in time when absolutely needed.



For more information, see the **Coordinate Application Recovery with Transaction Restore Points /doc/v11ace/69097.htm** in the *c-treeACE V11 Update Guide*.

8.9 Support for authentication files created with ctcmdset

This build introduces support for authentication files in the FairCom RTG interface.

Notes:

It is now possible to set the FAIRAUTH environment variable to the path location of an authentication file created with the **ctcmdset** utility before any operations that connect to the server.

If the FAIRAUTH environment variable is not set, the connection attempt is made as user GUEST.

The **ctcmdset** utility is used to make the encrypted password file. The plain text form of the file should be:

```
; User Id
```

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

8.10 Support for retrieving locker ID in ACUCOBOL

This modification allows a COBOL programmer using ACUCOBOL with FairCom RTG to call the **ct_lockerid()** function to get the c-treeACE Server Task# of the client holding a lock. The change simply adds a new function (**ct_lockerid**) to the list of functions loaded during runtime initialization so is callable from the ACUCOBOL runtime. In addition, the ACUCOBOL runtime needs to be re-compiled with ACUCOBOL's list of available C functions updated to include the **ct_lockerid()** function in the source module *direct.c* as follows:

```
extern int ct_lockerid (void); /* add this line */
struct DIRECTTABLE LIBDIRECT[] = {
 { "LOCKERID", FUNC ct_lockerid, C_int }, /* add this line */
 { NULL,   NULL,  0 }
 };
```

The function can be called from the COBOL application as follows:

```
       working-storage section.
       77  lockerid  pic x(4) comp-5.
       ...
       if stat = "93"
           call "LOCKERID" giving lockerid
           display "file is locked by task# " lockerid
```

Four steps are required to take advantage of this ability (this assumes you have already included the FairCom FairCom RTG file handler in the RTG runtime - see the FairCom RTG manual for complete information):

1. Copy the new *ctreeacu.c* into the */lib* directory of your ACUCOBOL installation. A common location is */opt/acugt/lib*.
2. Edit the *direct.c* module as indicated above.
3. Execute the system make utility to compile new FairCom RTG functionality into your ACUCOBOL *runcbl* runtime module. (This requires a compiler installed and available on your system.)
4. Copy the new *runcbl* into the */bin* directory of your ACUCOBOL installation.

> **Note**: *Be sure to back up existing* ***runcbl*** *modules first.*

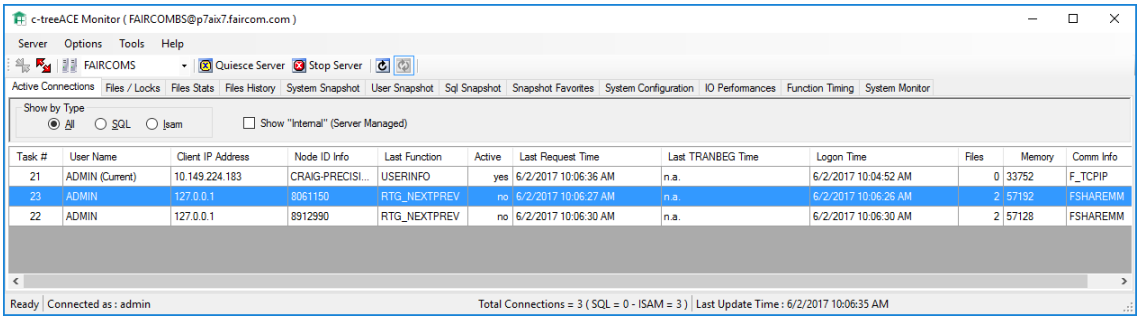
Once enabled, the “LOCKERID” function is available to any COBOL application using the newly compiled runtime. The Task ID is an internal thread ID per connection available within the FairCom RTG Server. *This ID is not directly associated with your operating system PIDs in any way.*

A task ID can be viewed via the FairCom RTG c-treeACE Monitor utility to identify exactly which runtime is holding a lock. If that connection is associated with other identifying information, such as their process ID (PID), an administrator can quickly determine who is holding a lock.

For example, consider with the following LOCKERID error output:

```
File is locked by task# :        23
ERROR: [9/057]
*** Execution aborted ***
Press <ENTER> key to exit...
```

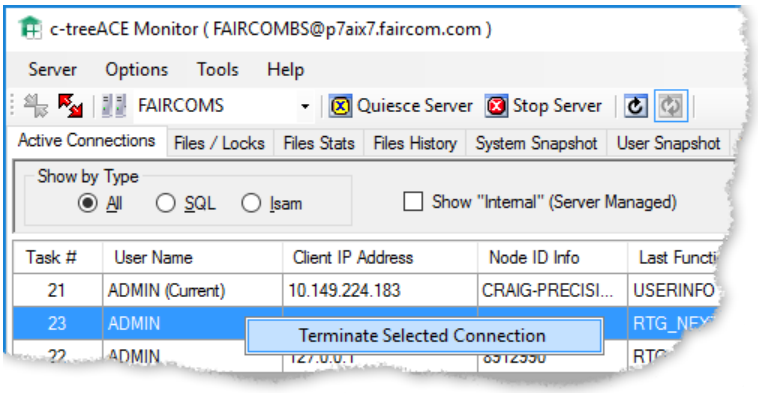
A Task ID can be identified using c-treeACE Monitor as shown in the screenshot below:



If *ctree.conf* specifies a PID to be passed in for each runtime user, an actual user can be determined. Any identifying information can be included in the name attribute, including IP addresses, local machine names, etc., from users' environment variables. In this case, inside the instance tag, include the name attribute with %p for the PID:

```
<instance  user="ADMIN" password="ADMIN"  server="BSTREET1"  name="%p">
...
</instance>
```

In fact, if absolutely necessary, an administrator can right-click on a connection in c-treeACE Monitor and kill that connection. *This action should be very carefully considered, however.*



8.11 Delete-flag support for <ctfixed> files to resolve c-tree errors 553 & 21

The delete-flag allows adding records that begin with 0xFF and have a length less than 9 bytes (5 bytes for non-huge files).

This modification introduces support for an internal record delete flag for FairCom RTG files created with the `<ctfixed>` option enabled. The delete-flag support allows a record beginning with 0xFF to be inserted in a *ctFIXED* file (a c-tree error **553** was returned before this change).

It also allows creating a *ctFIXED* huge file with less than a 9-byte record length or a *ctFIXED* non-huge file with less than a 5-byte record length (a c-tree error **21** was returned before this change).

9. Notable Compatibility Changes

This section lists important known compatibility changes from prior releases.

It is important to review these issues and ensure that your application continues to correctly function after upgrading to this latest release, V2.

9.1 Improved updates during scans on duplicate index now only updates record once

Unexpected behavior was observed when FairCom RTG scanned a file with an index that allowed duplicates: If a record was changed without updating the key used for scanning, it was possible to encounter the record again during the scanning phase. This caused the same records to be updated multiple times. The logic has been modified to correct this behavior.

9.2 Improved isCOBOL compatibility for UNLOCK operation

FairCom RTG has been modified to address an unexpected behavior when used with isCOBOL:

In an isCOBOL application the UNLOCK operation releases locks when it is called within an active transaction, while the expected behavior would be to not release any lock.

The logic has been modified to conform to this behavior.

9.3 ctutil returns syntax error if configuration file does not exist

The **ctutil** utility allows a configuration file to be specified with option *-c*. If the specified file does not exist, an error message is returned. The logic that opens the configuration file has been modified to return a more specific error message if the configuration file does not exist. The old message was:

```
Configuration syntax error. syntax error near - line 1, column 1
```

The new message is:

```
Configuration file does not exist
```

9.4 Error 12 creating a file when iscobol.file.index.data_suffix= is set to a space

Creating a file with no extension by setting the *iscobol.properties* setting *iscobol.file.index.data_suffix*= to a space, failed with error **12**. The problem occurred only when using the *iscobol.properties* setting and not when using the XML *ctree.conf* file. The logic has been modified to correct this.

9.5 Error 26 (FACS_ERR) mapped to BTRV error B_FILE_NOT_OPEN

FairCom RTG Error 26, **FACS_ERR**, is now mapped to the BTRV error 3, **B_FILE_NOT_OPEN**.

9.6 Error 9D:160 during READ NEXT/PREV

An unexpected error **9D:160** occurred if a DELETE operation removed the record being returned by a READ NEXT following a START operation. The logic has been modified to correct this error.

9.7 Improved error mapping in BTRV interface

A number of internal FairCom RTG errors that were mapped in the BTRV interface to the generic errors **2** (I/O error) and **19** (unrecoverable error) are now mapped to more precise error codes.

9.8 Return appropriate BTRV error in case of dead lock

FairCom RTG BTRV has been modified to return the appropriate BTRV errors in these situations:

- It returns BTRV error **78** in case of a dead lock.
- It returns BTRV error **130** when there is not enough memory to allocate lock tables.

9.9 Return error when unsupported key flag is passed from BTRV

Enhancements to FairCom RTG BTRV provide support for more BTRV key flags. A "not supported" error is returned if the key flag passed from the application is not supported.

The following key flags are currently supported:

- `DUP`
- `MOD`
- `SEG`
- `ALT`
- `REPEAT_DUPS_KEY`
- `EXTTYPE_KEY`

Support will be added for:

- `NUL`
- `NUMBERED_ACS`
- `NAMED_ACS`
- `DESC_KEY`
- `NOCASE_KEY`

9.10 Unexpected 4113 error (CTDBRET_CALLBACK_5)

Error 4113 (**CTDBRET_CALLBACK_5**) was observed during a FairCom RTG **ctutil** *-sqlize* operation with a XDD having more than 2500 fields. The logic has been changed to correct this.

Note: The error code returned when a blob fields does not specify the field containing its size has been changed from **CTDBRET_CALLBACK_5** to **CTDBRET_CALLBACK_11**. This represents a modification in behavior.

9.11 Unexpected 9D, 160 error during READ NEXT/PREV and REWRITE/DELETE operations

The following sequence of operations causes an error **9D, 160** (in IsCobol):

```
START - DELETE - READ NEXT
```

The START operation finds the record position of a given key but when the READ NEXT attempts to read the record, an error is returned because the record was deleted in the meantime.

Internally (in the driver), the error is expected in this situation but the COBOL user does not expect to see any error. There are other scenarios when the c-tree error 160 (**ITIM_ERR**) could legitimately occur but are not symptoms of an index problem or of multi-user interference caused by the application and, therefore, should be handled by the driver. The driver has been enhanced to handle and solve the error situation.

9.12 Behavior Change: Default <instance> attributes for isCOBOL

The default for the isCOBOL driver has been changed. The `<instance>` attribute's default has been changed to align to other COBOL runtimes:

- It does not check that the versions of the client and server are the same (*versioncheck="no"*)
- It does not connect when runtime starts, but it connects at first open (*connect="no"*).

10. Introducing Version 2 of FairCom RTG BTRV

FairCom RTG BTRV has undergone substantial improvements in this release to become a more mature product. From SQL support to expanded support for the BTRV API to a more powerful engine, V2 delivers. This section lists important changes that make FairCom RTG BTRV an all new product.

10.1 New: FairCom RTG BTRV Now Supports SQL

Version 2 of FairCom RTG BTRV now brings SQL to your Btrieve applications. The data stored by your existing Btrieve applications is now available to the multitude of tools that can connect using SQL and other standard relational interfaces.

In this release, FairCom RTG opens your applications to the world of possibilities that come with SQL access. FairCom RTG BTRV makes your data available to the industry's latest business intelligence (BI), analytical, and reporting tools.

SQL Schema Extractor Tool

Btrieve DDF dictionaries contain the file schema information necessary for sqlizing. The definitions are contained in three separate files: files; fields; indexes. This new tool reads those files and extracts the information to create SQL schemas.

-ddf2xdd

Transforms Btrieve Data Dictionary Files (DDFs) into c-tree XDD files, which contain schema information about the data and index structures.

You must first generate *.sav* files from *file.ddf*, *field.ddf*, and *index.dff* using the **btutil** command as follows:

```
butil -save FILE.DDF file.sav
butil -save FIELD.DDF field.sav
butil -save INDEX.DDF index.sav
```

You are now ready to use **ctutil** to create the XDD files, as shown in **Usage**.

Usage

```
ctutil -ddf2xdd DDF_directory [-rule=rules_file] [-tablename]
```

Where:

- *DDF_directory* is the name of the directory that contains the `.sav` files you created using **butil**.
- *-rule=rules_file* is an optional path to a rules file.
- *-tablename* (optional) uses the table name instead of the file name to determine the output file names.

  When exporting DDF files, a naming problem can occur if multiple tables in different directories have the same physical data file name. The *-tablename* parameter avoids naming conflicts by using table names instead of DDF file physical names to determine the XDD output file names.

One XDD file will be created for each file contained in *file.sav*. These files will be saved in *DDF_directory*.

Sqlize BTRV files

FairCom RTG BTRV provides the facilities you need to access data through SQL:

- data-type mappings between Btrieve and c-tree
- data-type conversion routines

10.2 Expanded, More Comprehensive BTRV API

The first release of FairCom RTG BTRV focused on the most frequently used Btrieve features. FairCom RTG BTRV V2 expands support for Btrieve features such as op-codes, extended operations, collating sequences, on-the-fly CREATE/DROP indices, and metadata dictionaries support (DDF) etc.

- **ACS support** (page 100)
- **UNLOCK operation** (page 100)
- **INSERT EXTENDED function** (page 100)
- **CREATE/DROP INDEX** (page 100)
- **BTRV function ordinal settings** (page 100)
- **Adding an index in shared mode** (page 101)

BTRV Extended Index Types

Support has been added to FairCom RTG BTRV for extended key types on the following BTRV types:

- STRING
- INTEGER
- IEEE
- DATE
- TIME
- DECIMAL
- MONEY
- NUMERIC
- ZSTRING
- UNSIGNED_BINARY
- AUTOINCREMENT
- BIT
- NUMERICSTS
- NUMERICSA
- CURRENCY
- NUMERICSLB
- NUMERICSLS
- NUMERICSTB
- NULL_INDICATOR

BTRV transactional type column = c-tree ISAM engine.

Relational type = c-tree SQL engine.

BTRV Exclusive transaction = Means only one user. c-tree doesn't have exclusive transaction; similar to single user.

ACS support

Prior to release V2, FairCom RTG BTRV supported only one ACS (alternate collating sequence) key flag. Originally, only the ALT key flag was supported. With this new feature, the ACS can be set to either ALT or NUMBERED_ACS as long as only one ACS is used (acsNumber = 0). The NAMED_ACS key flag is currently not supported.

UNLOCK operation

UNLOCK (opcode 27) operations are now supported in FairCom RTG BTRV interfaces.

INSERT EXTENDED function

Version 2 of FairCom RTG BTRV introduces support for the BTRV **INSERT EXTENDED** (BTRV opcode 40) function. The implementation follows all the BTRV specifications including returning the 4-byte record position of successfully inserted records.

This feature provides functionality beyond the specifications: The file can be opened with keyBuffer bias -120, which instructs FairCom RTG BTRV to return 8-byte record positions. In this case, the **INSERT EXTENDED** function returns the 8-byte record positions instead of 4-byte.

AUTOINC fields

In release V2 of FairCom RTG BTRV, support has been added for BTRV AUTOINC field. This feature provides support for fields that automatically increment.

CREATE/DROP INDEX

Release V2 of FairCom RTG BTRV implements CREATE INDEX operations to create an index on the fly.

BTRV function ordinal settings

When linking a FairCom RTG BTRV library with a Btrieve application, some obsolete Btrieve functions were not exported, which results in an unresolved external. These old functions are now exported to avoid linking problems.

Before this modification, FairCom RTG BTRV exported the following:

```
BTRCALL         @1
BTRCALLID       @2
WBTRVSTOP       @6
```

This change adds the following to the list of exported functions:

```
WBRQSHELLINIT  @3
WBSHELLINIT    @4
WBTRVINIT      @5
WBTRVIDSTOP    @7
DBUGetInfo     @9
DBUSetInfo    @10
```

Support for BTRV Create Index Operation

This release introduces support for creating new indexes on existing files (op code 31) in FairCom RTG BTRV.

Support for BTRV Add Index and Extended Index modes

Support has been enabled to FairCom RTG BTRV for for Add Index and Extended Index modes.

Adding an index to a data file open in shared mode

Beginning in FairCom RTG BTRV V2, it is possible to add an index to a data file that is open in shared mode if the data file is only open once by the calling connection. This support improves the integration between FairCom RTG BTRV and Btrieve applications by more closely matching Btrieve behavior. This feature changes the behavior of the **PRMIIDX()** and **RBLIIDX()** functions.

When adding an index to a file in shared mode, the data file is switched to exclusive mode, the index is added, then the data file is switched back to shared mode.

If adding an index member to a host index that is open in shared mode, we also switch the host index to exclusive mode, then switch it back to shared mode when we are done adding the index member.

Support for Exclusive Transactions in FairCom RTG BTRV

FairCom introduces exclusive transaction support required for better compliance with the Btrieve API. A file accessed within an exclusive transaction is now locked for the entire duration of the transaction.

10.3 Easy Migration to FairCom RTG BTRV

ctmigra utility to migrate data using BTRV interface

A new FairCom RTG utility, **ctmigra**, has been introduced to aid in migrating data. This utility copies records using a read/write loop. The ExtFH and BTRV interfaces are supported.

The utility uses the FairCom RTG Switcher to redirect calls to different libraries so as to read/write from/to any database that uses the supported interface. The redirection is defined through `<redirinstance>` elements of the *ctree.conf* configuration file (see the example below).

This modification also includes changes to *CTBTRV.C* to work with the **ctmigra** utility.

Usage of FairCom RTG **ctmigra** depends on your native data file types and your platform.

Usage

```
ctmigra btrv|extfh [OPTIONS] SOURCE DEST
```

where:

- *SOURCE* - name of the file to be copied.
- *DEST* - name of the file to which it will be copied.

Options for FairCom RTG Files

- *-n SERVERNAME* or *--dest-server=SERVERNAME* where *SERVERNAME* is the name of the destination c-treeACE Server
- *-u USERID* or *--dest-user=USERID* where *USERID* is the user name of the destination c-treeACE Server
- *-p PASSWORD* or *--dest-password=PASSWORD* where *PASSWORD* is the user password of the destination c-treeACE Server
- *-I STRING* or *--dest-idx-suffix=STRING* where *STRING* is the suffix of destination index file name
- *-N SERVERNAME* or *--source-server=SERVERNAME* where *SERVERNAME* is the name of the source c-treeACE Server
- *-U USERID* or *--source-user=USERID* where *USERID* is the user name of the source c-treeACE Server
- *-P PASSWORD* or *--source-password=PASSWORD* where *PASSWORD* is the user password of the source c-treeACE Server
- *-i STRING* or *--source-idx-suffix=STRING* where *STRING* is the suffix of source index file name
- *-l FILE* or *--log=FILE* where *FILE* is the name of the file to log additional information. To redirect log to stderr use - as *FILE*
- *-b RECORDS* or *--batch-size=RECORDS* where *RECORDS* is the number of records to read / write in batches
- *-a* (*--append-ext*) appends index extension instead of replacing data extension.
- *-o OWNER* (*--owner=OWNER*) specifies the BTRV file owner (can be used if you encountered an error **51** when attempting to migrate).

- *-r* (*--replace*) instructs **ctmigra** to replace the existing destination file (V11 and later).
- *-e*, *--encrypt=CIPHER* - Encrypt destination file with CIPHER algorithm.
- *-z*, *--datacompress=TYPE[;LEV][;STR]* - Compress destination data file with TYPE algorithm.
- *-t*, *--transaction=no|yes|logging* - Create destination data file with or without transaction support.

  The last three options enable the following configuration options on the destination file (corresponding to *-e*, *-z*, and *-t* respectively):

```
<encrypt type="CIPHER">
<datacompress type="TYPE"= level="LEV" strategy="STR">
<transaction logging="no|yes">
```

See usage examples in the next topics. Notice that the *Micro Focus COBOL Migration Example* can be useful for Btrieve users.

External Library Configuration for Native File Access

For certain data types, such as BTRV, original external libraries are required to access native data formats.

- *-s LIBRARY!FUNCTION* or *--source-lib=LIBRARY!FUNCTION* where *LIBRARY* is the external dynamically loadable library and *FUNCTION* is the interface entry-point function to handle the source file
- *-d LIBRARY!FUNCTION* or *--dest-lib=LIBRARY!FUNCTION* where *LIBRARY* is the external dynamically loadable library and *FUNCTION* is the interface entry-point function to handle the destination file

Note: Current usage options are always available when no command-line options are supplied.

Examples

If you are using FairCom RTG COBOL:

```
ctmigra.exe extfh -s MFFH.DLL -n FAIRCOMS@localhost -r C:\mydata\abc-btrv.dat abc-ctree.dat
```

If you are using FairCom RTG BTRV:

```
ctmigra.exe btrv -s wbtrv32.dll -n FAIRCOMS@localhost -r C:\mydata\abc-btrv.dat abc-ctree.dat
```

For more examples, see *Micro Focus COBOL Migration Example*.

ctmigra --quiet and --verbose options to select output information

These options for the **ctmigra** command-line utility suppress all output (*--quiet* or *-q*) or select the information to be sent to stdout (*--verbose=LEVEL* or *-v*). The *--verbose* parameter is a bitmask which can combine the following values:

1 - show message about final result

2 - show percentage progress

4 - show time spent in migration phases (read, write, finalize)

For example, to display everything, use *--verbose=7*, which is 1+2+4. If not specified, the *--verbose* level is *3* (1+2).

The *--quiet* option is identical to *--verbose=0*.

Alternative Usage

Most of the information you need for migration can be entered using the command-line parameters listed above. If you have additional considerations that dictate a more complex configuration, such as files that need to be treated specially, multiple clients, servers, etc., you can create a local configuration file (*ctree.conf*) for use during migration. (It can be edited as described in **Editing a Configuration File**.)

Note: The recommended best practice is to run **ctmigra** using the parameters listed above. If you must use the "alternative" approach described in this section, use the `bulkaddition` option in *ctree.conf*, which will greatly improve performance and it will optimize the index.

To use **ctmigra** with a local FairCom RTG configuration file, use the following command:

```
ctmigra btrv|extfh -c CONFIG_FILE SOURCE DEST
```

where:

- *SOURCE* - name of the file to be copied
- *DEST* - name of the file to which it will be copied
- *-c CONFIG_FILE* or *--config=CONFIG_FILE* where *CONFIG_FILE* is a valid FairCom RTG configuration file.

Be sure to include the `bulkaddition` option in *ctree.conf*.

Potential Errors and Troubleshooting

BTRV Error: -7	Your native source library is likely not available. Check the path to your native data file handling library and be sure it is specified with the *--source-lib=* option.
BTRV Error: 53	Attempted to open a non-BTRV file via a BTRV interface. This can possibly happen when you load a 32-bit BTRV DLL with a 64-bit version of the tool, or vice versa.

In FairCom RTG V2, error messages displayed by the **ctmigra** utility have been enhanced as follows:

- It now displays actual file names instead of $SOURCE$ and $DEST$.
- It now displays an error message if the source or destination c-tree Servers are not running (c-tree error 133).
- It now displays an error message if the user/password is not correct (c-tree error 450/451) or GUEST logon is disabled (c-tree error 470).
- It now displays an error message if c-tree Server is not valid due to OEM version incompatibility (c-tree error 530).
- It now displays an error message if the specified external library cannot be loaded.

Behavior Change: This modification changes the behavior of the **ctmigra** tool.

ctmigra option to specify BTRV owner

In FairCom RTG BTRV V2, a new **ctmigra** option was introduced to specify the BTRV file owner. The owner together with file name is passed to the Btrieve file system when opening the source file to migrate. This option can be used if you encountered an error **51** when attempting to migrate a Btrieve file to FairCom RTG. The option can be specified on the command line using either the single-dash or double-dash syntax:

```
-o OWNER
--owner=OWNER
```

This new option is also available in the RTG Migrate tool.

Support for converting Btrieve DDF into c-tree XDD

A switch has been added to **ctutil** to implement conversion from Btrieve's DDF files into c-tree XDD files, which contain schema information about the data and index structures. The format is as follows:

```
ctutil -ddf2xdd <dirname>
```

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

In the same directory, it creates one XDD file for each file contained in *file.sav*. These three files should be generated starting from *file.ddf*, *field.ddf*, *index.dff* using the **btutil** command as follows:

```
butil -save FILE.DDF file.sav
butil -save FIELD.DDF field.sav
butil -save INDEX.DDF index.sav
```

10.4 Advanced Features Make FairCom RTG BTRV More Powerful

Multi-thread support

In V2 and later, FairCom RTG BTRV introduces full support for multiple threads in the FairCom RTG BTRV interface. This interface now supports calling **BTRVID**/**BTRCALLID** with different *clientID* parameters within different threads.

Note: The stand-alone model of FairCom RTG BTRV achieves thread safety by serializing all calls.

Support for redirecting BTRV calls to other BTRV interfaces

In V2 and later, FairCom RTG introduces support for redirecting BTRV calls to other BTRV interfaces such as Pervasive Software / Actian Corporation Btrieve.

Use the `<redirinstance>` configuration element and specify the library and the entry-point function, which must have the prototype of the BTRCALLID function, using `<redirinstance lib="mylib" func="myfunc">` attributes.

The following is a configuration example to redirect to Pervasive Btrieve:

```
<redirinstance lib="C:\PVSW\Bin\WBTRV32.DLL" func="BTRCALLID">
 <file name="*.btr"/>
</redirinstance>
```

Support for reading extra file/key definitions from a resource

Some file/key definitions passed to FairCom RTG are not mapped to c-tree definitions because c-tree does not have their relative definition. However this information must persist because it must be returned back when requested. Release V2 adds support for reading the new resource types.

In addition this change adds support for Btrieve Key Flag ALT (Default Alternate Collating Sequence) and EXTTYPE_KEY (Extended Data Types).

It also adds support for persisting Page Size and Preallocated Pages values.

FPUTFGET Library

A multi-user standalone DLL / shared library (FPUTFGET) is now included with FairCom RTG BTRV. This mode supports the **Standalone Multi-User Model /doc/ctreeplus/30177.htm**, described in the *c-treeACE Programmer's Reference Guide*. This library exports the **BTRCALL** and **BTRCALLID** functions. It is located in the following directory:

FairCom\Vx.x.x.RTG\win32\Driver\ctree.btrv\ctreestd.dll

Support for returning 8-byte record position

In V2 and later, the B_GET_POSITION operation returns 8-byte record positions instead of the 4-byte record number used by Btrieve. The operations that accept a record position as input (currently B_GET_DIRECT and B_UNLOCK) now support 8-byte record positions as input.

By default the BTRV interface returns and expects 4-byte record positions, returning error **B_DATA_MESSAGE_TOO_SMALL** (97) if the record position is over 4GB.

To instruct the BTRV interface to return and expect 8-byte record positions, open the file using a -120 bias in *keyNum* parameter when requesting a B_OPEN operation. For example, to open a file exclusively, pass -124 instead of -4 in the *keyNum* parameter.

10.5 Improved Error Reporting

Improved error mapping in BTRV interface

A number of internal FairCom RTG errors that were mapped in the BTRV interface to the generic errors **2** (I/O error) and **19** (unrecoverable error) are now mapped to more precise error codes.

Return appropriate BTRV error in case of dead lock

FairCom RTG BTRV has been modified to return the appropriate BTRV errors in these situations:

- It returns BTRV error **78** in case of a dead lock.
- It returns BTRV error **130** when there is not enough memory to allocate lock tables.

Return error when unsupported key flag is passed from BTRV

Enhancements to FairCom RTG BTRV provide support for more BTRV key flags. A "not supported" error is returned if the key flag passed from the application is not supported.

The following key flags are currently supported:

- `DUP`
- `MOD`
- `SEG`
- `ALT`
- `REPEAT_DUPS_KEY`
- `EXTTYPE_KEY`

Support will be added for:

- `NUL`
- `NUMBERED_ACS`
- `NAMED_ACS`
- `DESC_KEY`
- `NOCASE_KEY`

10.6 BTRV Tutorial

A FairCom RTG BTRV tutorial guides you through the process of creating an application that takes advantage of the powerful FairCom RTG environment.

Initialize()

Define()

Manage()

Done()

Note: This tutorial explains how to program an application that uses c-treeACE. These procedures are NOT necessary for using FairCom RTG BTRV, which simply involves installing, migrating, and configuring as explained in the **FairCom RTG BTRV Users Guide** (https://docs.faircom.com/doc**/ctbtrv/**).

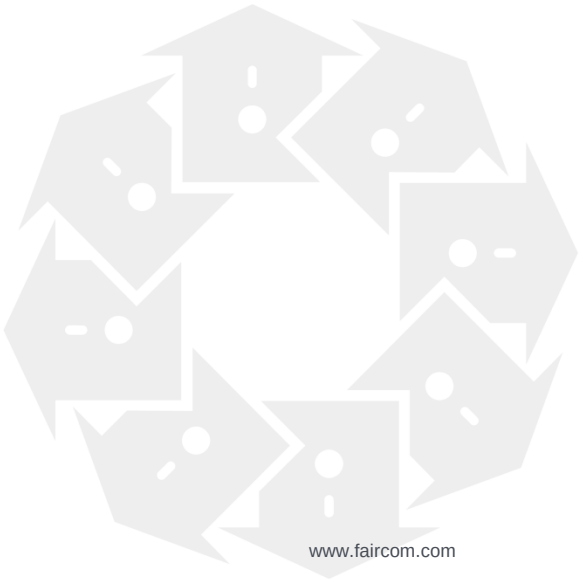
This tutorial is documented in the **FairCom RTG BTRV ReadMe**.

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()	c-treeACE Function name
Parameter	c-treeACE Function Parameter
`Code Example`	Code example or Command line usage
utility	c-treeACE executable or utility
filename	c-treeACE file or path name
`CONFIGURATION KEYWORD`	c-treeACE Configuration Keyword
CTREE_ERR	c-treeACE Error Code



Copyright Notice

Trademarks

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