

FairCom

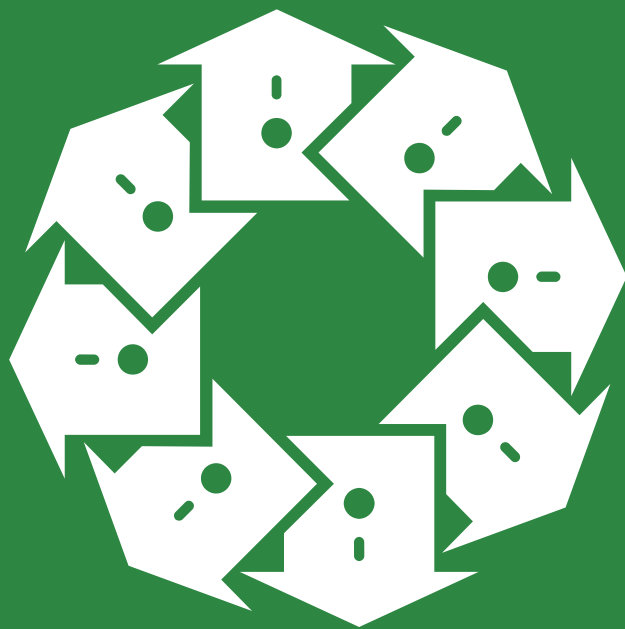
RTG

v5

Data Made Simple



FairCom



FairCom

Table of Contents

2	Welcome to RTG
4	Replication & High Availability
6	Unified Data Types—ACE and RTG
7	Partitioned Files
8	Hot Alter Table
10	JSON DB API
12	New Drivers & Tutorials for JSON DB API
14	FairCom MQ Broker
15	DB Notify
17	JavaScript for DB Notify
18	Web Apps
24	Dynamic Data and Index Optimization
25	Cache Enhancements
26	Security
29	Additional Features
36	More RTG Enhancements
40	Additional RTG Progress
43	SQL Improvements
44	Btrieve Compatibility



Welcome to FairCom RTG v5
Data Made Simple

As we enter our 45th year of developing high-speed data management solutions, FairCom is proud to bring you a significant evolution of the FairCom RTG product line. We've added many new features and enhancements to RTG. A few of the most prominent include:

- Server Clusters for High Availability and Disaster Recovery solutions
- FairCom DB Notify and FairCom MQ provide no-code change data capture (CDC) support for RTG*
- Unified Data Types between FairCom DB and RTG
- Alter Table & Hot Alter Table
- Partitioned Files for COBOL data files
- A completely new API, JSON DB, to access COBOL data via JSON commands and documents

**Note: DB Notify and FairCom MQ require additional licensing.*

In addition, FairCom has renamed c-treeRTG to FairCom RTG to better reflect the new Unified Data Type compatibility between FairCom DB and FairCom RTG and to reinforce interoperability across our product lines.

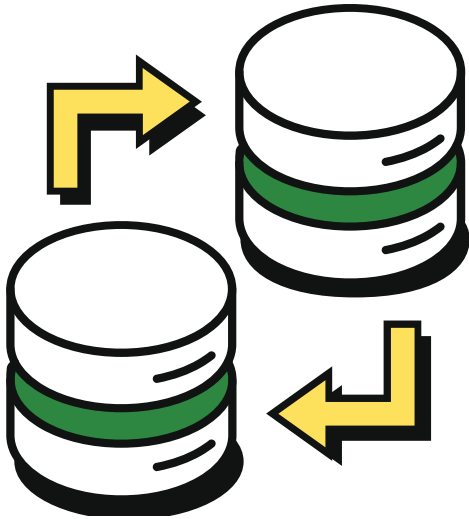
Many of the features mentioned in this booklet come from advancements in FairCom DB v13. These features have all been verified to work properly with RTG v5. Please note that in some cases, the new Unified Data Type support will be required, and this is noted accordingly.

FairCom's support team is available to answer questions and demonstrate this new technology.

Email support@faircom.com or call us at +USA 573-445-6833 to schedule a time to speak with a FairCom engineer.

This book highlights FairCom RTG v5's new features and includes references to our documentation for more details where applicable.

Replication and High Availability



Failover and high availability are critical to successful enterprise applications.

With this in mind, FairCom RTG v5 supports Linux Pacemaker clusters. Now, you can combine replication with

high-availability solutions to scale your app and create disaster recovery servers.

By setting the new RTG v5 <instance> configuration option [failovertimeout](#), you can control how much time passes before dropping a failed instance and connecting to the secondary instance defined by Pacemaker.

You can also create replication plans involving thousands of tables with optimal system resource usage. Create groups of files and manage their replication as a single unit, and use the new Replication Agent Manager to manage, start, and stop individual replication agents. These features are great for scaling and high availability in large, multitenant solutions.

RTG v5 also offers many performance gains and stability improvements, such as parallel replication, which speeds up the replication data apply process.

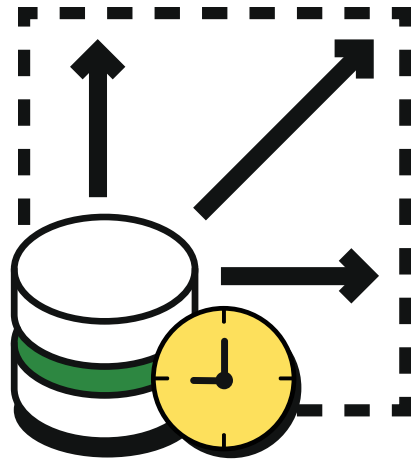
Managing replication is also simpler than ever with the new replication command line utility ([repadm](#)). This utility makes it easy to begin replication between FairCom servers. It also simplifies script replication setup and management. The improved Replication Manager configures, starts, stops, monitors, and troubleshoots replication processes.

Replication supports many [use cases](#)

Replicate data synchronously to create [highly available solutions](#).

Replicate data asynchronously to create [read-only servers](#) and [disaster recovery solutions](#).

Replicate data bidirectionally and asynchronously to create horizontally scaled, [eventually consistent solutions](#).



Unified Data Types—ACE and RTG

FairCom RTG v5 closes a compatibility gap between FairCom DB (previously known as c-treeACE) and FairCom RTG (previously known as c-treeRTG).

RTG v5 introduces a new [Unified Data Type](#) by standardizing the internal data file header to contain both the COBOL- and FairCom-type definitions of all fields. This means all applicable DB features are now available to RTG users.

For example, you can use Alter Table and Partitioned files after unifying the data file headers.

This update also makes DB files compatible with COBOL—without requiring an external XML file.

The COBOL field definitions in the internal data file header are what allow RTG to marshal your app's data to and from the disk. RTG supports the COBOL data types and file IO commands for major non-mainframe COBOL compiler

vendors, such as Varyant's isCOBOL, Micro Focus' EXTFH, Rocket Software's ACUCOBOL, etc.

In RTG v5, all newly created files default to the unified file format. You can still create new data files in the old format, but many features (like Alter Table and Partitioned Files) will be unavailable.

If you are new to RTG, this enhancement is unnoticeable, as you will only be working with files in the new format.

If you are an existing user, your files in the old format will continue to work. We encourage you to upgrade your files in order to take advantage of enhanced features today and in the future. Learn more in the RTG Documentation.

Partitioned Files

Files in the Unified Data Type format can use this feature with your existing COBOL/BTRV data.

[Partitioned files](#) are now supported for both the COBOL and BTRV editions of FairCom RTG. A partitioned file can efficiently and quickly purge or archive records. They are typically used to implement data retention policies and may slow query performance.

A partitioned file logically appears to be one data file but is physically a set of files whose contents are partitioned by the value of the partition key. Indexes on the partition file are also partitioned.

The partitioned file contains a host data file and its associated indexes combined with a partition rule. The host file does not contain actual data; rather, it's a logical file that appears to contain data.

The partition rule determines how to map the partition key value (e.g., a date, invoice number, or some other characteristic) to a physical partition member file and its indexes.

The server uses a partition rule to distribute records to the partition member files. The rule is typically based on time. For example, customer orders may be partitioned by calendar quarter. The server distributes all orders booked in the same quarter to the same partition member file. A member consists of a standard RTG data file and its associated indexes.

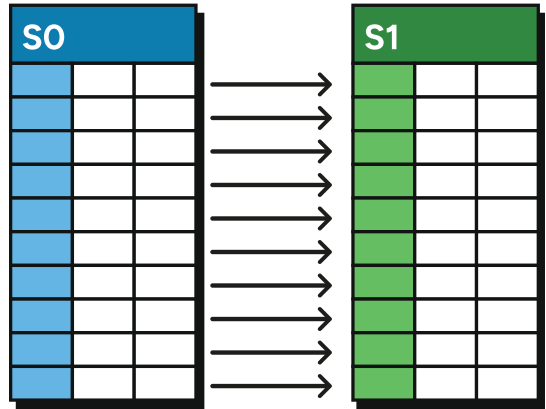
Use partitioned files to improve the scalability and availability of your data.

Hot Alter Table

Files in the new Unified Data Type format can use this feature with your existing COBOL/BTRV data.

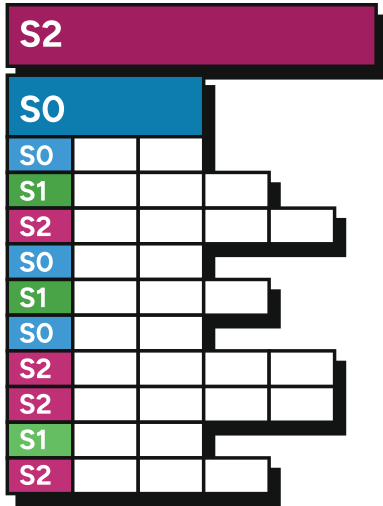
RTG v5 now supports both Alter Table and an enhanced [Hot Alter Table](#). This enhancement immediately changes a schema without rewriting all the records in the table, making modifying a table a very fast operation even when a data file contains billions of records.

All new data files created by RTG v5 support Hot Alter Table. You can also convert older data files to the unified format to support Hot Alter Table.



Multiple schemas

When using Hot Alter Table, a table can simultaneously contain records with different schemas. (A schema defines the data types and characteristics of each field in the record.) Hot Alter Table lets you modify the schema of a data file at any time. New records use the new field definitions, and existing records use the previous ones.



As the FairCom server reads or writes records, it upgrades records with old field definitions to the latest definitions.

Hot Alter Table adds a schema ID to each record's "header" (the record's metadata). The schema ID indicates which schema applies to the record (S0=the original schema, S1=the first altered schema, S2=the next altered

schema, and so forth.). The unified header in the data file contains the record converters necessary to convert records to each schema.

Convert records 'on the fly'

When the application reads a record with an older schema, the server converts it on the fly to the latest schema so the application always sees the latest version of the data. When an application updates a record, the server automatically promotes it to the current schema.

Limitations

You may not modify existing fields. See the *RTG Update Guide* for further details on using Alter Table.

JSON DB API

Files in the new Unified Data Type format can use this powerful API with your existing COBOL/BTRV data.

JSON Person

```
{
  "id": 1,
  "name": "Sam",
  "rank": 2,
  "email": {
    "name": "sam",
    "domain": "acme.com"
  }
}
```



Table Person

id	name	rank	email
1	"Sam"	2	{ "name": "sam", "domain": "acme.com" }

JSON command to create a person table

```
{
  "api": "db", "action": "createTable",
  "params": {
    "tableName": "person",
    "fields": [
      { "name": "name", "type": "varchar", "length": 50 },
      { "name": "rank", "type": "integer", "nullable": false },
      { "name": "email", "type": "json", "length": 65500 },
    ]
  }
}
```

For advanced RTG users, FairCom RTG v5 has a new API, [JSON DB](#), that lets you quickly build web applications and services on your ISAM files using any programming language, such as Java,

C#, JavaScript, Python, etc. You don't need a driver because JSON DB uses JSON and HTTP, which all programming languages support.

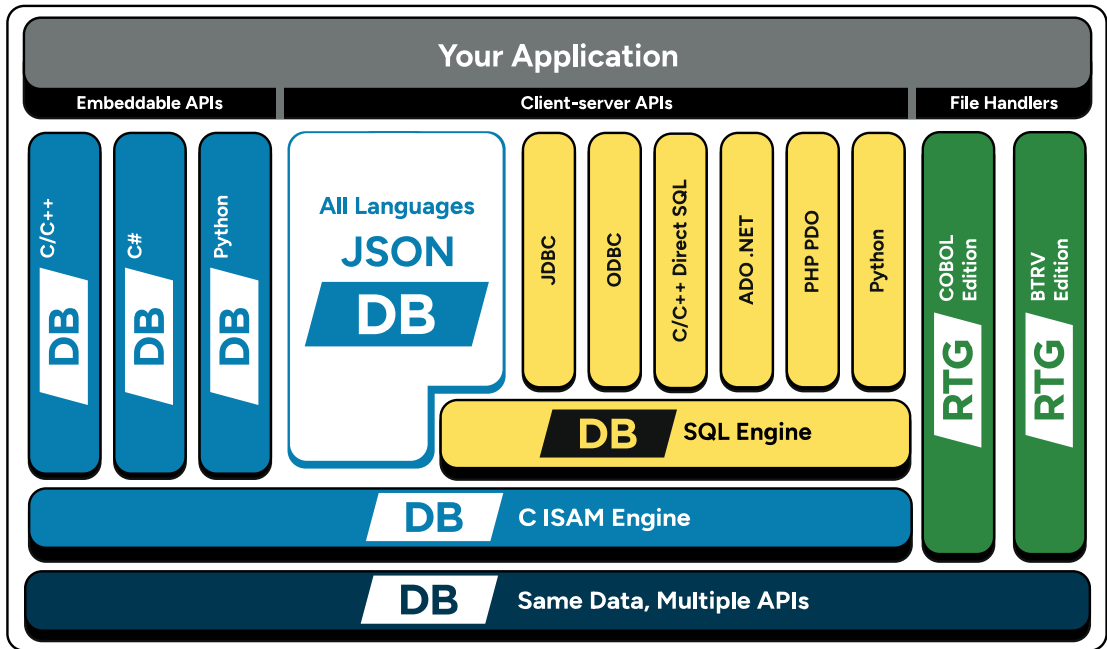
The JSON DB API uses simple JSON commands and automatically maps between ISAM and JSON. For example, you can use JSON commands to create tables, query data, run SQL statements, etc. You can query an ISAM record as a JSON document and save a JSON document into an ISAM record.

The JSON DB API manages all database operations:

- Create, alter, delete, list, and describe databases, tables, and indexes.
- Insert, update, and delete records using JSON.
- Run SQL queries and statements and return the results as JSON.
- Use specific JSON DB actions to query data with less code and faster results than SQL:
 - Look up records by key.
 - Tail the most recently inserted data.
- Retrieve all records as quickly as possible.
- Retrieve N records sorted ascending or descending.
- Retrieve sample data by skipping through records.
- Retrieve records matching a partial key.
- Retrieve records between two keys.
- Look up the closest matching record and retrieve its surrounding records.
- Use a cursor to return paginated data efficiently.
- Use a cursor to walk records forward or backward while skipping over data.

New Drivers and Tutorials for the JSON DB API

Files in the new Unified Data Type format can use these drivers with your existing COBOL/BTRV data.



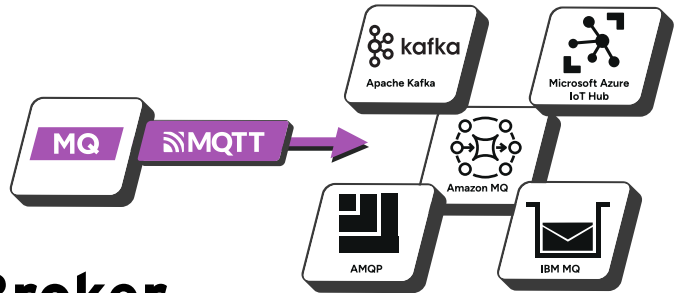
FairCom RTG v5 includes tutorials for using the JSON DB API with JSON commands over HTTP.

To help you get started, we've included JSON DB client libraries for the following languages:

- Java
- Node.js (JavaScript)
- Node-RED
- Python

You can find these drivers in the <faircom>/drivers folder:

- ▼ FairCom-RTG-Cobol.windows.64bit.v5.0.2.18
 - > config
 - > data
- ▼ drivers
 - > java.json.db
 - > nodejs.json.db
 - node-red.json.db
 - > python.json.db



FairCom MQ Broker

[FairCom MQ](#) is a new message queue product from FairCom that extracts, transforms, and queries data from our products, including FairCom RTG.

With the combination of [DB Notify](#)—a feature optimized to work with FairCom MQ to publish high-velocity messages with guaranteed delivery—customers can now track any changes to COBOL files and automatically convert them into JSON messages, which can then be consumed by other processes.

FairCom MQ is a high-speed, highly reliable, low-maintenance message broker with an embedded database. It supports all of the major MQTT

protocols, including MQTT 3.1, MQTT 3.1.1, and MQTT 5.

If you're unfamiliar with MQTT, know that it's a client-server messaging protocol for software to publish and subscribe to messages. Its key features are simplicity and ease of use.

We—along with many other companies—have chosen MQTT because it's easier to use and implement than other messaging systems, such as AMQP. For more details, see the *FairCom MQ* and the *DB Notify Concepts* documentation pages.



DB Notify for Unified Data Types

Easily switch from a 'pull' to a 'push' query system & automatically generate a native JSON version of your data—without code!

FairCom RTG can use the FairCom MQ message broker and DB Notify as a change data capture solution with MQTT topics. A topic is equivalent to a “queue” under the MQTT messaging protocol. This feature synchronizes data changes in RTG with external systems: For example, synchronizing microservice caches with a FairCom database.

DB Notify automatically captures data change events as JSON messages and uses MQTT to publish them to MQ. Any systems that subscribe to these messages via MQ receive guaranteed notification of data changes. DB Notify can send hundreds to thousands of data change events per second to an unlimited number of subscribers.

These data changes can be sent across the enterprise by sending to other messaging systems, including Kafka, IBM MQ, AMQP brokers (ActiveMQ, RabbitMQ, etc.), and MQTT brokers (FairCom MQ, HiveMQ, Mosquito). Data can also go to custom software

through simple MQTT drivers, which let you subscribe to MQTT messages. In addition, you can send data to cloud providers through their built-in brokers.

The best part of DB Notify? It doesn't require programming. Instead, you use simple JSON configuration files to specify which tables and records you want DB Notify to track.

Note: DB Notify requires a separate FairCom MQ license. Please contact FairCom for more information.

Example Use Cases:

- Microservices subscribing to data change events to keep their caches current. Each microservice can have a different subset of data.
- Notifying a service that a transaction has crossed a threshold.
- Replicating data to any SQL and NoSQL database. Pushing data changes to data lakes, data warehouses, and data marts.
- Sending alerts to security systems when sensitive data changes or when a data change violates security policies.
- Delivering data change events to SaaS applications in the cloud and on-prem ERP systems to process transactions and update master data.
- Updating massive numbers of mobile devices in real-time with changes to configuration, lookup data, metadata, etc.

JavaScript for DB Notify

Perform data transformations

As established in the DB Notify and FairCom MQ sections, DB Notify detects data change events in RTG and sends them to MQ. A data change event occurs when a record in a data file is inserted, updated, or deleted. MQ sends out the data changes as a JSON document containing the record's data fields before and after the change.

When an RTG data file has a schema, a FairCom MQ transform can use JavaScript to transform the data in the JSON data change messages.

Before the data change messages are published, JavaScript can make the following changes:

- Clean up data, such as removing extra spaces in strings.
- Transform values, such as converting Fahrenheit to Celsius, miles to kilometers, seconds to minutes, etc.
- Take a data record, update existing properties with new data, and add new properties.

See the [Code Package API](#) documentation for more details.

Web Apps

FairCom RTG v5 provides enhancements to three browser-based applications:

Data Explorer

Monitor

ISAM Explorer

FairCom / RTG



Data Explorer

Manage FairCom databases including data, tables, indexes, and objects. Also explore FairCom's JSON APIs.



Monitor

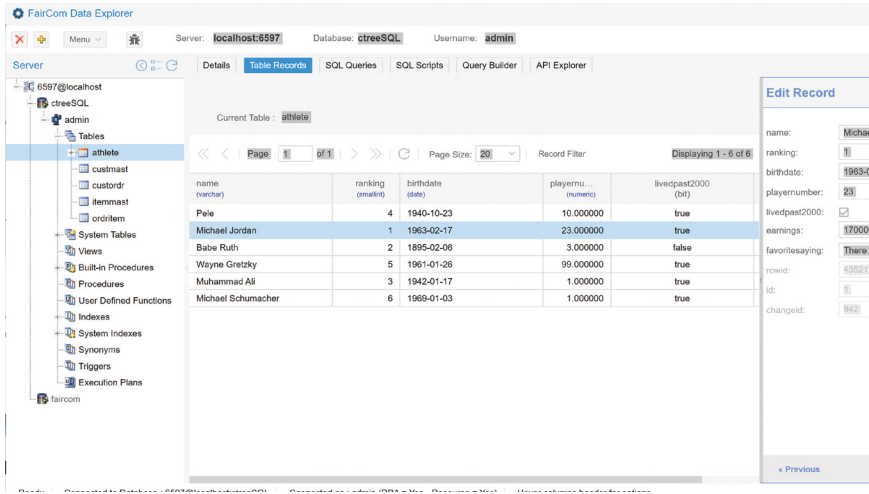
View configuration. Monitor live connections, transactions, files, locks, IO, memory, etc. Capture & view periodic performance snapshots. Dynamically filter log events.



ISAM Explorer

Create, view, and modify FairCom ISAM files and databases.

Data Explorer manages your 'SQL-ized' RTG data and databases



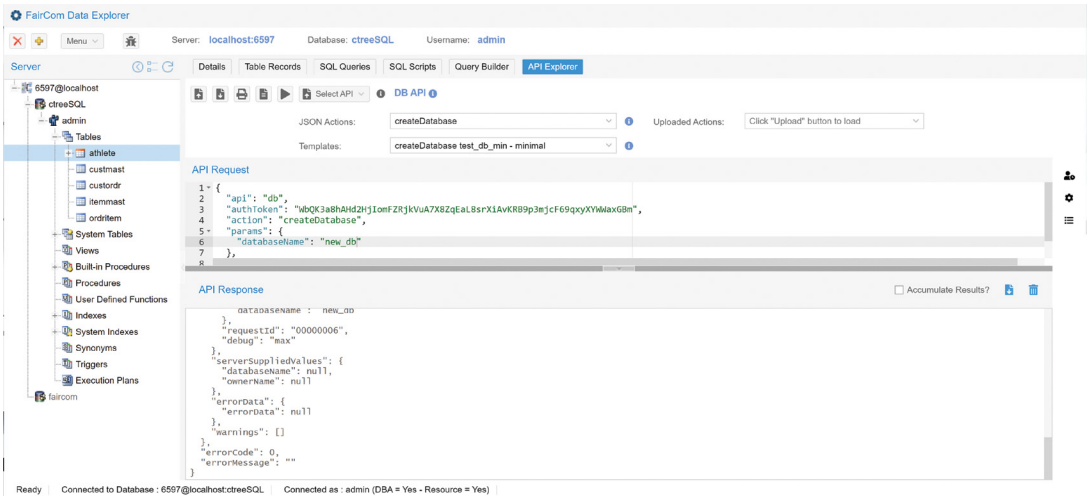
FairCom RTG v5 makes creating, altering, and deleting tables, indexes, and SQL views simpler than ever before.

While Data Explorer has always been able to insert, update, and delete records, it now includes a built-in binary editor to edit binary data.

Use the HEX editor to change any byte of a binary value. View images embedded in binary fields. Save the binary content of any field to disk, edit it on disk, and upload the result right back into the field.

In addition, Data Explorer's new Form View simplifies viewing and editing hundreds of fields in a single record.

API Explorer lets you manage your databases with JSON



Data Explorer has a new API Explorer tab in v5. This integrated development environment for JSON APIs lets you run simple JSON commands to perform all database and administrative operations. It also provides integrated help and settings for each command.

You can copy the JSON commands you create with API Explorer and run them in Python scripts to automate database processes. Build entire applications by embedding JSON commands in your favorite programming language.

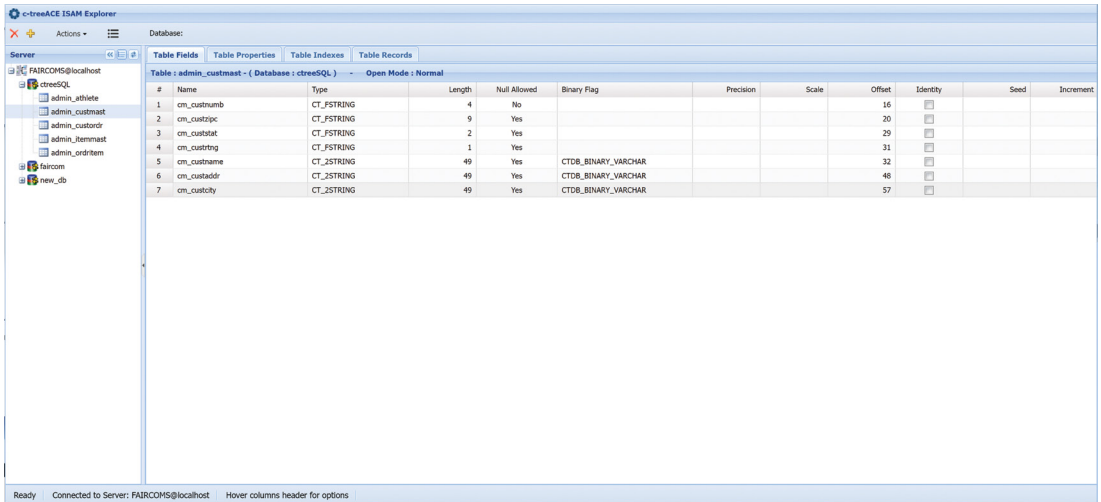
Use the Monitor app in your Ops Center for live dashboards and troubleshooting



The Ops Center provides hundreds of metrics, and you can create and save pages to monitor the ones that matter most. Use Monitor to display live dashboards on multiple monitors in your Ops Centers.

It can also help troubleshoot and solve issues like slow performance, long-running locks, users consuming too many resources, long-running queries, etc.

Use ISAM Explorer to see and manage your COBOL data files from the ISAM perspective



The screenshot shows the c-treeACE ISAM Explorer interface. On the left, a tree view shows the database structure under 'FAIRCOMS@localhost', including 'ctreeSQL', 'admin_adviser', 'admin_custmast', 'admin_custordr', 'admin_itemmast', 'admin_orditem', 'faircom', and 'new_db'. The main pane displays the 'Table : admin_custmast - (Database : ctreeSQL)' in 'Open Mode : Normal'. The table structure is as follows:

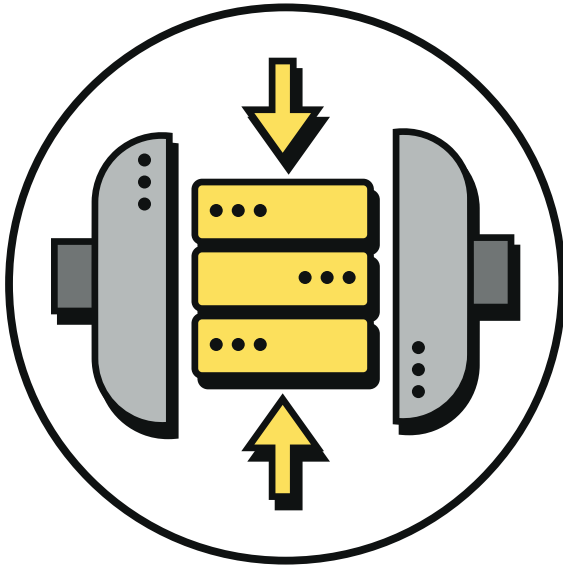
#	Name	Type	Length	Null Allowed	Binary Flag	Precision	Scale	Offset	Identity	Seed	Increment
1	cm_custnumb	CT_FSTRING	4	No				16	☐		
2	cm_custzipc	CT_FSTRING	9	Yes				20	☐		
3	cm_custstat	CT_FSTRING	2	Yes				29	☐		
4	cm_custring	CT_FSTRING	1	Yes				31	☐		
5	cm_custname	CT_FSTRING	49	Yes	CTDB_BINARY_VARCHAR			32	☐		
6	cm_custaddr	CT_FSTRING	49	Yes	CTDB_BINARY_VARCHAR			48	☐		
7	cm_custcity	CT_FSTRING	49	Yes	CTDB_BINARY_VARCHAR			57	☐		

The status bar at the bottom indicates 'Ready Connected to Server: FAIRCOMS@localhost Hover columns header for options'.

Note: This feature requires Unified Header support. Files created before RTG v5 will need to be converted. See the Unified Data Type support discussion for details.

ISAM Explorer handles data files from the ISAM perspective. You can view and set the ISAM properties on a table or index.

In contrast, Data Explorer works with tables and indexes from the SQL and JSON DB perspective.



Dynamic Data and Index Optimization

A v5 server allows your application to [compact and rebuild specific index and data files](#) on the fly without affecting database operations, including reading and writing to these files. This greatly improves file maintenance without causing application downtime.

Cache Enhancements

Dynamic cache resizing

Data caching is one of the most important subsystems of a database server, including FairCom RTG. Caching allows the server to quickly process data in RAM instead of retrieving it from disk.

With v5, your application can resize the FairCom server's data and index caches on demand without restarting the server. This ability is important when the host does not have enough RAM to support the simultaneous needs of the server, applications, and OS caches.

The cache size can also dynamically increase as needed for additional database connections or before running an intensive data-processing task.

You can also have your app dynamically decrease the cache size to return RAM to the OS.

Cache priming at server startup

FairCom servers have long supported loading data into the data cache when an application opens a file. We call this "[cache priming](#)."

FairCom RTG v5 now lets you prime the cache when the server starts by adding the subsystem cache command [prime_cache_at_startup](#) to `ctsrvr.cfg`. This command configures the FairCom server to use a multi-threaded background process to load specific files and indexes into the cache during server startup. You can also adjust cache allocation per file and optionally configure record loading to forward or reverse index order.

Caching at server startup gives the FairCom RTG server a high-speed, in-memory cache with predictable, low-latency response times.

Security

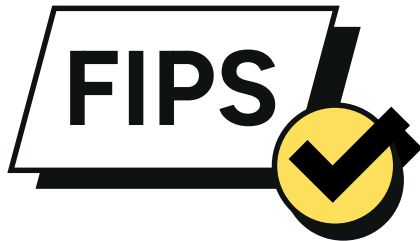
OpenSSL

Cryptography and SSL/TLS Toolkit

OpenSSL v3

FairCom RTG v5 uses [OpenSSL 3.0](#) for all its secure communications, including TCP, HTTPS, WebSocket Secure (WSS), and MQTT Secure (MQTTS).

FairCom products are FIPS-ready



Federal Information Processing Standards (FIPS) are standards that

protect sensitive or valuable data by verifying software compliance with their cryptographic requirements.

FairCom uses the OpenSSL toolkit to encrypt communications in transit and at rest. OpenSSL integrates with the FIPS-verification technology, and FairCom has completed all necessary integrations with OpenSSL to support the FIPS 140–2 standard.

Client certificate authentication

Client apps can now authenticate with FairCom products using [X.509 client certificates](#) signed by your organization's private certificate authority. Client certificates and their private key are a better alternative to passwords because they are tamperproof, prevent brute-force password attacks, and expire. FairCom command line utilities now support client certificates in addition to usernames and passwords.

Client certificates are tamperproof because the organization cryptographically signs them.

They also impede brute-force attacks because an attacker can't guess the private key or impersonate the certificate. Client certificates expire to limit the time window of an attack.

Private certificate management

You must use certificates to create secure connections to the FairCom server. Secure connections prevent malicious users from impersonating your servers and from listening in to communications between your servers.

FairCom provides Python scripts and tutorials that make it easy for you to [create and manage certificates](#). You can create your own private certificate authority, create server certificates, create client certificates, and renew expired certificates.

You aren't required to use Python scripts, and your security team can create a separate server certificate and private key file for each FairCom server. They're also able to create client certificates to authenticate your software with a FairCom server.

Secure comms by default

To ensure [secure communications](#) by default, an RTG v5 server will only accept connections that use secure encryption algorithms.

You can configure a FairCom server to support less secure or insecure communications as needed.

HSTS enforced by default

FairCom servers include an application server that accepts web connections. A FairCom RTG v5 server supports the HTTP Strict Transport Security (HSTS)

protocol, which prevents insecure connections.

You can override this behavior to allow insecure connections, such as HTTP, WS, and MQTT.

Secure ECDHE connections

FairCom servers support SSL connections using ECDHE when Diffie–Hellman parameters are present in a server certificate file.

Elliptic-Curve Diffie–Hellman Ephemeral (ECDHE) is a key exchange algorithm that allows two parties to establish a shared secret over an insecure communication channel. It is a variant of the Diffie–Hellman key exchange that uses elliptic-curve cryptography to provide more robust security with smaller key sizes.

Signed Windows and MacOS binaries

As a registered and notarized developer for Windows and Apple applications, all FairCom products sign their binaries for Windows and MacOS. These signatures ensure the FairCom product you download comes from FairCom and hasn't been tampered with.

LDAP

FairCom servers have long supported [LDAP \(Lightweight Directory Access Protocol\)](#), an external service that authenticates and authorizes a username and password combination. FairCom supports MS Active Directory, OpenLDAP, and OpenDJ.

Enhanced Record and Field Callbacks

Users coding in C can use these features to improve and customize processes.

FairCom RTG v5 allows you to write callbacks in C to preprocess records. You can modify data before the server stores it in the database and before the server returns it.

These enhanced callbacks also let you create and register a custom function to convert field values when the Hot Alter Table process updates a record.

Registered table callbacks work at the record and field levels.

RTG uses a fallback pattern to run callbacks, making them more manageable. For example, when you don't register a callback library at the table level, the server uses the last used callback library. If you never specified a callback library, the server returns an error.

Larger Default Index Page Size

FairCom RTG v3 increased the default PAGE_SIZE from 8K to 32K. We've received great feedback from this switch and have continued to keep the default PAGE_SIZE at 32K in v5.

RTG v5 also notifies you when you need to rebuild index files to increase performance.

Because most operating and storage systems are now optimized to process data in larger chunks, most applications benefit when RTG processes data with a page size of 32K or 64K. You can change the page size in ctsrvr.cfg.

In v5, FairCom RTG automatically adjusts the page size of its internal files (dictionaries, *.FCS files, etc.) to match the page size in ctsrvr.cfg because it increases data processing speed.

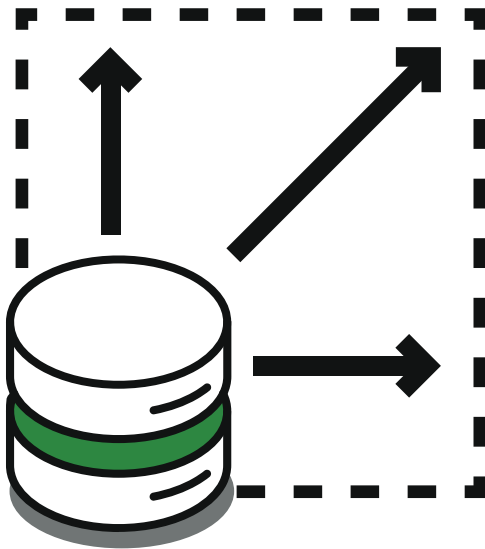
The server adds a warning log entry to CTSTATUS.FCS when the server opens a file with a smaller page size. ***You should check for these warnings and convert application indexes to use the larger page size as needed.***

Dynamic Plugin Management

Dynamically start, stop, and change plugins while the server is running.

FairCom RTG v5 provides a command-line utility and API calls to [start, stop, and change the settings for individual plugins](#) while the server is running, so you no longer need to stop and restart the server. There are a number of available plugins.

Read-Only Servers



Create highly available and scalable solutions.

FairCom RTG v5 lets you create multiple [read-only servers](#) and use data replication to keep them in sync with a source server. This lets you implement high availability while also increasing read scalability. You can also populate a server with data and make it read-only to keep end users from changing its data.

A read-only server blocks normal users and applications from inserting, updating, and deleting records on it. It does not stop data replication from synchronizing changes from a source server to a read-only server. When ReadOnly Mode is turned on or off, the server logs a line to the CTSTATUS.FCS file.

Better Backups

Back up your server without waiting for current transactions to complete

FairCom RTG v5 no longer requires a backup to wait until all outstanding transactions are complete. Add [!ALLOW_TRAN](#) to the dynamic dump script to start a backup without waiting for transactions to finish.

This setting increases the backup size proportional to the active transaction size at the time of the backup. It also requires additional processing when restoring the backup.

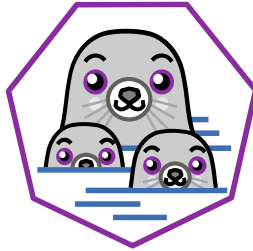
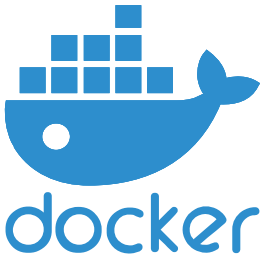
Restore files to a specific path

With v5, you can restore files to a different path than the original backup path, which is great when restoring a backup for a new database. Use the [!REDIRECT](#) option in a dynamic dump restore script to redirect all directory names to a specified directory name. The new directory name can be a full or a relative path.

Improved performance of backup status messages

RTG v5 delivers backup status messages efficiently and no longer slows the backup process when the client is on a different computer than the server.

Docker and Podman



Run FairCom RTG in Docker and Podman for deployment to hosting services, such as Kubernetes.

Creating a [Docker or Podman](#) image of FairCom RTG is already simple; but to make it even easier, FairCom RTG v5 now provides a tutorial and utility to help automate the process.

Rocket Enterprise Suite (formerly Micro Focus)

Rocket Software offers some of the latest mainframe modernization capabilities with Rocket Enterprise Suite 9.0, enabling you to transform existing mainframe applications, processes, and infrastructure to meet your business needs.

FairCom engineers have extensive experience working with enterprise customers using Rocket Software. By providing a COBOL Driver compatible with Rocket Enterprise Suite, the FairCom RTG Server offers a fast and reliable engine to replace the database (VSAM) support on the mainframe.

This is a very specialized domain, and we encourage any mainframe users who are considering migrating to a Linux-based open system alternative to give us a call.

Improved Logging

```
Fri Feb 16 13:23:06 2024 - User# 00001 SQL Subsystem Started
Fri Feb 16 13:23:06 2024 - User# 00022 FairCom App Server-12.6.2.137(Build-240119)
Fri Feb 16 13:23:06 2024 - User# 00022 cthttpd - Starting App server
Fri Feb 16 13:23:06 2024 - User# 00001 Notification service initialized
Fri Feb 16 13:23:06 2024 - User# 00001 Waiting for plugins to finish their initialization
Fri Feb 16 13:23:06 2024 - User# 00025 Server using TCP/IP Socket Port Number: 5597
Fri Feb 16 13:23:06 2024 - User# 00022 cthttpd - Loaded web app: Ace Monitor
Fri Feb 16 13:23:06 2024 - User# 00022 cthttpd - Loaded web app: Data Explorer
Fri Feb 16 13:23:06 2024 - User# 00022 cthttpd - Loaded web app: ISAM Explorer
Fri Feb 16 13:23:06 2024 - User# 00022 cthttpd - Loaded web app: Rest CRUD
Fri Feb 16 13:23:06 2024 - User# 00022 cthttpd - Loaded web api: admin
Fri Feb 16 13:23:06 2024 - User# 00022 cthttpd - Loaded web api: db
Fri Feb 16 13:23:06 2024 - User# 00026 Creating Database...
Fri Feb 16 13:23:06 2024 - User# 00026 faircom
```

Each log entry is now shown on a single line

The CTSTATUS.FCS status file defaults to a [single-line format](#), making it easier to script and automate reading and parsing the file.

System logs grow larger and purge faster

In FairCom RTG v5, system log files—SYSLOGDT.FCS and SYSLOGIX.FCS (see SYSLOG)—can grow up to 100 million terabytes and be completely purged in seconds.

Enhancements to RTG's Command Line Tool *ctutil*

Using [ctutil](#) lets you examine files, extract data records, change the maximum record size, and rebuild corrupted indexes. It's important in many aspects of FairCom RTG's operation, including preparing data for SQL access and importing BTRV data. See [ctutil Commands](#) for a list of commands organized by type.

Here are additional enhancements made to this utility in RTG v5:

- **file password support:** *ctutil* now accepts a new global command line argument to specify the file password: "-o word".
- **[ctutil -unload](#):** To read records > 64KB.
- **[ctutil -info](#):** To display "Index node size." This information is helpful when determining if a file was created with a prior version of FairCom RTG. By

default, newer versions of RTG (v3 and later) create files with a 32768-byte node size; prior versions used an 8192-byte node size.

- **[ctutil -conv](#) to retrieve numeric storage convention:** The existing "*ctutil -conv*" command can now retrieve the numeric storage convention of an existing file if the convention ID parameter is omitted.
- **[ctutil -ddf2xdd](#) enhancement on produced file names:** For some tables, the name in the DDF does not match the file name. This causes difficulties when trying to match the XDD with the file on disk during the "sqlinfo" or "sqlize" process because we use the DDF name to determine the XDD file name. RTG v5 now uses the "file location" to determine the XDD file name so that it matches the data file name.

- **ctutil -alter:** To display new record length on success.
- **ctutil -applyrules:** See the outcome of applying rules to an existing XDD file. While this was already possible using the ctutil “-xfd2xdd” command (in ctutil, the XDD and xfd field are silently interchangeable), it was counterintuitive. This new command has been added to specifically apply rules to an XDD file.
- **ctutil -copy:** Prevent file overwrite during FairCom RTG copies. The default behavior of the FairCom RTG “ctutil -copy” is to overwrite the destination file, if it exists. This modification introduces support for not overwriting the existing destination file. A specific error is now returned in this case.

More RTG Enhancements

Faster RTG prefetch with ‘allowwriters’

The average time of the RTG_NEXTPREV function increases with the number of records added in a transaction.

The FairCom RTG [<prefetch>](#) functionality locks records being prefetched until they are actually read by the user to prevent a prefetched record from being changed by another user before it’s actually read. The prefetch locks are read locks, and the prefetched records are freed when a new set of records is prefetched.

When `<prefetch allowwriters=“yes”>` is enabled, a prefetch lock is not placed so that another user can update prefetched records.

The call to free the locks was previously performed when `<prefetch>` was enabled, regardless of the `<prefetch allowwriters>` option.

An unnecessary lock call is now avoided if the `<prefetch>` option is enabled, thereby improving performance.

Performance improvements

The table below shows the dramatic improvement in performance due to page size modification when running a very long transaction.

	pre-v5 (8k)	v5 (8k)	v5 (32k)
Time	08:53	03:10	01:15
Records/Second	485	1360	3446

The numbers in parenthesis indicate the setting for PAGE_SIZE. Notice the impressive gain when PAGE_SIZE is set to 32K.

More `<prefetch>` improvements

- An unexpected lock error requesting READ WITH WAIT LOCK when `<prefetch>` is enabled has been resolved.
- Enhanced RPC logic to allocate the smaller input and output buffers.
- Enhanced performance performing START+READ NEXT operations when `<prefetch>` is enabled.

Enforce maximum RTG file size

A new `<maxsize>` configuration can be enabled to specify the maximum file size limit. The default value is 0, which means that files can grow up to the limit of the file system where the file is located. The `<maxsize>` value unit is MB.

When a file is created with this option enabled, an error is returned when the file reaches the specified size in MB. For example, if a file is created with `<maxsize>1024</maxsize>`, the data file can grow up to 1 gigabyte.

When the data file reaches the file size limit, an internal FairCom RTG error 51 is returned, which is mapped to a specific interface error. For example, a COBOL 9/194 is returned to Micro Focus COBOL, or BTRV error 132 to a BTRV application.

In addition, a new ctutil command is available to set the maximum file size of an existing file: `-maxsize`. The ctutil `-maxsize` usage syntax is: `ctutil -maxsize file max_file_size` where *file* is the file name without extension and *max_file_size* is the maximum file size for a data file in MB. An attempt to set the maximum file size limit on a file that already has a larger file size will fail.

‘Locked Record Error’ messages now include task ID

COBOL applications are frequently written with a pessimistic locking strategy that can lock other users out of a record, requiring action from a system administrator. Previous versions of RTG made identifying the locking user tedious, as they identified connections with an internal task ID rather than simple process PIDs.

In FairCom RTG v5, all error messages regarding record/file locks also print the internal RTG task ID of the locker thread. For example, “ERROR 5:42:0 record is locked by taskid:24”.

This task ID owner can then be directly determined from other RTG tools such as FairCom DB Monitor and ctadmn, which also provide the IP address and other connection attributes associated with that taskid.

Additional RTG Progress

- **Automatically reconnect to server upon disconnection:** Use the [<instance autoreconnect>](#) attribute to automatically reconnect to server upon disconnection.
- **Co-file lock mode that requires all co-files to release the lock:** We've implemented new [record-level unlock behavior](#) that matches a type of unlock behavior that COBOL supports.
- **Two-phase commit support:** This modification introduces support for two-phase commit in RTG by implementing the FairCom DB function [TRANRDY\(\)](#).
- **EXTFH logging attribute to select whether to dump FCD data:** This revision introduces the attribute [<extfh fcd>](#) to [<log><debug><extfh>](#) setting to indicate if the FCD data structure should be dumped to the RTG log file. The [<extfh fcd>](#) attribute is disabled by default, meaning that only the EXTFH opcode is written to the RTG log file.
- **Function Monitor to log named RTG called functions:** The Function Monitor is a FairCom RTG subsystem for directly logging API calls sent to the server process. It's extremely useful when diagnosing certain classes of performance problems and tracking specific return errors. Previously, Function Monitor did not show the names of RTG functions, as they all come in via a single RPC call. The names of these functions are now resolved and displayed along with the return code (retval) providing a detailed diagnostic call log.
- **New log messages for <log><debug><file>:** The [<log><debug><file>](#) configuration enables logging messages when a file is created, opened, or closed.

No message is logged for other file operations, such as remove, rename, recopy, etc. This modification adds `<log>` `<debug>` `<file>` messages for file remove, rename, and copy.

- **Configuration option to create backward-compatible files:** This modification introduces a new configuration option `<fileversion>` that can be set to a major version number of RTG (e.g., `<fileversion>2</fileversion>`).
- **Log current process pid for LOCK errors:** The `<log>` configuration attribute `<log whoformat>` allows the specification of the format of the log entry portion that prints which client application (who) logged the entry.

The value of `<log whoformat="">` is a character string that can contain [substitution specifiers](#) such as `%p` for process ID, `%t` for thread ID, etc. The default value is `<log whoformat="%08Xt> ">` which prints the thread ID ("`%t`") in uppercase

hexadecimal format ("`%Xt`") with an 8-digit fixed width ("`%8Xt`"), left-padded with zeroes (0) instead of spaces ("`%08Xt`"), and followed by a literal `> "` as a separator ("`%08Xt> "`).

- **Owner attribute for log message:** You can now use an owner `"%o"` substitution specifier that is expanded as part of the server task ID. The task ID is available as soon as an instance is successfully connected to a server. Before the connection occurs, the value of the task ID is zero (0).

`"%o"` substitution can be used in all configuration elements where substitution specifiers are accepted. For instance, it can be used to modify the `<log>` entry format as follows: `<log whoformat="%03o> ">`, which prints the task ID ("`%o`") with a 3-digit fixed width ("`%3o`"), left-padded with zeroes (0) instead of spaces ("`%03d`"), and followed by a literal `> "` as separator ("`%03o> "`).

- **Prevent duplicate file name extensions with new configuration element attributes:** The `<datafilesuffix>` configuration element sets the data file extension of files created by FairCom RTG. The file extension defined by `<datafilesuffix>` is appended to the file name passed by the application when the file is created. Sometimes a file name passed by an application already includes the file extension, and the resulting data file name has a double extension. For example, if the application passes “custmast.dat” and the `<datafilesuffix>` is “.dat”, the resulting data file name is “custmast.dat.dat”.

Two new attributes have been added to the `<datafilesuffix>` configuration element:

- `<datafilesuffix append= [ “y” | “n” ] >`
- `<datafilesuffix casesensitive= [ “y” | “n” ] >`.

The `<datafilesuffix append>` value is a Boolean that defines if the file extension should be appended to the file name. If it is true, the specified data file suffix is appended to the data file name regardless if the data file name already ends with the specified data file suffix. If it is false, the data file suffix is appended only if the file name does not end with the specified data file suffix. The default of `<datafilesuffix append>` is true.

The `<datafilesuffix casesensitive>` value is a Boolean that defines if checking the end of the data file name for the data file suffix should be case sensitive. If it is true, the check is case-sensitive, otherwise it’s case-insensitive. The default of `<datafilesuffix casesensitive>` is true. This configuration is meaningful only if `<datafilesuffix append>` is false.

SQL Improvements

Faster SQL performance

[Large string fields](#)—up to 65,500 bytes—allocate memory more efficiently for faster performance.

Converting the NUMBER type to BIGINT is now 100 times faster.

More capable SQL

SQL and all other FairCom APIs can specify when the server sets the value of an [autotimestamp](#) field. The server can set the timestamp when you insert a record, update a record, or insert and update a record.

Use [SYSLOG_EXCLUDE_SQL_USER](#) to configure the server to turn off SQL logging for specific users, such as system processes that fill the logs with unhelpful events.

[Stored procedures](#) can access error codes and messages using the

`DhSQLException.getSqlState()` and `DhSQLException.getMessageText()` functions.

Improvements to Direct SQL API for C and C++ programmers

FairCom has added new cursor functions to the [Direct SQL API](#). Cursor functions efficiently retrieve records from a SQL query by tracking the cursor's current position. Cursors can efficiently paginate data, tail the end of a file, retrieve records forward and backward, get the top N records, skip records forward or backward, and repeatedly jump to the first and last record.

We've also added new functions to the Direct SQL API for sending in parameters to stored procedures and setting the value of output parameters.

Btrieve Compatibility: Enhanced Support

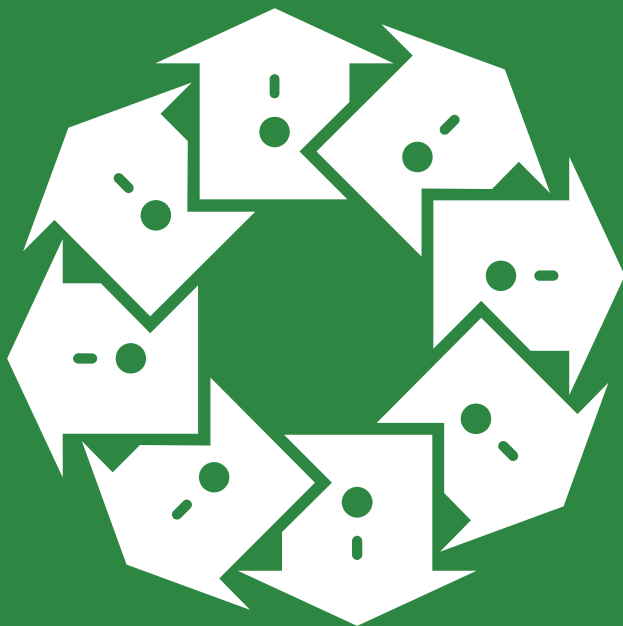
We're happy to continue supporting customers who enjoy our BTRV Edition of FairCom RTG.

We've added a number of new operations.

Highlights include:

- Any-segment and all-segment NULL key flags
- Operation Find Percentage or B_GET_PERCENT (opcode 45)
- Operations GET NEXT/PREVIOUS EXTENDED
- Owner and Password support: SET OWNER and CLEAR OWNER
- Alternate collating sequence for GNXSEG key segments
- Operation GET DIRECT/CHUNK
- Indirect record buffers in operation GET DIRECT/CHUNK
- Operation UPDATE CHUNK

- Support for BLOB and CLOB
- Support for NOPASS file permission allows read-only access
- LOGIN function parameter to set existing <instance> user credentials
- 'GET KEY' functions
- LOGICAL and TIMESTAMP Key Types
- Manually defined key numbers
- Data-only files
- Operations BTRV B_STEP_NEXT_EXT and B_STEP_PREVIOUS_EXT
- CHUNK support for reading variable length records larger than 64KB
- Filter support for CHUNK retrievals
- CT_BT_BIT field type for BTRV support needs.
- Case insensitive ZSTRING
- UNC file names in ctbutil utility: The UNC file name is translated into a URL (ctree://hostname:servername/filename) and then passed to RTG



FairCom