



FairCom®

c-tree **Plus**®
V9

**c-tree ODBC
Driver**
c-tree V4 Edition Guide

c-tree ODBC Driver

c-tree V4 Edition Guide



Copyright © 1992-2008 FairCom Corporation All rights reserved. No part of this publication may be stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior written permission of FairCom Corporation. Printed in the United States of America.

Information in this document is subject to change without notice.

Trademarks

c-tree, c-tree Plus, r-tree, the circular disk logo, and FairCom are registered trademarks of the FairCom Corporation. c-treeACE SQL, c-treeACE SQL ODBC, c-treeACE SQL ODBC SDK, c-treeVCL/CLX, c-tree ODBC Driver, c-tree Crystal Reports Driver, c-treeDBX, and c-treePHP are trademarks of FairCom Corporation. The following are third-party trademarks: AMD and AMD Opteron are trademarks of Advanced Micro Devices, Inc. Macintosh, Mac OS, and Xcode are trademarks of Apple Inc., registered in the U.S. and other countries. Borland, the Borland Logo, Delphi, C#Builder, C++Builder, Kylix, and CLX are trademarks or registered trademarks of Borland Software Corporation in the United States and other countries. Business Objects, the Business Objects logo, Crystal Reports, and Crystal Enterprise are trademarks or registered trademarks of Business Objects SA or its affiliated companies in the United States and other countries. DBstore is a trademark of Dharma Systems, Inc. HP and HP-UX are registered trademarks of the Hewlett-Packard Company. AIX, IBM, OS/2, OS/2 WARP, and POWER5 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. Intel, Itanium, Pentium and Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. LynxWorks, LynxOS and BlueCat are registered trademarks of LynxWorks, Inc. Microsoft, the .NET logo, MS-DOS, Visual Studio, Windows, Windows Mobile, Windows server and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Novell and NetWare are registered trademarks of Novell, Inc., in the United States and other countries. QNX and Neutrino are registered trademarks of QNX Software Systems Ltd. in certain jurisdictions. Red Hat and the Shadow Man logo are registered trademarks of Red Hat, Inc. in the United States and other countries, used with permission. SCO and SCO Open Server, are trademarks or registered trademarks of The SCO Group, Inc. in the U.S.A. and other countries. SGI and IRIX are registered trademarks of Silicon Graphics, Inc., in the United States and/or other countries worldwide. Sun, Sun Microsystems, the Sun Logo, Solaris, SunOS, JDBC, Java and all Java-based trademarks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. UNIX and UnixWare are registered trademarks of The Open Group in the United States and other countries. Linux is a trademark of Linus Torvalds in the United States, other countries, or both. All other trademarks, trade names, company names, product names, and registered trademarks are the property of their respective holders.

FairCom welcomes your comments on this document and the software it describes. Send comments to:

Documentation Comments
FairCom Corporation
6300 W. Sugar Creek Drive
Columbia, MO 65203

Portions © 1987-2008 Dharma Systems, Inc. All rights reserved. This software or web site utilizes or contains material that is © 1994-2007 DUNDAS DATA VISUALIZATION, INC. and its licensors, all rights reserved.

6/26/2008

CONTENTS

Quick Start	1
1.1 ODBC History/Overview	1
1.2 Installation	1
1.3 Tutorial Setup.....	1
1.4 Tutorial	2
1.5 Tutorial File Layout	4
Getting Started with Your Data	7
2.1 Installing a Vendor Supplied Script.....	7
2.2 Browse Method	8
2.3 Script Method	9
ODBC/Application Setup	11
3.1 ODBC Setup Dialog Options.....	11
3.2 Options>>.....	12
3.3 FairCom Data Dictionary - In Depth.....	14
Defining Multiple Databases	14
Adding Files to the Dictionary	14
3.4 Generic ODBC Compliant Application Notes.....	14
ctree ODBC Driver Technical Details	17
4.1 ctree ODBC Driver Purpose.....	17
4.2 Driver Types.....	17
4.3 Requirements.....	18
4.4 ODBC Compliance.....	18
4.5 SQL Conformance	18
Minimum SQL Grammar	19
Core SQL Grammar	19
Extended SQL Grammar	21
4.6 Driver Constraints	22
4.7 Licensing	23
Advanced Topics	25
5.1 Performance Optimization	25
5.2 Data Dictionary Creation - Import Method	25
Import Script.....	26
Wildcard	26
Multiple Database Script	26
5.3 OTP Files	26
OTP File Requirements	27
OTP File Organization	27
OTP File Layout	27
OTP File Contents	27
Data File Description Record.....	28
Field Description Record.....	28
Index File Description Record.....	29
Optional Index Member Record.....	30
Key Segment Description Record.....	31
filmod Values.....	31
segmode Values	32
5.4 File Locations	34

Table of Contents

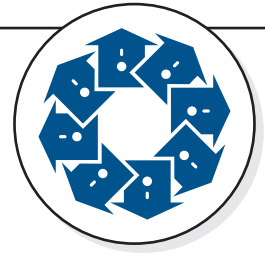
32-bit Microsoft Supplied Files.....	34
32-bit FairCom Supplied Files.....	34
FairCom Supplied Sample Files	34
5.5 Registry Options	35
Force Close Option	36
Adding a Shareable DSN.....	36
Optionally Disable Floating Point Unit Exceptions.....	37
5.6 DSN-less ODBC connections	37
<hr/>	
Errors	39
6.1 DBOPEN DICTDBGETBYNAME(FAIRCOM.DB)=101	39
6.2 No such database	39
6.3 No table names appear in the Select Table dialog box	39
6.4 Not able to open *.FCS file	39
6.5 Specified driver could not be loaded.....	39
6.6 System Catalog's Files Need to be Rebuilt	40
<hr/>	
Glossary	41
Index	43

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.
<i>FairCom Terminology</i>	FairCom technology term.
FunctionName()	c-tree Function name.
<i>Parameter</i>	c-tree Function Parameter.
Code Examples	Code example or Command line usage.
utility	c-tree executable or utility.
<i>filename</i>	c-tree file or path name.
CONFIGURATION KEYWORDS	c-treeACE Configuration Keyword.
BIG_ERR	c-tree Error Code.



Quick Start

Welcome to the c-tree® ODBC Driver - c-tree® V4 Edition. This powerful tool opens many possibilities for accessing and maintaining the data created by your c-tree V4 application. This guide is intended for the final user of the c-tree ODBC Driver and assumes an intermediate knowledge of the Microsoft Windows operating system.

This chapter provides a quick start with the c-tree ODBC Driver. It includes:

- A high-level overview of ODBC,
- Installation instructions, and
- An introductory tutorial.

1.1 ODBC History/Overview

Today's computer user demands direct and easy access to personal and corporate data using popular off-the-shelf applications. Microsoft Corporation's ODBC standard provides this type of open connectivity.

Database drivers that map a particular vendor's API (Application Programming Interface) to the ODBC standard API make this connection possible. This process is analogous to Windows printer drivers. The printer drivers allow an application developer to support virtually every printer. The developer programs to the Windows printer interface, and the printer manufacturer provides a driver that works with the Windows interface to drive the printer. Though database portability is more complicated than the printer support, the two are conceptually similar.

The c-tree ODBC Driver gives ODBC compliant applications, like Microsoft Access, Seagate Crystal Reports, and Visual Basic, access to c-tree V4 files.

1.2 Installation

Place the c-tree ODBC Driver disk into the appropriate drive. If Setup does not start automatically, execute **setup.exe** from the c-tree ODBC Driver disk and follow the instructions on the installation screens.

1.3 Tutorial Setup

The c-tree ODBC Driver installs ready to use with the tutorial files. Working through a quick tutorial increases familiarity with the operation of the Driver. This tutorial uses Microsoft Query. Even if you do not have Query, we still recommend reading this section and following along in your chosen ODBC

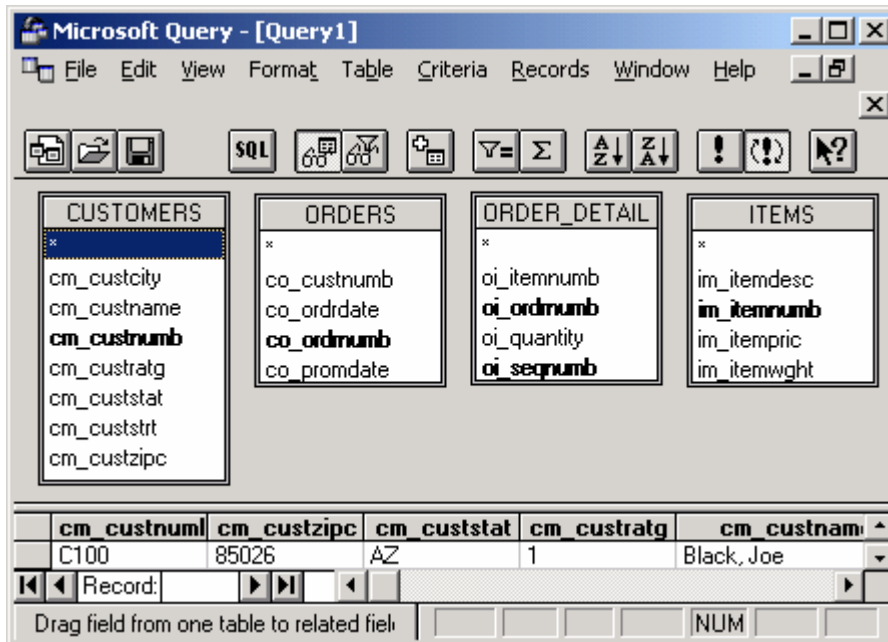
application. You will find the information in this chapter helpful since most ODBC applications share similar appearance and behavior with regard to defining and accessing ODBC Drivers.

A database can be thought of as a collection of related files. Large applications may have many different databases. Typically, files in the same database are related through c-tree V4 indices. The tutorial consists of a database of four data files and their associated indices. These files are described at the end of this chapter.

1.4 Tutorial

1. Start Microsoft Query (double click the Microsoft Query icon).
2. Select **New...** from the Microsoft Query File menu.
3. From the Choose Data Source dialog box, choose the Databases tab. For simplicity, remove the checkmark from "Use the Query Wizard to create/edit queries". Highlight the c-tree ODBC Driver and click **OK**.
4. In the Add Tables dialog box listing the four sample files, continue by adding all four files in the following order (Choose each file and select Add). After all four files are added, select Close from the Add Tables dialog box. The Microsoft Query active window should appear similar to the figure below.

Figure 1: Microsoft Query Active Window



5. Add file relationships between the four by selecting the cm_custnumb field in the CUSTOMERS file by clicking and holding the left mouse button. Drag the cursor over to the co_custnumb field in the ORDERS file and release the left mouse button. This should draw a line between the two fields, as shown in the following figure.

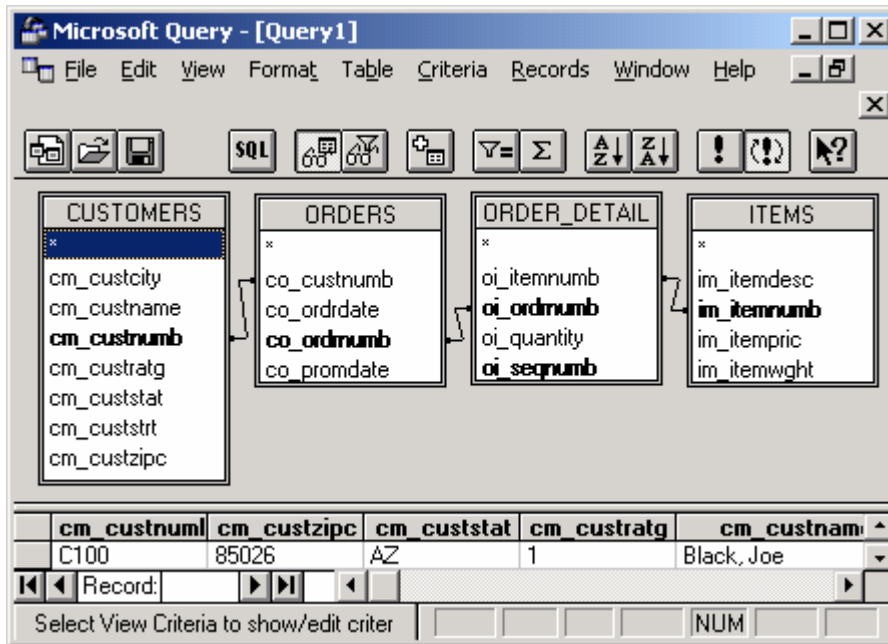
Using the following table and figure, relate the ORDERS file to ORDER_DETAIL and ORDER_DETAIL to the ITEMS file.

Data File 1	Common Field	Data Field 2	Common Field
CUSTOMERS	cm_custnumb	ORDERS	co_custnumb
ORDERS	co_ordrnumb	ORDER_DETAIL	oi_ordrnumb

Data File 1	Common Field	Data Field 2	Common Field
ORDER_DETAIL	oi_itemnumb	ITEMS	im_itemnumb

See "[Tutorial File Layout](#)" for complete file descriptions.

Figure 2: Microsoft Query Active Window with Lines



- Once the files are related, double-click any field to retrieve the data for that file. Since the files are all related, the corresponding data across files will be properly aligned.

Note: To select all fields for a particular file using Microsoft Query, double click the asterisk (*) at the top of the file box.

- The last step for this tutorial, which is optional, is to insert (write) a few new records into a data file. By default, all c-tree Drivers are read ONLY. It is possible to enable writes for all but the c-tree Read ONLY Drivers, but by doing so you are taking responsibility for the integrity of your data. The authors of your c-tree V4 application have taken extreme care to ensure the integrity of your data files. For example, in an accounting system, an order total may be the sum of several individual products. If you override the price of one of the products, and do not adjust the order total accordingly, the data integrity could be compromised. If you are prepared to take responsibility for the integrity of your data, the c-tree ODBC Driver can be enabled for writing by using the ODBC Setup icon (see "[ODBC/Application Setup](#)") and changing the file modes in the OTP file from SHARED to EXCLUSIVE. If you are unsure about the ramifications of enabling the write feature, consult the software vendor of your c-tree V4 application. If you decide to enable the write capability, the following steps may be used to verify its operation. Microsoft Query only allows single file updates. For this reason, do the following:

- Remove all but one of the data files from your query. We deleted Orders, Order_Detail, and Items. This is done by highlighting the file to be removed and pressing the delete key.
- Pull down the Records menu and enable Allow Editing.
- Display the desired fields.
- Using the Customers scroll bar, move to the end of the Customers data. The last record in a file enabled for inserts will have an asterisk (*) in the first column.

- e) Enter data into the desired fields and then press Enter to insert the data. Once the last field has been filled in, press the Enter or Tab key to go to the first column of the next row. This inserts the new data and updates the record count in the bottom left corner of the dialog box.

1.5 Tutorial File Layout

The Tutorial uses the following four data files and six indices:

Data file	Symbolic index name	Key segment(s)	Unique
custmast.dat	cm_custnumb_idx	cm_custnumb	yes
itemmast.dat	im_itemnumb_idx	im_itemnumb	yes
custordr.dat	co_ordnumb_idx	co_ordnumb	yes
custordr.dat	co_custnumb_idx	co_custnumb	no
ordritem.dat	oi_ordnumb_idx	oi_ordnumb, oi_seqnumb	yes
ordritem.dat	oi_itemnumb_idx	oi_itemnumb	no

Note: The unique column indicates if the index accepts duplicate data values.

custmast.dat, the Customer Master File, contains the table CUSTOMERS made up of the fields shown below:

Symbolic name	Field description	Field type	Field length
cm_custnumb	Customer number	CT_STRING	4
cm_custzipc	Customer zip code	CT_STRING	9
cm_custstat	Customer state	CT_STRING	3
cm_custratg	Customer rating	CT_STRING	1
cm_custname	Customer name	CT_STRING	47
cm_custadr	Customer address	CT_STRING	47
cm_custcity	Customer city	CT_STRING	47

itemmast.dat, the Item Master File, contains the table ITEMS made up of the fields shown below:

Symbolic name	Field description	Field type	Field length
im_itemwght	Item weight	CT_INT4	4
im_itempric	Item price	CT_MONEY	4
im_itemnumb	Item number	CT_STRING	5
im_itemdesc	item description	CT_STRING	48

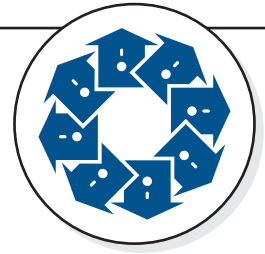
custordr.dat, the Customer Order File, contains the table ORDERS made up of the fields shown below:

Symbolic name	Field description	Field type	Field length
---------------	-------------------	------------	--------------

Symbolic name	Field description	Field type	Field length
_o_delflag	Order delete flag	CT_INT4	4
co_ordrdate	Order date	CT_DATE	4
co_promdate	Order promise date	CT_DATE	4
co_ordrnumb	Order number	CT_STRING	6
co_custnumb	Customer number	CT_STRING	4

ordritem.dat, the Order Item File, contains the table ORDER_DETAIL made up of the fields shown below:

Symbolic name	Field description	Field type	Field length
_i_delflag	Order item delete flag	CT_INT2	2
oi_seqnumb	Order item stock number	CT_INT2	2
oi_quantity	Order item quantity	CT_INT2	2
oi_ordrnumb	Order item number	CT_STRING	6
oi_itemnumb	Item number	CT_STRING	5



Getting Started with Your Data

The FairCom Data Dictionary, the Driver's catalog of available data files, created in ["Quick Start"](#) was specific to the sample tutorial. This chapter helps you get an ODBC compliant application operational with your c-tree V4 data. To take full advantage of the c-tree ODBC Driver features, consult ["ODBC/Application Setup"](#) and ["Advanced Topics"](#) for additional application setup information and performance tips.

There are a few ways to provide the information needed to build the data dictionary: with a vendor-supplied script, by browsing for the files, or with a locally created script.

- If your vendor provided a script, this is the simplest solution. Proceed to ["Installing a Vendor Supplied Script"](#).
- To simply browse your disks for the necessary files, proceed to ["Browse Method"](#), the next simplest method.
- If you know the names and locations of the files and would like to create a script, proceed to ["Script Method"](#).

2.1 Installing a Vendor Supplied Script

All ODBC Drivers need some form of data dictionary to provide file definitions to ODBC compliant applications. In an effort to make the c-tree ODBC Driver easier to install, FairCom has made it possible for your software vendor to create the FairCom Data Dictionary script for you.

Check the c-tree ODBC Driver package for an existing script file named *VENDOR.DB*. Check the pocket inside the cover of this guide for a document titled, "Notes from your Software Provider". This document contains additional instructions from your software vendor. If your vendor did not supply a *VENDOR.DB* file, you must use either the Browse Method or the Script method to create your data dictionary..

To create the dictionary, copy *VENDOR.DB* to the Data Dictionary Path, *C:\FAIRCOM\ODBC\V4* by default, and rename it to the Script Name, *FAIRCOM.DB* by default, as follows:

```
COPY A:\VENDOR.DB C:\FAIRCOM\ODBC\V4\FAIRCOM.DB
```

Note: You may receive a warning message stating that *FAIRCOM.DB* already exists. Most likely, an existing *FAIRCOM.DB* is left over from the tutorial, in which case you can answer the prompt with a Yes to replace the file.

The FairCom Data Dictionary (*CTSYSCAT.FCS*) will now automatically be created upon the first access to the c-tree ODBC Driver by an ODBC compliant application. Once the data dictionary is created, you can delete *FAIRCOM.DB*.

2.2 Browse Method

This method allows you to browse in search of the c-tree V4 files to be included in the Driver Data Dictionary. Enable this feature by specifying a wild card identifier in the Script Name setting of the c-tree ODBC Driver setup window, as follows:

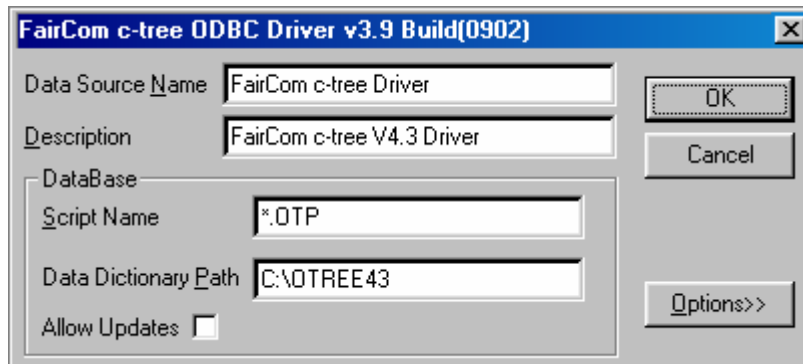
1. Access the c-tree ODBC Driver setup window by selecting the ODBC icon (32-bit ODBC for the 32-bit drivers) from the Control Panel.
2. Highlight the appropriate c-tree ODBC Driver and click **Configure....**
3. In the c-tree ODBC Driver setup window, enter a desired database file name filter by inserting a file name and wildcard combination (as discussed below) into the Script Name prompt. The two standard wild cards supported are as follows:
 - * match all characters
 - ? match a single character

For example, with the following OTP files: *ABBC.OTP*, *ABCD.OTP*, and *ABDD.OTP*.

- Entering AB* matches all three files.
- Entering AB?D.OTP matches *ABCD.OTP* and *ABDD.OTP* only.

The figure below shows a wildcard search for *.OTP, indicating find all files that have a .OTP file extension.

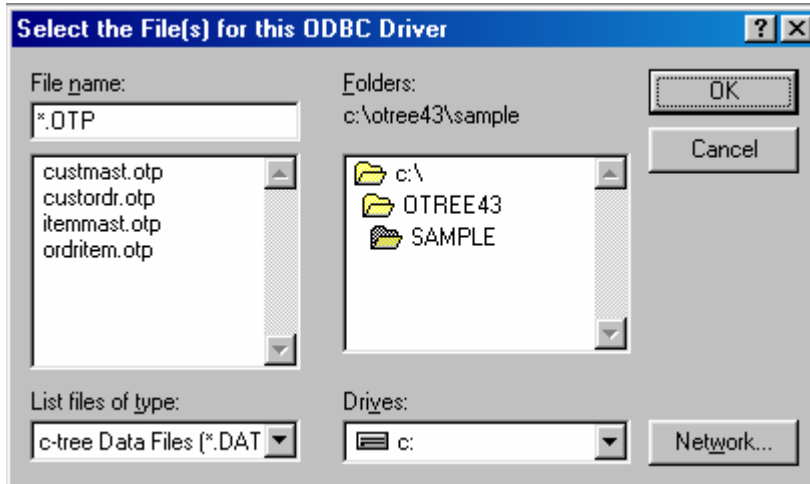
Figure 3: Wildcard Search



4. Once the Script Name setting contains a wildcard, close the Setup window by clicking **OK**. In the Data Sources window, select **OK**.

5. The first ODBC compliant application that selects the c-tree ODBC Driver will cause a Database File Selection window to appear asking if you “want to browse for the files to be used by this ODBC Driver”. By answering “Yes”, you are presented with the dialog in the figure below.

Figure 4: Selecting Files



6. From the File Browse window, highlight the files you would like available to the c-tree ODBC Driver. Please note the following two points:
- To select multiple files, hold down the shift key or control key while selecting files.
 - Files can be selected only from one directory at a time. If you have files in multiple directories, select all of the files from the first directory then click OK. After inserting these files, the c-tree ODBC Driver asks if you wish to select additional files. If so, click Yes to include files from other directories and append them into one common FairCom Data Dictionary.
7. Once you have selected all of the files, answer **No** to indicate that there are no more files to select. The c-tree ODBC Driver will automatically create the FairCom Data Dictionary, *CTSYSCAT.FCS*.

2.3 Script Method

By creating a simple text script, the FairCom Data Dictionary can be created automatically upon the first access to the c-tree ODBC Driver by an ODBC compliant application. A sample text script, *FAIRCOM.DB*, is included and is placed in the ODBC installation directory (*C:\FAIRCOM\ODBC\V4* by default). The format of the text script is as follows. *alias_name* is the symbolic file name referenced from an ODBC application. *file_name* is the actual file name as it resides on disk, including the directory structure.

```
alias_name file_name
```

The text file should list the file alias names and the actual file names on a separate line for each file, as shown in the following example.

```
CUSTOMERS      C:\FAIRCOM\ODBC\V4\SAMPLE\custmast.OTP
ORDERS         C:\FAIRCOM\ODBC\V4\SAMPLE\custordr.OTP
ORDER_DETAIL  C:\FAIRCOM\ODBC\V4\SAMPLE\ordritem.OTP
ITEMS         C:\FAIRCOM\ODBC\V4\SAMPLE\itemmast.OTP
```

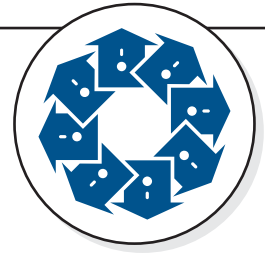
The steps to create the FairCom Data Dictionary using the script method are:

1. Create the script with the format defined above.
2. Specify the name of the script with the Script Name setting in the c-tree ODBC Driver setup window.
3. Set the Data Dictionary Path setting in the Options section of the c-tree ODBC Driver setup window to the directory where the script resides.

4. With the script in place, when the first ODBC compliant application accesses the c-tree ODBC Driver, the FairCom Data Dictionary, *CTSYSCAT.FCS*, will be created in the Data Dictionary Path from the information located in the script file.

After successfully creating the dictionary, the text script is no longer needed.

Note: A third method for creating the FairCom Data Dictionary is available for advanced users. This method is to use the **import.exe** utility shipped with the c-tree ODBC Driver. See [“Data Dictionary Creation - Import Method”](#) for additional information.



ODBC/Application Setup

This chapter details the c-tree ODBC Driver Setup dialog and provides valuable insight into optimal application setup. This information is important for understanding the features and options required to effectively use the c-tree ODBC Driver.

3.1 ODBC Setup Dialog Options

The c-tree ODBC Driver is optimized to simplify setup. The c-tree ODBC Setup dialog box is accessible from the Windows Control Panel ODBC icon or your ODBC compliant application and provides the following basic settings and options (see the figure below).

Figure 3-1: Setup Dialog Box

Figure 5: Setup Dialog Box

The screenshot shows the 'FairCom c-tree ODBC Driver v3.9 Build(0902)' dialog box. It has a title bar with a close button. The main area contains several input fields and checkboxes. On the right side, there are 'OK', 'Cancel', and 'Options>>' buttons. The 'Options' section is expanded, showing various configuration options.

Field/Option	Value
Data Source Name	FairCom c-tree Driver
Description	FairCom c-tree V4.3 Driver
DataBase	
Script Name	FAIRCOM.DB
Data Dictionary Path	C:\OTREE43
Allow Updates	<input type="checkbox"/>
OEM to ANSI Data Conversion	<input type="checkbox"/>
Max columns per table	256
Column Order	<input checked="" type="radio"/> By Table Position <input type="radio"/> By Name
Max number of files	200
Data Buffer Size	64
Data Padding	<input checked="" type="radio"/> NULLs <input type="radio"/> Spaces <input type="radio"/> Spaces w/NULL Trm
Key Padding	<input checked="" type="radio"/> NULLs <input type="radio"/> Spaces
Index Buffer Size	64
Sector Size	16
Special Data Type Conversion DLL Name	None
Debugging	<input type="checkbox"/> c-tree Plus Debug <input type="checkbox"/> Debug Indexes

Data Source Name

Arbitrary database (application view) name.

Description

Arbitrary description of database.

Script Name

Default: *FAIRCOM.DB*

Database script file name. See [“Options>>”](#). This prompt is not case sensitive. See additional notes in [“Driver Types”](#).

Data Dictionary Path

Default: installed path

FairCom Data Dictionary location. The c-tree ODBC Driver searches this path for the data dictionary script specified in Script Name. If found, the contents of the script are used to recreate *CTSYSCAT.FCS* in this path.

Allow Updates

Default: NO

Enabling this feature allows writes, or inserts, into files available to the c-tree ODBC Driver.

Note: By enabling this option, you will be responsible for maintaining the integrity of your c-tree V4 data. The authors of your c-tree V4 based application have taken extreme care to ensure the integrity of your. For example, in an accounting system, an order total may be the sum of several individual products. If you override the price of one of the products, and do not adjust the order total accordingly, the data integrity could be compromised. If you are unsure about the ramifications of enabling this write feature, consult the software vendor from whom your c-tree V4 based application was purchased.

3.2 Options>>

The **Options>>** button is intended for developers or advanced users only. By selecting the **Options>>** button, it is possible to specify several initialization parameters. For the c-tree tutorial and the first execution of your ODBC compliant application, FairCom recommends using the default values. The following options are available:

OEM to ANSI Data Conversion

Default: OFF

Allows customers to choose to perform the OEM to ANSI conversion for string data. Consult your vendor before changing this setting.

Max columns per table

Default: 256

Specifies the maximum number of columns to be supported by a given file. The maximum is presently 5000. However, please note that some ODBC compliant applications limit this value to 256 or 300. Even though the c-tree ODBC Driver can open tables with up to 5000 columns, only 300 columns may be specified in a given SQL query select list.

Max number of files

Default: 200

Maximum number of concurrently open c-tree V4 files (data plus index).

Buffer Sizes - Data/Index

Default: 64

These values set the amount of memory for data and index caching in multi-user non-server mode only. Typically, the larger the value the better the performance. The memory is calculated as follows: Bytes in RAM = (Sector Size X Buffer Size X 128).

Sector Size

Default: 16

Index node size. Set the same value as the target c-tree V4 files, or larger if they are not superfiles. The default value of 16 yields a 2K node size (16 x 128). Each time this value is changed, *CTSYSCAT.FCS* must be recreated to match the new value. Normally, the default value will be best, but your application vendor will know the optimum value for your c-tree V4 files.

Column Order

Default: Table Position

This prompt specifies how column names (field names) are displayed. The default method lists the columns in the order they appear in the table. Alternatively, the column names can be ordered By Name, which sorts them in alphabetical order before displaying.

Data Padding

Default: NULLs

This option allows the padding byte for string data to be altered.

Note: This option should only be used by advanced users.

Key Padding

Default: Default

This option allows the padding byte for target key values to be altered.

Note: This option should only be used by advanced users.

Special Data Type Conversion DLL Name

This entry allows developers to add support for their own data types with the c-tree Driver SDK. Your vendor will specify any changes required.

c-tree V4 Debug

Default: OFF

Enabling this feature sends debugging information to a log file named *CTODBC.LOG*.

Debug Indexes

Default: OFF

Enabling this feature sends debugging information to a log file named *CTODBC.LOG*.

3.3 FairCom Data Dictionary - In Depth

This section contains expands on the information about the FairCom Data Dictionary provided in the browse and script method sections. Instructions include defining multiple databases, adding files to the dictionary, and general ODBC notes.

Defining Multiple Databases

One important concept is that each database defined by the Script Name setting in the c-tree ODBC Setup dialog box can be thought of as a view into a larger database. The FairCom Data Dictionary (*CTSYSCAT.FCS*) can hold many database names (views), each containing many file (table) names. The fewer files in each database, the faster data access will be. It is advisable to split your application files across many databases, each containing different groups of files.

For example, the tutorial has four files: Customer, Orders, Order_Detail, and Items. If a particular user only requires access to information from the Customer file, create a database with only the Customer file. This is done with the automatic dictionary method by creating a dictionary script with just the customer file, *CUSTOMER.DB*, as follows:

```
CUSTOMERS      c:\FAIRCOM\ODBC\V4\sample\custmast.OTP
```

Using the ODBC Setup icon in the Windows Control Panel, a new ODBC Driver entry can be created that specifies the Script Name to be *CUSTOMER.DB*.

Adding Files to the Dictionary

To add additional files to the FairCom Data Dictionary, create a new script listing all files to include in that database. The new files will be appended to the database name that corresponds to the script name. This also allows you to remove files from a database. The re-creation of the dictionary takes place the next time the c-tree ODBC Driver is accessed.

For example, to add a new file *ORDERS* to *CUSTOMER.DB*, a script would be created as follows:

```
CUSTOMERS      c:\FAIRCOM\ODBC\V4\sample\custmast.OTP  
ORDERS         c:\FAIRCOM\ODBC\V4\sample\order.OTP
```

For this example, ensure the Script Name is set to *CUSTOMER.DB* and the location specified by the Data Dictionary Path is the location of *CUSTOMER.DB*. The next time an ODBC application accesses the c-tree ODBC Driver that references *CUSTOMER.DB* as the Script Name, the *ORDERS* file will be appended to the dictionary. After the file is appended to the dictionary, the script (*CUSTOMER.DB*) is no longer needed.

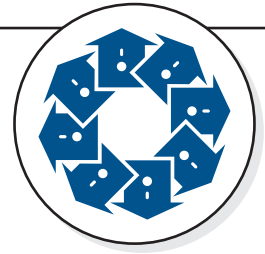
3.4 Generic ODBC Compliant Application Notes

ODBC compliant applications, such as Microsoft Query, keep their own internal list of supported drivers. Therefore, if you remove a driver from the Windows ODBC setup icon, the driver will still appear in the Microsoft Query internal driver list. See your ODBC compliant application instructions for the specific procedures used to remove ODBC drivers from its internal list.

Most ODBC compliant applications provide a mechanism for searching to the first or last record of a file. If a multi-file search does not have file relationships defined, or if they are defined incorrectly, the search may take a long time. For example, if a report listing one field from each of the four files in the c-tree ODBC tutorial was executed without file relationships, it could take several minutes. The ODBC application will search all permutations of field cross-referencing in an effort to establish a proper link. To prevent extended search times, ALWAYS define file relationships for multi-file operations.

When creating multiple database references in the Data Dictionary, *CTSYSCAT.FCS*, it may be necessary to exit the ODBC compliant application in between creating dictionaries. For example, to create a dictionary entry named *CUSTOMER*, create a script named *CUSTOMER.DB* and place it in the directory specified by the Data Dictionary Path setting. Start an ODBC compliant application, such as Microsoft Query. When the Driver referencing *CUSTOMER.DB* as the Script Name is selected, the dictionary will have *CUSTOMER* entered as a valid database name. Before creating a new dictionary entry by referencing a new script, first exit the ODBC application (Microsoft Query in this example).

The c-tree ODBC Driver configuration automatically creates a file DSN during installation, and updates it when saving the current configuration.



ctree ODBC Driver Technical Details

This chapter provides additional background and details for the c-tree ODBC Driver and is not considered required reading.

4.1 ctree ODBC Driver Purpose

The c-tree ODBC Driver is a single-tier driver that interfaces directly to the c-tree V4 API. The term single-tier indicates the driver contains all of the program logic, including an SQL interpreter, to handle requests from a front-end application.

The c-tree ODBC Driver handles the application conversion processes necessary for ODBC compliant applications to access c-tree V4 files. These conversion processes are:

- Connecting the front-end application to a c-tree V4 database (data and index file(s)).
- Defining data - includes retrieving data file-specific information (such as file mode), and creating and deleting data and index files.
- Manipulating data - includes adding, deleting, retrieving and updating database files.
- Disconnecting from a c-tree V4 database.
- Performing general utility functions - includes retrieving extended error messages, comparing data for equality, etc.
- Processing transactions - includes logging, committing, and rollback.
- Process optimization - includes processing joins and filters.

4.2 Driver Types

The c-tree ODBC Driver - c-tree V4 Edition is available in the 32-bit implementation designed for Windows 95/98 and Windows NT/2000 in the following configurations:

End-User Driver

This Driver is configured by default for read-only access but can be enabled to support database writes for files opened in EXCLUSIVE mode.

End-User Read-Only Driver

This Driver does NOT provide the ability to perform database writes or to create tables. It is designed for read-only access to existing c-tree V4 files.

4.3 Requirements

The c-tree ODBC Driver operates on Windows 95 or higher with at least 4 MB of RAM. 2 MB of available hard drive space is required. Your application vendor must also provided meta-data for the data files in the form of an external parameter file.

4.4 ODBC Compliance

The ODBC standard has three levels of compliance: core, level 1 and level 2. The c-tree ODBC Driver fully supports Core and Level 1, and supports some ODBC Level 2 functions:

Core Functions	Level 1 Functions	Level 2 Functions
SQLAllocConnect	SQLBindParameter	SQLBrowseConnect
SQLAllocEnv	SQLColumns	SQLDataSources
SQLAllocStmnt	SQLDriverConnect	SQLDescribeParam
SQLBindCol	SQLGetConnectOption	SQLDrivers
SQLCancel	SQLGetData	SQLMoreResults
SQLColAttributes	SQLGetFunctions	SQLNativeSql
SQLConnect	SQLGetInfo	SQLNumParams
SQLDescribeCol	SQLGetStmntOption	SQLPrimaryKey
SQLDisconnect	SQLGetTypeInfo	
SQLError	SQLParamData	
SQLExecDirect	SQLPutData	
SQLExecute	SQLSetConnectOption	
SQLFetch	SQLSetStmntOption	
SQLFreeConnect	SQLSpecialColumns	
SQLFreeEnv	SQLStatistics	
SQLFreeStmnt	SQLTables	
SQLGetCursorName		
SQLNumResultCols		
SQLPrepare		
SQLRowCount		
SQLSetCursorName		
SQLTransact		

The level of functionality supported by FairCom is consistent with other ODBC drivers in the marketplace and meets or exceeds the needs of most popular ODBC compliant applications.

4.5 SQL Conformance

The c-tree ODBC Driver fully supports the minimum SQL grammar and a portion of the core and extended SQL grammar requirements of the ODBC standard, as shown in the following tables:

Minimum SQL Grammar

The minimum Grammar requirements (fully met by the c-tree ODBC Driver) are as follows:

- Create Table
- Delete (searched)
- Drop Table
- Insert
- Select
- Update (searched)

The following tables provides some examples of the minimum SQL grammar supported by the c-tree ODBC Driver:

Grammar	Examples	Comments
CREATE TABLE	CREATE TABLE sal (emp_id integer, name char(50), salary float, hire_date date) CREATE TABLE emp (emp_id integer NOT NULL, PRIMARY KEY (emp_id))	Column constraint definitions supported: NOT NULL. Table constraint definitions supported: UNIQUE and PRIMARY KEY DEFAULT. Default-value is not supported.
DELETE	DELETE FROM sal WHERE name = 'John Smith'	
DROP TABLE	DROP TABLE sal	[CASCADE RESTRICT] is not supported.
INSERT	INSERT INTO sal VALUES (34086, 'Fred Black', 45000.00, '1992-05-25')	
SELECT	SELECT * FROM sal SELECT emp.emp_id, sal.salary FROM emp, sal WHERE emp.emp_id = sal.emp_id	
UPDATE	UPDATE sal SET salary = 35000.00 WHERE emp_id = 25089	

Core SQL Grammar

The Core Grammar supported by the FairCom ODBC Driver is as follows:

- Create Index
- Drop Index
- Select
 - Approximate numeric literal
 - Between predicate
 - Correlation name
 - Exact numeric literal
 - IN predicate
 - Set function

- Subqueries

The following tables provides some examples of the core SQL grammar supported by the FairCom ODBC Driver:

Grammar	Examples	Comments
CREATE INDEX	CREATE INDEX empidx ON emp (emp_id, emp_name)	To designate a key as a primary key, FairCom supports the UNIQUE option.
CREATE VIEW	CREATE VIEW vv_sal (v_col1, v_col2) AS SELECT emp_id, name FROM sal	The column list is optional
DROP INDEX	DROP INDEX emp.empuniq	
DROP VIEW	DROP VIEW vw_sal	[CASCADE RESTRICT] is not supported.
SELECT	SELECT COUNT(emp_id), dept FROM mgrs GROUP BY dept HAVING dept > 15	In addition to supporting an order by on a column-list, as specified in the ODBC Programmer's Reference, FairCom has extended the syntax to support an order by on an expression-list or on any expression in a group by expression-list. For example: SELECT * FROM emp ORDER BY a+b,c+d,e This causes the result table to be ordered by three expressions: a+b, c+d, and e. If the expression is a positive integer literal, then that literal will be interpreted as the number of the column in the result set and ordering will be done on that column. No ordering is allowed on set functions or an expression that contains a set function.

The following types of subqueries are supported: comparison, exists, quantified, in, and correlated. Order by clauses are not allowed in a subquery clause.

Grammar	Examples	Comments
approximate-numeric-literal	SELECT * FROM results WHERE quotient = -4.5E-2	
between-predicate	SELECT c1 FROM emp WHERE emp_id BETWEEN 10000 AND 20000	The syntax expr1 BETWEEN expr2 AND expr3 returns TRUE if expr1 >= expr2 and expr1 <= expr3. expr2 and expr3 may be dynamic parameters (e.g., SELECT * FROM emp WHERE emp_id BETWEEN ? and ?).
correlation-name	SELECT * FROM emp t1, addr t2 WHERE t1.emp_id = t2.emp_id	FairCom supports both table and column correlation names.
exact-numeric-literal	INSERT INTO cars (car_no, price) VALUES (49042, 49999.99) SELECT * FROM numtbl WHERE c1 = -208.6543189	
in-predicate	SELECT * from colors WHERE color IN ('red', 'blue', 'green')	

Grammar	Examples	Comments
set-function	<pre>SELECT COUNT (a+b) FROM q SELECT MIN (salary) FROM emp</pre>	<p>MIN(expr), MAX(expr), AVG(expr), SUM(expr), COUNT(*), and COUNT(expr) are supported. COUNT(expr) counts all non-NULL values for an expression across a predicate. The following example counts all the rows in q where a+b does not equal NULL:</p> <pre>SELECT COUNT (a+b) FROM q</pre>
inner join syntax	<pre>SELECT * FROM mytableA, mytableB WHERE myColA = myColB SELECT * FROM mytableA, mytableB INNER JOIN myColA = myColB</pre>	<p>These two statements are considered identical.</p>

Extended SQL Grammar

The Extended Grammar supported by the c-tree ODBC Driver is as follows:

- Left Outer Join (two or three-table outer join)
- Select
 - date arithmetic
 - date literal
 - time literal
 - timestamp literal
- Unions
- Use

The following table provides some examples of the extended SQL grammar supported by the c-tree ODBC Driver:

Grammar	Examples	Comments
LEFT OUTER JOIN	<p>Two-table outer join:</p> <pre>SELECT * FROM emp LEFT OUTER JOIN dept ON emp.deptID = dept.deptID</pre> <p>Three-table outer join:</p> <pre>SELECT * FROM (emp LEFT OUTER JOIN dept ON emp.deptID = dept.deptID) LEFT OUTER JOIN addr ON emp.empID = addr.empID</pre> <p>Embedded in vendor strings:</p> <pre>SELECT t1.deptno, ename FROM {oj emp t2 LEFT OUTER JOIN dept t1 ON t2.deptno = t1.deptno}</pre>	<p>FairCom supports two-table outer joins.</p> <p>In addition to simple two-table outer joins, FairCom supports n- way nested outer joins.</p> <p>The outer join may or may not be embedded in a vendor string. If a vendor string is used, the ODBC driver will strip it off and parse the actual outer join text.</p>
UNION	<pre>SELECT name, status FROM tech_staff UNION SELECT name, status FROM adm_staff</pre>	<p>UNION eliminates duplicate rows.</p>

Grammar	Examples	Comments
UNION ALL	SELECT name, status FROM tech_staff UNION ALL SELECT name, status FROM adm_staff	UNION ALL preserves duplicate rows.
date-literal	SELECT * FROM emp WHERE hire_date < '1992-02-02' SELECT * FROM emp WHERE hire_date < {d '1992-02-02'}	FairCom supports the following date literal format: 'yyyy-mm-dd'. Dates may be in the range of year 0 to 9999. Date constants may be expressed in SQL statements as a character string or embedded in a vendor string. FairCom treats the character string representation as a string of type SQL_CHAR and the vendor string representation as a value of type SQL_DATE. This becomes important when conversions are attempted. For example, CONVERT({d '1992-02-02'}, SQL_TIMESTAMP) is valid, whereas CONVERT('1992-02-02', SQL_TIMESTAMP) returns an invalid SQL_TIMESTAMP value.
time-literal	SELECT * FROM bday WHERE btime = '10:04:29' SELECT * FROM bday WHERE btime = {t '10:04:29'}	FairCom supports the following time literal form: 'hh:mm:ss'. Time constants may be expressed in SQL statements as a character string or embedded in a vendor string. FairCom treats the character string representation as a string of type SQL_CHAR and the vendor string representation as a value of type SQL_TIME.
timestamp-literal	SELECT * FROM bday WHERE btime = '1965-08-25 05:25:00' SELECT * FROM bday WHERE btime={ts '1965-08-25 05:25:00'}	FairCom supports the following timestamp literal format: 'yyyy-mm-dd hh:mm:ss'. Timestamp constants may be expressed in SQL statements as a character string or embedded in a vendor string. FairCom treats the character string representation as a string of type SQL_CHAR and the vendor string representation as a value of type SQL_TIMESTAMP.
date arithmetic	SELECT * FROM inv WHERE inv_date > '1993-01-01' AND inv_date < {d '1993-01-01'} + 30 SELECT * FROM pay WHERE pay_date - inv_date > 30	FairCom supports adding or subtracting an integer from a date where the integer is the number of days to add or subtract, and the date is embedded in a vendor string. (This is equivalent to executing a CONVERT on the date.) FairCom also supports subtracting one date from another to yield a number of days.
extended predicates	{pred contains, col1, 'text'}	Uses extended vendor string syntax.

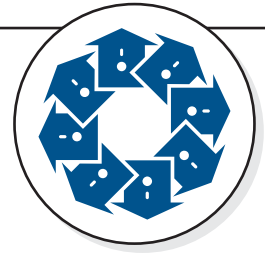
4.6 Driver Constraints

The following limits apply to the c-tree ODBC Driver:

Explanation	Value
Concurrently open tables:	
16-bit multi-user non-server driver	305
All others	405
Number of rows:	2 billion
Number of columns in a query:	256
Size of a column:	2 gigabytes
Number of connections:	limited by memory
Table, index and column name length:	64 characters
Table qualifier:	68 characters
Maximum characters in a literal:	1000
Maximum columns in a CREATE TABLE statement:	256
Maximum ANDed predicates. This example uses three ANDed predicates: SELECT * FROM abc WHERE c1 AND c2 AND c3 AND c4	300
Number of joined tables:	limited by memory

4.7 Licensing

The c-tree ODBC Driver is licensed for use on a single computer. For pricing information, please contact your software provider or nearest FairCom office.



Advanced Topics

This chapter describes advanced topics, such as performance optimization, files installed, and changes made during the install.

5.1 Performance Optimization

- The fewer file names in a given database, the faster the data access will be. Since the c-tree ODBC Driver can support multiple database names, it is usually advantageous to create multiple views into the database. See [“Data Dictionary Creation - Import Method”](#) for further information.
- The fewer records per data file, the faster the data access will be.

5.2 Data Dictionary Creation - Import Method

The FairCom Data Dictionary can be created with the FairCom import utility, **import.exe**, using a text-based script with the following layout:

```
DATABASE d_name
```

```
TABLE alias_name1(\path\file_name1)  
TABLE alias_name2(\path\file_name2)
```

- *d_name* is an arbitrary database name or view over the data files (i.e., *FAIRCOM.DB* is the database name used in the tutorial).
- *alias_name1* and *alias_name2* are arbitrary names for the data files residing on disk (i.e., *CUSTOMERS* and *ORDERS* are sample alias names from the sample import script, *SAMPLE.TXT*).
- *path1* and *path2* are either fully qualified paths or relative paths to the current logical drive (i.e., *IC:\FAIRCOM\ODBC\V4\SAMPLE* from the tutorial import script, *SAMPLE.TXT*).
- *file_name1* and *file_name2* are the actual file names residing on disk (i.e., *CUSTMAST.OTP* and *CUSTORDR.OTP* from *SAMPLE.TXT*).

The example below contains one database view (*FAIRCOM.DB*) and four file members (*CUSTOMERS*, *ORDERS*, *ORDER_DETAIL*, and *ITEMS*).

```
DATABASE FAIRCOM.DB
```

```
TABLE CUSTOMERS(c:\faircom\odbc\V4\sample\CUSTMAST.OTP)  
TABLE ORDERS(c:\faircom\odbc\V4\sample\CUSTORDR.OTP)  
TABLE ORDER_DETAIL(c:\faircom\odbc\V4\sample\ORDRITEM.OTP)  
TABLE ITEMS(c:\faircom\odbc\V4\sample\ITEMMAST.OTP)
```

Import Script

The import utility (**import.exe**) is a DOS-based, command line utility executed from the MS-DOS prompt. To create the dictionary for the ODBC tutorial, execute the import utility by specifying the path to **import.exe** located in the \FAIRCOM\ODBC\V4 directory and pass the import script on the command line as follows:

```
c:\faircom\odbc\v4\import sample.txt
```

The Data Dictionary file, *CTSYSCAT.FCS*, is created in the root directory of your local drive. For example, if \FAIRCOM\ODBC\V4 is on drive C, *CTSYSCAT.FCS* is created in the C:\ directory. Move this file to a new location by copying it to the desired location and specifying the path in the ODBC Setup dialog box.

The import utility overwrites any existing *CTSYSCAT.FCS* file in the root directory of the local drive.

When building your own script, it is recommended to fully specify the file paths (i.e., include the logical drive letter in addition to the path, C:\FAIRCOM\ODBC\V4\SAMPLE as opposed to \FAIRCOM\ODBC\V4\SAMPLE).

Wildcard

Rather than building your own script, **import.exe** can be passed a wild card to build the script. Execute the import utility by specifying the path to **import.exe** and pass the parameters “-f *.OTP” as follows:

```
c:\faircom\odbc\v4\import -f *.OTP
```

Passing “-f *.OTP” tells **import.exe** to create the Data Dictionary (*CTSYSCAT.FCS*) and add all of the files with the “.OTP” extension located in the current directory. When the import utility sees the -f switch, it also creates a text-based import file, *IMPORT.TXT*, listing all the data files located in the current directory. This script can be used by **import.exe** to re-create the dictionary.

Multiple Database Script

Splitting application files into distinct database groups enhances performance of the ODBC Driver. This is easily done with **import.exe**.

```
DATABASE FAIRCOM.DB
```

```
TABLE CUSTOMERS(c:\faircom\odbc\v4\sample\CUSTMAST.OTP)  
TABLE ORDERS(c:\faircom\odbc\v4\sample\CUSTORDR.OTP)  
TABLE ORDER_DETAIL(c:\faircom\odbc\v4\sample\ORDRITEM.OTP)  
TABLE ITEMS(c:\faircom\odbc\v4\sample\ITEMMAST.OTP)
```

```
DATABASE CUSTOMER.DB
```

```
TABLE CUSTOMERS(c:\faircom\odbc\v4\sample\CUSTMAST.OTP)
```

The above example illustrates adding a new database view (*CUSTOMER.DB*) over the Customer file (*CUSTMAST.OTP*). When this script is passed to **import.exe**, the data dictionary created contains two databases: *FAIRCOM.DB* with four data files; and *CUSTOMER.DB* with one data file. To switch between the databases, change the name specified in the Script Name setting in the ODBC Setup dialog.

5.3 OTP Files

OTP files allow access to c-tree data files without embedded resources. This feature is required for the c-tree ODBC Driver - c-tree V4 Edition, but is usually implemented by the application developer because it requires detailed knowledge of the record layout and data/index file relationships.

OTP File Requirements

To implement the c-tree Drivers without *IFIL* and *DODA* resources, you must:

- Create an o-tree Parameter, OTP, file for each of the application's data files. These are usually located in the directory with the data files or another convenient location.
- Create a DB file listing the OTP files that define the structure of your application's data and index files, or browse for OTP files are described in [“Getting Started with Your Data”](#).

OTP File Organization

OTP files are ASCII text files specifying the characteristics of the data files and indices used in an application program. An OTP file is required for each data file that is not using resources. While these files are similar to ISAM parameter files used by c-tree, OTP and parameter files are not interchangeable.

An OTP file consists of five types of records:

- Data File Description Record;
- Field Description Records;
- Index File Description Records;
- Optional Index Member Records;
- Key Segment Description Records.

Each OTP file begins with one Data File Description Record specifying the number of fields and indices it uses. This is followed by a Field Description Record for each field in the record. For each index there is a group of records beginning with the Index File Description Record or Index Member Record followed by one or more Key Segment Description Records.

The overall organization of the records is shown in the following schematic and reflects the hierarchical relationship among the data files and indices:

OTP File Layout

```
Data File Description Record
  First Field Description Record
  :::
  Last Field Record
  First Index File Description Record
    First Key Segment Description Record
    :::
    Last Key Segment Record
  Optional Index Member Records
    First Key Segment Description Record
    :::
    Last Key Segment Record
  :::
  Last Index File or Index Member Record
    First Key Segment Description Record
    :::
    Last Key Segment Record
```

OTP File Contents

Each field comprising the OTP file records is separated from neighboring fields by one or more “white space” characters: blanks, tabs, or new lines. In the following subsections, if a parameter corresponds to a formal parameter in a c-tree function call, the parameter name is included after the parameter's brief description (e.g., *filnam* in the first position of the data file description record).

Data File Description Record

Each Data File Description Record consists of eight required fields, the first four fields corresponding to parameters of **CreateDataFile()**:

- data file name (*filnam*)
- record length (*datlen*)
- file size extension parameter (*xtdsiz*)
- file mode (*filmod*)
- number of associated index files (*num_idx*)
- number of fields (*#flds*)
- symbolic name (without spaces) of the first field in the data record (*fst fldnam*)
- symbolic name of the last field in the data record (*lst fldnam*)

filnam	datlen	xtdsiz	filmod	num_idx	#flds	fst fldnam	lst fldnam
custordr.dat	24	4096	1	2	5	_o_delflag	co_custnumb

Parameters

- *filnam*
The name of the data file, including its extension and (optional) path. For the c-tree V4.x Driver, *filnam* cannot include a path.
If a path is not specified, the path for the OTP file as defined in the data dictionary script (*VENDOR.DB*) file will be used for the path of the data and index files. The data file name must agree with the operating system's file naming conventions.
- *datlen*
The record length of the file for fixed-length files. For variable-length files this is the size of the fixed-length portion of the record.
- *xtdsiz*
The file extension size. This is the amount by which the file is increased when additional space is needed.
- *filmod*
The c-tree file mode used by your application. See "[filmod Values](#)" for more information.
- *num_idx*
The number of indices associated with this data file. Each such index will be either a separate c-tree file or a member of an index file.
- *#flds*
The number of fields contained in the data record, which should match the number of Field Description Records in your OTP file.
- *fst fldnam*
The symbolic name of the first field entered in the field section.
- *lst fldnam*
The symbolic name of the last field entered in the field section.

Field Description Record

field name	offset	data type	length
co_ordrdate	4	75	4

Parameters

- *field name*
The symbolic name of the field used to identify a region of the data record.

- *offset*

The physical offset from beginning of the record (base 0) at which this field starts.

- *data type*

A numeric representation of the data type contained in this field, from the table below.

Data Type Values	Data Type Descriptions
16	1 byte signed character.
24	1 byte unsigned character.
33	2 byte signed integer.
41	2 byte unsigned integer.
51	4 byte signed integer.
59	4 byte unsigned integer.
67	4 byte integer representing money in pennies.
75	a date value represented as a 4 byte long.
83	a time value (seconds since midnight) as a long.
91	single float.
103	double float.
144	fixed length string.
146	null delimited varying length string.

- *length*

The field length in bytes.

Index File Description Record

Each Index File record is comprised of eleven required fields, the first seven fields corresponding to parameters of **CreateIndexFile()**:

- index file name (*filnam*)
- key length (*keylen*)
- key type (*keytyp*)
- duplicate flag (*keydup*)
- number of additional index file members (*nomemb*)
- file size extension parameter (*xtdsiz*)
- file mode (*filmod*)
- NULL key flag (*nulkey*)
- empty character (*empchr*)
- number of key segments (*numseg*)
- symbolic index name (*symname*)

```

filnam      keylen keytyp keydup nomemb xtdsiz filmod nulkey empchr numseg
custordr.idx 6      4      0      1      4096  1      1      32     1  custordr.1
  
```

Parameters

- *filnam*
The physical index file name without a path. The index file name must agree with the operating system's file naming conventions.
- *keylen*
The length of the key in this index.
- *keytyp*
The key type which can be:

0	fixed length key	8	padding compression
4	leading character compression	12	4 & 8 combined

- *keydup*
A value of 1 allows duplicates. A value of 0 does not allow duplicates. If duplicates are allowed, the last four bytes of the key value are reserved for the four bytes of the associated data record position. For example, a key length of twelve bytes combined with duplicate values results in eight bytes for the actual key value and four bytes for the tie-breaking data record position.
- *nomemb*
The number of additional index members contained in this index not counting the host index (physical file).
- *xtdsiz*
As with data files this is the amount by which the file will be extended when additional space is needed.
- *filmod*
c-tree file mode used to open the file.
- *nulkey*
When set to 1, key values containing only the empty character, *empchr*, are excluded from the index. When set to 0 there is no check for NULL keys.
- *empchr*
The character determining if a key entry is a NULL key.
- *numseg*
The number of segments comprising the key. These are described below in the "[Key Segment Description Record](#)".
- *symbolic name*
A symbolic name used to identify this particular index.

Optional Index Member Record

It is not necessary for each index associated with a data file to be in a separate file. Indices can be combined in the same physical file. If the *nomemb* parameter of an Index File Description Record is greater than zero, *nomemb* Index Member Records must follow the Index File Description Record. For example, if a data file has three indices, and if the second index has one additional member, (*nomemb* equals one), then there will be a total of two Index Description Records and one Index Member Record.

The Index Member Record is a subset of the Index File Description Record composed of seven required fields:

- key length (*keylen*)
- key type (*keytyp*)
- duplicate flag (*keydup*)
- NULL key flag (*nulkey*)

- empty character (*empchr*)
- number of key segments (*numseg*)
- symbolic index name (*symname*)

These fields have the same interpretation as the corresponding fields in the Index Description Record.

keylen	keytyp	keydup	nulkey	empchr	numseg	symname
8	4	1	1	32	1	custodr.2

Parameters

- *keylen*
The length of the key.
- *keytyp*
The type of the key as described for the host index above.
- *keydup*
1=allow duplicates, 0=do not allow duplicates.
- *nulkey*
1=check for NULL keys, 0=do not check for NULL keys.
- *empchr*
The character value used for the NULL key check.
- *numseg*
The number of segments comprising the key.
- *symbolic name*
The symbolic name to be used to identify this index.

Key Segment Description Record

Each key value is composed of one or more segments. A segment is simply a sequence of bytes from a specified offset within the data record. Each Key Segment Description record is comprised of three fields:

- segment position (*soffset*)
 - segment length (*slength*)
 - segment mode (*segmode*)
- | | | |
|-----------|-----------|-----------|
| (soffset) | (slength) | (segmode) |
| 19 | 4 | 2 |

Parameters

- *soffset*
The offset, in bytes, from the beginning of the record. When using segment types 4 or 5, represents a field number, starting at zero, in the variable-length portion of a variable-length record, not counting fields from the fixed-length portion.
- *slength*
The length of this segment of the key from 'offset'.
- *segmode*
The type of data that makes up this segment, as shown in the table in ["segmode Values"](#).

filmod Values

At file creation and open, the *filmod* parameter is used to specify important file characteristics. Due to the number of different options it is not practical to show you all possible combinations.

When you are asked to supply the *film* parameter you can combine the various options by simply adding them together (e.g., *ctSHARED* with *ctVLENGTH* = 1 + 4, or 5).

Certain options are mutually exclusive, so they should not be used together. Only one option should be used from each of the following groups:

- *ctVIRTUAL* or *ctPERMANENT*
- *ctEXCLUSIVE*, *ctSHARED* or *ctREADFIL*
- *ctFIXED* or *ctVLENGTH*
- *ctPREIMG* or *ctTRNLOG*

The following table defines the values for the file modes. You should always use the defined name for the values if possible. However, you need the numeric values if you are using an ISAM parameter file. These values can be found in *ctport.h*.

File mode	Decimal value	Hexadecimal value
<i>ctEXCLUSIVE</i>	0	0x0000
<i>ctSHARED</i>	1	0x0001
<i>ctVIRTUAL</i>	0	0x0000
<i>ctPERMANENT</i>	2	0x0002
<i>ctFIXED</i>	0	0x0000
<i>ctVLENGTH</i>	4	0x0004
<i>ctREADFIL</i>	8	0x0008
<i>ctPREIMG</i>	16	0x0010
<i>ctTRNLOG</i>	48	0x0030
<i>ctWRITETHRU</i>	64	0x0040
<i>ctCHECKLOCK</i>	128	0x0080
<i>ctDUPCHANEL</i>	256	0x0100
<i>ctSUPERFILE</i>	512	0x0200
<i>ctCHECKREAD</i>	1024	0x0400
<i>ctDISABLERES</i>	2048	0x0800
<i>ctMIRROR_SKP</i>	8192	0x2000
<i>ctOPENCRIPT</i>	16384	0x4000
<i>ctLOGIDX</i>	32768	0x8000

segmode Values

The segment position specifies the location of the key segment within the data record. This can be specified in one of three ways, as determined by the segment mode value (as discussed later);

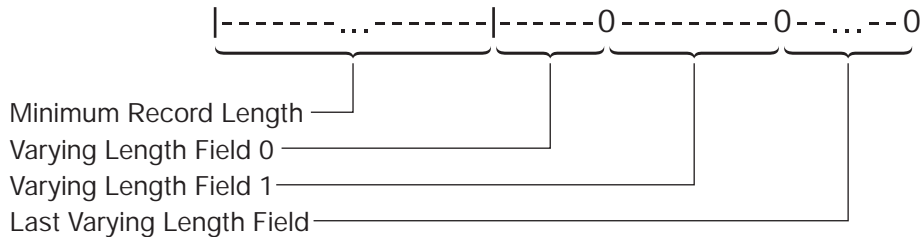
1. Absolute byte offset. The number of bytes from the beginning of the data record to the first byte of the key segment. The smallest offset, zero, means that the segment begins with the first byte of the record. If a segment begins with the nth byte of the record, its offset value should be n minus one.

This option works with fixed length fields, which have known offsets.

2. Relative field number. The segment position is treated as a field number, not a byte offset. This applies only to variable-length data records. With variable-length records, all of the fields in the variable-length portion are variable-length fields terminated with a delimiter. This delimiter is typically the NULL character, unless the default is changed by **SetVariableBytes()**. A segment position of zero corresponds to the first variable-length field, beginning at the first byte beyond the variable-length file's fixed-length portion. When a variable-length file is created, the record length parameter is treated as the minimum record length, which corresponds to the fixed-length portion of the record.

This option works with variable-length string types using NULL terminators.

The following schematic demonstrates how varying-length fields are numbered. The delimiter is shown as a NULL byte, but can be any delimiter you set:



0 Signifies a Null Byte

The following table lists the valid key segment mode values.

Mode value	Symbolic constant	Segment Position Interpretation	Explanation
0	<i>REGSEG</i>	Absolute byte offset	No transformation.
1	<i>INTSEG</i>	Absolute byte offset	Unsigned integer/long.
2	<i>UREGSEG</i>	Absolute byte offset	Lower case letters converted to upper case.
3	<i>SRLSEG</i>	Absolute byte offset	Automatic 4 byte sequence number.
4	<i>VARSEG</i>	Relative field #	No transformation, pad to length.
5	<i>UVARSEG</i>	Relative field #	Lower case letters are converted to upper case, pad to length.
8	<i>SGNSEG</i>	Absolute byte offset	Signed integer/long.
9	<i>FLTSEG</i>	Absolute byte offset	Floating point (float or double).

5.4 File Locations

The installation places several files on your system. The Microsoft and FairCom supplied files are installed in the Windows or `\windows\system` directories. The c-tree Sample data files are installed in the FairCom created `C:\FAIRCOM\ODBC\4` directory or the directory specified during installation.

32-bit Microsoft Supplied Files

<i>ODBC32.DLL</i>	32-bit thinking Driver Manager
<i>ODBCCP32.CPL</i>	ODBC C Panel
<i>ODBCINT.DLL</i>	Language DLL
<i>ODBCAD32.EXE</i>	32-bit Administrator program
<i>ODBCCP32.DLL</i>	32-bit installer DLL
<i>ODBCCR32.DLL</i>	32-bit Cursor library
<i>ODBCINST.HLP</i>	Installer help file
<i>ODBCINST.CNT</i>	Installer help file table of contents
<i>CTL3D95.DLL</i>	Windows 95 three dimensional style library
<i>CTL3DNT.DLL</i>	Windows NT three dimensional style library
<i>MSVCRT20.DLL</i>	C run-time library

32-bit FairCom Supplied Files

<i>OTC_ODBC.DLL</i>	c-tree 32-bit ODBC Driver
<i>OTC_TREE.DLL</i>	c-tree 32-bit c-tree V4 interface
<i>SIMSPY32.DLL</i>	ODBC Driver spy debug DLL
<i>OTC_CNFG.DLL</i>	c-tree 32-bit configuration DLL

FairCom Supplied Sample Files

<i>FAIRCOM.DB</i>	Script for tutorial, creates dictionary for sample files
<i>SAMPLE.TXT</i>	Script for creating the tutorial data dictionary
<i>IMPORT.EXE</i>	FairCom Data Dictionary creation utility
<i>CUSTMAST.OTP</i>	Tutorial customer master o-tree parameter file
<i>CUSTMAST.DAT</i>	Tutorial customer master data file
<i>CUSTMAST.IDX</i>	Tutorial customer master index file

<i>CUSTORDR.OTP</i>	Tutorial customer order o-tree parameter file
<i>CUSTORDR.DAT</i>	Tutorial customer order data file
<i>CUSTORDR.IDX</i>	Tutorial customer order index file
<i>ITEMMAST.OTP</i>	Tutorial item master o-tree parameter file
<i>ITEMMAST.DAT</i>	Tutorial item master data file
<i>ITEMMAST.IDX</i>	Tutorial item master index file
<i>ORDRITEM.OTP</i>	Tutorial order item o-tree parameter file
<i>ORDRITEM.DAT</i>	Tutorial order item data file
<i>ORDRITEM.IDX</i>	Tutorial order item index file

5.5 Registry Options

The c-tree ODBC Driver sets several Registry keys. Use the [ConfigDSN\(\)](#) function to programmatically configure the c-tree ODBC Driver Data Source. Most c-tree Driver subkeys correspond to configuration entries shown in the table below. The ForceClose entry is only adjustable in the Registry. All the subkeys below are in the key \\HKEY_CURRENT_USER\\Software\\ODBC\\ODBC.INI\\ FairCom c-tree Driver by default unless you created your own DSN.

Registry subkey	Configuration option
Alignment	Alignment
AllowUpdates	Allow Updates
CaseSensitive	Case-insensitive string Comparison
ColumnsOrderedBy	Column Order
DataBuffers	Data Buffer Size
DataDictionary	Script Name
DBQ	Data Dictionary Path
DebugCtree	c-tree Plus Debug
DebugIndex	Debug Indexes
Description	Description
EnableFPUExceptions	See below
Files	Number of Files
ForceClose	Not a configuration option
GuestLogin	Logon to Server as Guest
IndexBuffers	Index Buffer Size
MaxColSupport	Table's max number of columns

Registry subkey	Configuration option
OEMtoANSI	OEM to ANSI Data Conversion
Protocol	Driver Type
Sectors	Sector Size
ServerName	Server Name
SpecialTypes	Special Data Type Conversion DLL Name
StringDataPadding	Data Padding
StringKeyPadding	Key Padding

Force Close Option

For performance reasons, the c-tree ODBC Driver keeps data and index files open until the database is changed or the connection is closed. It is possible to force the Driver close files when the SQL engine requests files to be closed. To do so, create a registry key named ForceClose, as shown above, and set its value to "Yes" to enable this option.

Adding a Shareable DSN

The Registry feature allows the creation and use of shareable file DSNs by the 32bit c-tree ODBC Driver. The Driver creates unshareable file DSNs automatically when creating user or system DSNs.

To create a shareable file DSN:

1. Start the 32bit ODBC Administrator.
2. Select the **File DSN** tab.
3. Click the **Add...** button.
4. Select the ODBC driver for which you wish to create a file DSN (either "FairCom 32bit ODBC Driver" or "FairCom c-tree Driver").
5. Click the **Advanced...** button and enter the desired driver-specific keywords (from the list above).
Both DRIVER= (driver name) and DBQ= (data dictionary path) are required. For example:
`DRIVER={FairCom 32bit ODBC Driver}`
`DBQ=c:\faircom\odbc\32bit`
6. Click the **OK** button.
7. Click the **Next** button.
8. Enter the name of your new file DSN.
9. Click the **Next** button.
10. Click the **Finish** button.

If after you follow the above steps, the ODBC administrator returns the error "A connection could not be made using the file data source parameters entered. Save non-verified file DSN?", you probably omitted needed parameters or specified invalid parameter values (see step 5 above). Click **Cancel** and start over at step 1.

After a file DSN is successfully created in this manner, it can't be configured from the ODBC Administrator, however, it is a plain text file that can be edited to change its settings.

This type of file DSN is known as a shareable file DSN - if you arrange for it to be shared by multiple machines (each with the c-tree ODBC Driver installed), it will serve as an ODBC data source that is identically configured on all of these machines.

Optionally Disable Floating Point Unit Exceptions

By default, the c-tree ODBC Driver enables the divide by zero FPU (floating point unit) exception so that it can detect when a divide by zero error occurs. A customer requested the ability to avoid enabling this exception to work around a problem with Microsoft's .Net Font class constructor, which raises an ArithmeticException exception if floating point unit exceptions have been enabled.

The c-tree ODBC Driver checks the registry for a string value named EnableFPUEXceptions under the registry subkey:

```
HKEY_CURRENT_USER\Software\ODBC\ODBC.INI\FairCom 32bit Driver
```

If this string value exists and is set to No, the c-tree ODBC Driver does not enable FPU exceptions.

5.6 DSN-less ODBC connections

The c-tree ODBC Driver can be used without a DSN (data source name) by specifying the ODBC Driver Name rather than a DSN in the connection string, along with the desired options:

```
DRIVER=FairCom X ODBC Driver;keyword1=value1;...;keywordN=valueN;
```

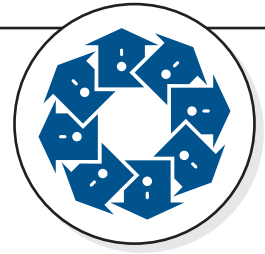
The DRIVER specification above must be either "FairCom 32bit ODBC Driver" (for the c-tree Plus Driver) or "FairCom c-tree ODBC Driver" (for the c-tree V4 Driver). The Driver supports the following option keywords in the connection string.

Keyword	Supported values (default value listed first)
Alignment	Default, 1, 2, 4, 8
AllowUpdates	No, Yes
CaseSensitive	Yes, No
ColumnOrder	Position, Name
CommEncryption	No, Yes
DataBuffers	64
DataDictionary	FAIRCOM.DB
DBQ	c:\otree
DebugCtree	No, Yes
DebugIndex	No, Yes
Files	395
GuestLogin	Yes, No
IndexBuffers	64
MaxColSupport	256

Keyword	Supported values (default value listed first)
OEMtoANSI	No, Yes
Protocol	MultiUser, TCPIP, NETBIOS, SPX
Sectors	16
ServerName	FAIRCOMS
SpecialTypes	None
StringDataPad	Zeros, Spaces, NulTermSpaces
StringKeyPad	Default, Zeros, Spaces

These keywords correspond to the registry settings described in [“Registry Options”](#), with the following exceptions:

- ColumnOrder corresponds to the ColumnsOrderedBy registry setting.
- StringDataPad corresponds to the StringDataPadding registry setting.
- StringKeyPad corresponds to the StringKeyPadding registry setting.
- There are no supported connection string keywords for the ForceClose and KanjiConvert registry settings.
- The MaxRows setting can be used to limit the number of accessible rows, though there is no corresponding registry setting.



Errors

6.1 DBOPEN DICTDBGETBYNAME(FAIRCOM.DB)=101

The specified database, *FAIRCOM.DB* in this message, could not be located. This error message is typically seen if Debug is enabled in the c-tree ODBC Setup dialog box.

1. The Data Dictionary Path was not properly specified. See [“ODBC Setup Dialog Options”](#).
2. The FairCom Data Dictionary has not been created.

6.2 No such database

1. If the FairCom Data Dictionary has been created, ensure the Data Dictionary Path in the ODBC Setup dialog box points to *CTSYSCAT.FCS*.
2. If the path is correct, re-create the FairCom Data Dictionary (*CTSYSCAT.FCS*).

6.3 No table names appear in the Select Table dialog box

Be sure the table name paths specified in the dictionary create script (i.e., *FAIRCOM.DB* for the ODBC tutorial) or the import script (i.e., *SAMPLE.TXT* for the ODBC tutorial) point to where the tables (files) are located.

6.4 Not able to open *.FCS file

1. *CTSYSCAT.FCS* not created. Create the Data Dictionary.
2. *CTSYSCAT.FCS* not in the proper path. This file must reside in Data Dictionary Path as specified in the c-tree ODBC setup window.

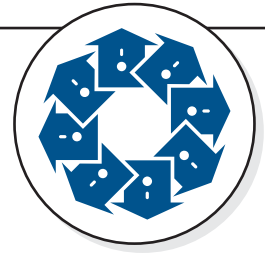
6.5 Specified driver could not be loaded

Possible solutions:

- Driver DLL may be missing. Reinstall the Driver.
- Driver DLL may be corrupt. Reinstall the Driver.
- Driver DLL not in *windows\system* directory. Place Driver DLL in the *windows\system directory*.

6.6 System Catalog's Files Need to be Rebuilt

Click **OK** to delete and recreate *CTSYSCAT.FCS* using the data dictionary script specified in the c-tree ODBC Driver setup dialog box. Otherwise, *CTSYSCAT.FCS* will not be touched and the connection attempt will fail.



Glossary

The following terms are used throughout this guide.

32-bit

An application designed to operate only on 32-bit operating systems, such as Windows 95/98 and Windows NT. The theory of 32-bit applications is that they are typically superior since they utilize the underlying power of the newer operating systems and hardware available today.

c-tree®

FairCom's legacy data engine. The parent product for c-tree Plus.

c-tree Plus®

FairCom's data engine. Used in many applications and embedded systems to manage data.

column

Synonymous with the definition of field. In the relational model, column is used in place of the term field. Column and field are used interchangeably in this guide.

CTSYSCAT.FCS

The FairCom Data Dictionary file.

data dictionary

A list of the data files used in various databases.

directory

A location where files are stored on disk. A directory can be thought of as a drawer in a file cabinet. Each file folder within the drawer can be thought of as a separate file, or collection of like information.

FAIRCOM.DB

The default script file used to create the FairCom Data Dictionary.

field

A specific piece of information stored within a record. Many fields are commonly stored in one record. For example, a record storing a customer address might contain fields for the customer's name, address, city, state, zip, and phone number. This example therefore contains 6 fields in each record.

file

A collection of like information, referred to as records. See the definitions for directory, record and table for further information.

index file

A special type of file that provides a mechanism for performing fast data retrieval.

record

A piece of information stored within a data file. Expanding on the file cabinet example used in the directory definition, each piece of paper found within a file folder can be thought of as a record. A record is a unique piece of information similar to other pieces of information (papers) within the file folder.

row

Synonymous with definition of record. In the relational model, row is used in place of the term record. Row and record are used interchangeably in this guide.

segment

A piece of information (field) stored within an index file. An index can be made up of many fields or even portions of fields. Each field or portion of a field stored within an index is called a segment.

table

A term synonymous with the definition of file. In the relational model, table is used in place of file. Table and file are used interchangeably in this guide.

unique index

An index that does not accept duplicate records, indicating each value stored in the index is unique. An index defined over a customer social security number field would typically be a unique index. If the index supports duplicate records, it is said to be non-unique or duplicate allowed. An index defined over one field, last name, would most likely accept duplicate records since it may be common to have more than one customer with the last name of Smith.

VENDOR.DB

A script file used to build the data dictionary for a vendor's application data files.

Index

3	
32-bit FairCom Supplied Files.....	34
32-bit Microsoft Supplied Files.....	34
A	
Adding a Shareable DSN.....	36
Adding Files to the Dictionary.....	14
Advanced topics.....	25
create data dictionary.....	25
file locations.....	34
OTP files.....	26
performance optimization.....	25
Advanced Topics.....	7, 25
B	
Browse method.....	8
Browse Method.....	7, 8
C	
Compliance	
levels.....	18
Constraints.....	22
Conversion	
process.....	17
Core SQL Grammar.....	19
ctree ODBC Driver Purpose.....	17
ctree ODBC Driver Technical Details.....	17
D	
Data	
file description record.....	28
Data dictionary.....	14
add files.....	14
create.....	25
error.....	39
path.....	11
steps to create.....	9
Data Dictionary Creation - Import Method.....	10, 25
Data File Description Record.....	28
Database	
multiple.....	14
multiple references.....	14
multiple script.....	26
DBOPEN	
DICTDBGETBYNAME(FAIRCOM.DB)=101.....	39
Defining Multiple Databases.....	14
Driver	
compliance.....	18
constraints.....	22
licensing.....	23
purpose.....	17
requirements.....	18
SQL conformance.....	18
types.....	17
Driver Constraints.....	22
Driver types.....	17
end-user.....	17
end-user read-only.....	17
Driver Types.....	12, 17
DSN-less ODBC connections.....	37
E	
Errors.....	39
driver not loaded.....	39
no such database.....	39
no table name.....	39
not able to open .fcs file.....	39
rebuild system catalog file.....	40
unable to locate database.....	39
Extended SQL Grammar.....	21
F	
FairCom Data Dictionary - In Depth.....	14
FairCom Supplied Sample Files.....	34
FAIRCOM TYPOGRAPHICAL CONVENTIONS.....	v
Field description record.....	28
Field Description Record.....	28
File	
data description record.....	28
locations.....	34
OTP.....	26
File Locations.....	34
filmod values.....	31
filmod Values.....	28, 31
Force Close Option.....	36
G	
Generic ODBC Compliant Application Notes.....	14
Getting Started with Your Data.....	7, 27
Glossary.....	41
I	
Import	
method.....	25
script.....	26
Import Script.....	26
Index	
file description record.....	29
member record.....	30
Index File Description Record.....	29
Installation.....	1
vendor supplied script.....	7
Installing a Vendor Supplied Script.....	7
K	
Key Segment Description Record.....	30, 31
Keys	
segment mode.....	32
segment position.....	32

L			
Licensing	23		
Limitation	22		
M			
Minimum SQL Grammar	19		
Multiple database script	26		
Multiple Database Script	26		
N			
No such database	39		
No table names appear in the Select Table dialog box	39		
Not able to open *.FCS file	39		
O			
ODBC Compliance	18		
ODBC History/Overview	1		
ODBC Setup Dialog Options	11, 39		
ODBC/Application Setup	3, 7, 11		
Optional Index Member Record	30		
Optionally Disable Floating Point Unit Exceptions	37		
Options>>	12		
OTP file	26		
content	27		
layout	27		
organization	27		
requirements	27		
OTP File Contents	27		
OTP File Layout	27		
OTP File Organization	27		
OTP File Requirements	27		
OTP Files	26		
Overview	1		
P			
Performance optimization	25		
Performance Optimization	25		
Purpose	17		
Q			
Quick Start	1, 7		
R			
Record			
data file description	28		
field description	28		
index file description	29		
optional index member	30		
Registry Options	35, 38		
Requirements	18		
S			
Script			
import	26		
method	9		
multiple database	26		
vendor supplied	7		
Script Method	7, 9		
segmode Values	31, 32		
Segmode values	32		
Setup	11		
dialog	11		
Specified driver could not be loaded	39		
SQL conformance	18		
SQL Conformance	18		
SQL grammar			
core	19		
extended	21		
minimum	19		
System Catalog_s Files Need to be Rebuilt	40		
T			
Tutorial	2		
file layout	4		
setup	1		
Tutorial File Layout	3, 4		
Tutorial Setup	1		
V			
Values			
filmod	31		
segmode	32		
Vendor supplied script	7		
W			
Wildcard	26		