

Informix C-ISAM DataBlade Module

User's Guide

Version 1.0
December 1999
Part No. 000-6498

Published by INFORMIX® Press

Informix Corporation
4100 Bohannon Drive
Menlo Park, CA 94025-1032

© 1999 Informix Corporation. All rights reserved. The following are trademarks of Informix Corporation or its affiliates, one or more of which may be registered in the United States or other jurisdictions:

Answers OnLine™; C-ISAM®, Client SDK™; DataBlade®, Data Director™; Decision Frontier™; Dynamic Scalable Architecture™; Dynamic Server™; Dynamic Server™, Developer Edition™; Dynamic Server™ with Advanced Decision Support Option™; Dynamic Server™ with Extended Parallel Option™; Dynamic Server™ with MetaCube®; Dynamic Server™ with Universal Data Option™; Dynamic Server™ with Web Integration Option™; Dynamic Server™, Workgroup Edition™; Dynamic Virtual Machine™; Enterprise Decision Server™; Formation™; Formation Architect™; Formation Flow Engine™; Frameworks for Business Intelligence™; Frameworks Technology™; Gold Mine Data Access®; i.Reach™; i.Sell™; Illustra®, Informix®, Informix® 4GL; Informix® COM Adapter™; Informix® Informed Decisions™; Informix® InquireSM; Informix® Internet Foundation.2000™; InformixLink®, Informix® Red Brick® Decision Server™; Informix Session Proxy™; Informix® Vista™; InfoShelf™; Interforum™; I-Spy™; Mediization™; MetaCube®, NewEra™; Office Connect™; ON-Bar™; OnLine Dynamic Server™; OnLine/Secure Dynamic Server™; OpenCase®, Orca™; PaVER™; Red Brick® and Design; Red Brick® Data Mine™; Red Brick® Mine Builder™; Red Brick® Decisionscape™; Red Brick® Ready™; Red Brick Systems®; Regency Support®, Rely on Red BrickSM; RISK®, Solution DesignSM; STARindex™; STARjoin™; SuperView®, TARGETindex™; TARGETjoin™; The Data Warehouse Company®; Universal Data Warehouse Blueprint™; Universal Database Components™; Universal Web Connect™; ViewPoint®, Visionary™; Web Integration Suite™. The Informix logo is registered with the United States Patent and Trademark Office. The DataBlade logo is registered with the United States Patent and Trademark Office.

Documentation Team: Mark Jeske, Carol Trese, Juliet Shackell, Oakland Editing and Production team

GOVERNMENT LICENSE RIGHTS

Software and documentation acquired by or for the US Government are provided with rights as follows:

- (1) if for civilian agency use, with rights as restricted by vendor's standard license, as prescribed in FAR 12.212;
- (2) if for Dept. of Defense use, with rights as restricted by vendor's standard license, unless superseded by a negotiated vendor license, as prescribed in DFARS 227.7202. Any whole or partial reproduction of software or documentation marked with this legend must reproduce this legend.

Table of Contents

Introduction

In This Introduction	3
About This Manual	3
Software Dependencies	4
Assumptions About Your Locale.	4
Documentation Conventions	4
Typographical Conventions	5
Icon Conventions	6
Sample-Code Conventions.	6
Additional Documentation	7
Online Manuals	7
Printed Manuals	7
Error Message Documentation	8
Documentation Notes and Release Notes.	8
Compliance with Industry Standards	9
Informix Welcomes Your Comments	9

Chapter 1

Overview of the Informix C-ISAM DataBlade Module

In This Chapter	1-3
What Is the C-ISAM DataBlade Module?	1-3
The SQL Access Component	1-4
The Server Storage Component	1-5
When to Use the C-ISAM DataBlade Module Components	1-6
Data Migration	1-7
C-ISAM Application Changes.	1-8
SQL-Based Applications	1-8
Database Server Features	1-9
Limitations	1-9

Chapter 2	C-ISAM DataBlade Module Configuration	
	In This Chapter	2-3
	Installing the DataBlade Module	2-3
	Registering the DataBlade Module	2-4
	C-ISAM DataBlade Module Directory Structure	2-5
	Creating a New Database for C-ISAM Tables	2-7
	Environment Variable Settings	2-8
	CISAMDB_DIR	2-8
	CISAMDB_DATABASE	2-9
	INFORMIXDIR, INFORMIXSQLHOSTS, and INFORMIXSERVER	2-9
	ONCONFIG	2-9
	PATH.	2-10
	Shared Library Search Path	2-10
	Environment Variable Summary	2-11
Chapter 3	SQL Access Component	
	In This Chapter	3-3
	SQL Access Architecture	3-3
	Accessing C-ISAM Files in Native Format	3-4
	Setting cisamdbd Daemon Configuration Parameters	3-5
	CISAMDB_MAXUSERS	3-5
	CISAMDB_TRANSFERSIZE.	3-6
	CISAMDB_WAITTIME.	3-7
	Sample ONCONFIG File	3-7
	Running the cisamdbd Daemon Process	3-8
	Starting the cisamdbd Daemon	3-8
	Stopping the cisamdbd Daemon	3-8
	Removing cisamdbd Shared Memory and Semaphores	3-9
	Verifying That the cisamdbd Daemon Is Running	3-9
	Creating Database Tables for C-ISAM Data Files	3-10
	Creating an extspace	3-10
	Creating Database Tables	3-11
	Database Considerations	3-16
	Database Logging	3-16
	Locking	3-17
	Indexing.	3-18
	Table Fragmentation	3-18
	C-ISAM Access Permissions	3-18

Chapter 4

Server Storage Component

In This Chapter	4-3
Server Storage Architecture	4-3
Executing Server Storage Applications	4-4
Recompiling and Linking C-ISAM Applications	4-5
Referencing the Macro	4-5
Recompiling Existing C-ISAM Applications	4-6
Creating Databases, Tables, and Indexes	4-7
Creating a Database for Server Storage.	4-7
Creating Tables for Server Storage	4-8
Creating Indexes	4-9
Database Considerations	4-10
Access Permissions	4-10
Logging	4-11
Locking	4-11
Indexing	4-11
isdatabase() Function	4-12
C-ISAM Function Differences	4-13
isaddindex()	4-13
isaudit()	4-14
isbuild()	4-14
isdelcurr()	4-15
isdelete()	4-15
isdelindex()	4-16
isdelrec()	4-16
islock()	4-16
islogopen()	4-16
isopen()	4-16
isread()	4-17
isrecover()	4-17
isrename()	4-17
isrewcurr()	4-17
isrewrec()	4-18
isrewrite()	4-18
iswrcurr()	4-18
iswrite()	4-18
New Global Variables	4-19
iserrfile.	4-19
iserrline	4-19
issqlcode	4-19

Chapter 5	Advanced Topics	
	In This Chapter	5-3
	Setting Debugging Options	5-3
	Debugging the SQL Access cisamdbd Daemon Process	5-3
	Debugging SQL Access Client Database Sessions.	5-5
	Debugging Server Storage Applications	5-5
	Using User-Defined Types	5-7
Appendix A	The cisamdbdemo Demonstration Script	
	Index	

Introduction

In This Introduction	3
About This Manual.	3
Software Dependencies	4
Assumptions About Your Locale.	4
Documentation Conventions	4
Typographical Conventions	5
Icon Conventions	6
Sample-Code Conventions	6
Additional Documentation	7
Online Manuals	7
Printed Manuals	7
Error Message Documentation	8
Documentation Notes and Release Notes.	8
Compliance with Industry Standards	9
Informix Welcomes Your Comments.	9

In This Introduction

Read this introduction for an overview of the information provided in this manual and for an understanding of the documentation conventions used.

About This Manual

This manual describes how to configure and use the Informix C-ISAM DataBlade module to provide SQL access for Informix C-ISAM applications. Informix C-ISAM is an application programming interface (API) used to manage indexed sequential access method (ISAM) files. The C-ISAM DataBlade module adds SQL access and database management capabilities to C-ISAM. The C-ISAM DataBlade module is for customers who have developed applications using C-ISAM and want to add RDBMS functionality to their systems or to migrate these applications to an RDBMS environment.

This manual is written for the following users:

- Database administrators
- Database developers

You are assumed to have the following background:

- A working knowledge of your computer, your operating system, and the utilities that your operating system provides
- Some experience working with relational databases or exposure to database concepts
- Some experience with database server administration, operating-system administration, or network administration

If you have limited experience with relational databases, SQL, or your operating system, refer to the [Getting Started](#) manual for your database server for a list of related reading.

Software Dependencies

This manual is written with the assumption that you are using one of the following Informix database servers:

- Informix Dynamic Server 2000, Version 9.2
- Informix Dynamic Server with Universal Data Option, Version 9.14

Assumptions About Your Locale

Informix products can support many languages, cultures, and code sets. All culture-specific information is brought together in a single environment, called a Global Language Support (GLS) locale.

The examples in this manual are written with the assumption that you are using the default locale, **en_us.8859-1**. This locale supports U.S. English format conventions for dates, times, and currency. In addition, this locale supports the ISO 8859-1 code set, which includes the ASCII code set plus 8-bit characters such as é, è, and ñ.

If you plan to use nondefault characters in your data or your SQL identifiers, or if you want to conform to the nondefault collation rules of character data, you need to specify the appropriate nondefault locale.

For instructions on how to specify a nondefault locale, additional syntax, and other considerations related to GLS locales, see the [Informix Guide to GLS Functionality](#).

Documentation Conventions

This section describes the conventions that this manual uses. These conventions make it easier to gather information from this and other volumes in the documentation set.

The following conventions are described:

- Typographical conventions
- Icon conventions
- Sample-code conventions

Typographical Conventions

This manual uses the following conventions to introduce new terms, illustrate screen displays, describe command syntax, and so forth.




Convention	Meaning
KEYWORD	All primary elements in a programming language statement (keywords) appear in uppercase letters in a serif font.
<i>italics</i> <i>italics</i> <i>italics</i>	Within text, new terms and emphasized words appear in italics. Within syntax and code examples, variable values that you are to specify appear in italics.
boldface <i>boldface</i>	Names of program entities (such as classes, events, and tables), environment variables, file and pathnames, and interface elements (such as icons, menu items, and buttons) appear in boldface.
monospace <i>monospace</i>	Information that the product displays and information that you enter appear in a monospace typeface.
KEYSTROKE	Keys that you are to press appear in uppercase letters in a sans serif font.
◆	This symbol indicates the end of product- or platform-specific information.
→	This symbol indicates a menu item. For example, “Choose Tools→Options ” means choose the Options item from the Tools menu.



Tip: When you are instructed to “enter” characters or to “execute” a command, immediately press RETURN after the entry. When you are instructed to “type” the text or to “press” other keys, no RETURN is required.

Icon Conventions

Comment icons identify three types of information, as the following table describes. This information always appears in italics.

Icon	Label	Description
	<i>Warning:</i>	Identifies paragraphs that contain vital instructions, cautions, or critical information
	<i>Important:</i>	Identifies paragraphs that contain significant information about the feature or operation that is being described
	<i>Tip:</i>	Identifies paragraphs that offer additional details or shortcuts for the functionality that is being described

Sample-Code Conventions

Examples of SQL code occur throughout this manual. Except where noted, the code is not specific to any single Informix application development tool. If only SQL statements are listed in the example, they are not delimited by semicolons. For instance, you might see the following code:

```
CONNECT TO stores_demo
...

DELETE FROM customer
      WHERE customer_num = 121
...

COMMIT WORK
DISCONNECT CURRENT
```

To use this SQL code for a specific product, you must apply the syntax rules for that product. For example, if you are using DB-Access, you must delimit multiple statements with semicolons. If you are using an SQL API, you must use EXEC SQL at the start of each statement and a semicolon (or other appropriate delimiter) at the end of the statement.



Tip: *Ellipsis points in a code example indicate that more code would be added in a full application, but it is not necessary to show it to describe the concept being discussed.*

For detailed directions on using SQL statements for a particular application development tool or SQL API, see the manual for your product.

Additional Documentation

For additional information, you might want to refer to the following types of documentation:

- Online manuals
- Printed manuals
- Error message documentation
- Documentation notes and release notes

Online Manuals

An Answers OnLine CD that contains Informix manuals in electronic format is provided with your Informix products. You can install the documentation or access it directly from the CD. For information about how to install, read, and print online manuals, see the insert that accompanies the CD.

Printed Manuals

To order printed manuals, call 1-800-331-1763 or send email to moreinfo@informix.com. Please provide the following information when you place your order:

- The documentation that you need
- The quantity that you need
- Your name, address, and telephone number

UNIX

Error Message Documentation

Informix software products provide ASCII files that contain all of the Informix error messages and their corrective actions.

To read error messages and corrective actions on UNIX, use one of the following utilities.

Utility	Description
finderr	Displays error messages online.
rofferr	Formats error messages for printing.

Note that these utilities are not shipped as part of the C-ISAM DataBlade module but are part of your Informix database server product. ♦

WIN NT

To read error messages and corrective actions in Windows environments, use the Informix find error utility. To display this utility, choose **Start→Programs→Informix** from the Task Bar. ♦

Instructions for using the preceding utilities are available in Answers OnLine. Answers OnLine also provides a list of error messages and corrective actions in HTML format.

Note that the error-finding utilities are not shipped as part of the C-ISAM DataBlade module but are part of your Informix database server product.

Documentation Notes and Release Notes

In addition to printed documentation, the following sections describe the online files that supplement the information in this manual. Please examine these files before you begin using your database server. They contain vital information about application and performance issues.

UNIX

On UNIX platforms, the following online files may appear in the `$INFORMIXDIR/release` directory. Replace `x.y` in the file names with the version number of C-ISAM DataBlade module.

Online File	Purpose
<code>CISAMDBDOCx.y</code>	The documentation notes for your version of this manual describe topics that are not covered in the manual or that were modified since publication.
<code>CISAMDBRELx.y</code>	The release notes describe feature differences from earlier versions of Informix products and how these differences might affect current products. This file also contains information about any known problems and their workarounds.

♦

Compliance with Industry Standards

The American National Standards Institute (ANSI) has established a set of industry standards for SQL. Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.135-1992), which is identical to ISO 9075:1992. In addition, many features of Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL CAE (common applications environment) standards.

Informix Welcomes Your Comments

Let us know what you like or dislike about our manuals. To help us with future versions of our manuals, we want to know about any corrections or clarifications that you would find useful. Include the following information:

- The name and version of the manual that you are using
- Any comments that you have about the manual
- Your name, address, and phone number

Write to us at the following address:

Informix Software, Inc.
Oakland Technical Publications Department
300 Lakeside Drive
Suite 2700
Oakland, CA 94612

If you prefer to send electronic mail, our address is:

`doc@informix.com`

The **doc** alias is reserved exclusively for reporting errors and omissions in our documentation.

We appreciate your suggestions.

Overview of the Informix C-ISAM DataBlade Module

In This Chapter	1-3
What Is the C-ISAM DataBlade Module?	1-3
The SQL Access Component	1-4
The Server Storage Component	1-5
When to Use the C-ISAM DataBlade Module Components	1-6
Data Migration	1-7
C-ISAM Application Changes.	1-8
SQL-Based Applications.	1-8
Database Server Features	1-9
Limitations	1-9

In This Chapter

This chapter introduces the Informix C-ISAM DataBlade module. This chapter includes the following sections:

- [“What Is the C-ISAM DataBlade Module?”](#) next
- [“When to Use the C-ISAM DataBlade Module Components”](#) on page 1-6
- [“Limitations”](#) on page 1-8

These sections briefly describe the C-ISAM DataBlade module and point to where you can find more information in this document or in other documents.

What Is the C-ISAM DataBlade Module?

Indexed Sequential Access Method (ISAM) file management technology is the precursor to today’s relational database management systems (RDBMSs). Because ISAM provides fast and cost-effective access to data without the overhead of an RDBMS, ISAM continues to be a viable environment for many companies. However, ISAM environments lack key features of RDBMS environments, including SQL access, archiving and recovery, high availability, replication, and so on.

Informix C-ISAM is an application programming interface (API) used to manage ISAM files. The C-ISAM DataBlade module is for customers who have developed applications using C-ISAM and plan to add RDBMS functionality to their systems or to migrate these applications to an RDBMS environment.

The C-ISAM DataBlade module adds SQL access and database management capabilities to C-ISAM. It enables you to migrate C-ISAM data to an Informix database server without modifying the C-ISAM applications that access this data. It also provides SQL access to data still stored in a C-ISAM file management system. The C-ISAM DataBlade module brings the features of the RDBMS environment to the C-ISAM environment.

The C-ISAM DataBlade module comprises two components:

- SQL Access
- Server Storage

The SQL Access component provides an SQL interface to the C-ISAM data, and the Server Storage component provides the capability to store ISAM data directly in the database server while allowing C-ISAM programs to continue accessing this data.

The SQL Access Component

The SQL Access component of the C-ISAM DataBlade module places an SQL interface on top of existing C-ISAM data files that are stored on the operating system file system. This enables you to continue to maintain user data in the existing C-ISAM data files while providing concurrent access to that data through SQL. With SQL Access, you can execute all standard SQL statements, including SELECT, INSERT, UPDATE, and DELETE.

To the SQL client, the C-ISAM data “looks and feels” like a standard RDBMS table and can be accessed using any SQL tool that supports database server connectivity (including ODBC and JDBC). SQL Access uses the Virtual Table Interface (VTI) feature of Informix Dynamic Server 2000. VTI is an application programming interface for developing gateways to data not stored in the database server, including flat files, another RDBMS, or any system on the internet. To the user, the data looks as if it is stored in the database server; therefore it is called a *virtual* table.

This architecture is shown in the following illustration.

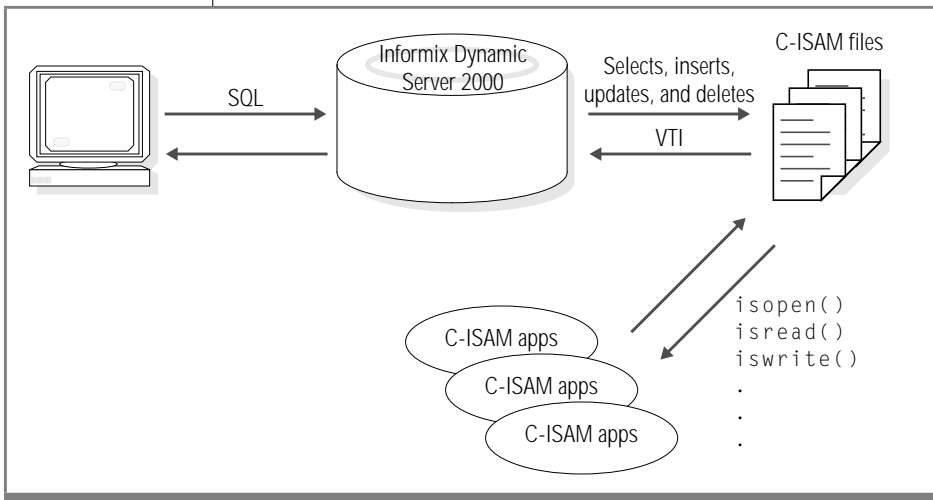


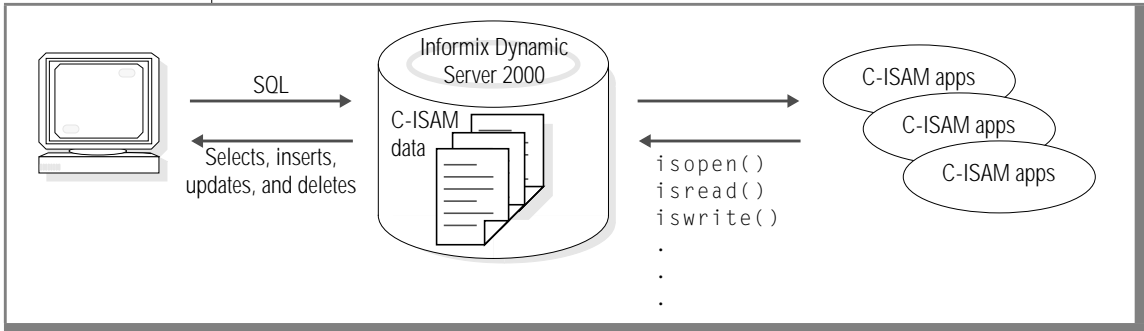
Figure 1-1
The SQL Access Component

The Server Storage Component

You can store ISAM data directly in your database server using the Server Storage component. This gives C-ISAM users all the advantages of an RDBMS, without requiring changes to the C-ISAM program code. To the C-ISAM client, the data “looks and feels” like a standard ISAM file. However, the data is maintained in a physical table in Informix Dynamic Server. This allows the data to be accessed concurrently by *both* C-ISAM and SQL clients.

This architecture is shown in the following illustration.

Figure 1-2
The Server Storage Component



When to Use the C-ISAM DataBlade Module Components

In simplest terms, the SQL Access and Server Storage components both provide SQL access to C-ISAM data. The primary difference between the two components is where the C-ISAM data resides:

- In the file system as **.dat** files for SQL Access
- In the database for Server Storage

The SQL Access and Server Storage components can be used together: they are not mutually exclusive. For example, you can use SQL Access to load data for the tables used in Server Storage. Another possibility is that you might migrate only specific applications and the associated data to Server Storage and leave other data intact in C-ISAM format. By providing both components, the C-ISAM DataBlade module provides you with the flexibility to access your existing data through SQL.

The following tables illustrate the differences between the components in terms of data migration, C-ISAM application changes, SQL-based applications, and database server features.

Data Migration

The following table describes the differences between the components in terms of data migration.

Issue	SQL Access	Server Storage
Data file records must be transferred to new tables inside the database server?	No	Yes. SQL Access can be used to transfer data file records to new tables in the database server instead of writing unload and load programs.
Indexes must be rebuilt?	No	Yes, either before or after data rows are loaded.
File size limited to 2 GB?	Yes	No. Data and all indexes can occupy a maximum of 16 million server pages. For 2 KB page ports, this results in 32 GB of storage space. Systems with 4 KB page ports result in 64 GB of storage space.

C-ISAM Application Changes

The following table describes the differences between the components in terms of C-ISAM application changes.

Issue	SQL Access	Server Storage
Applications must be re-compiled?	No	Yes
Source code changes might be required?	No	Maybe. Although a re-compilation should be the only requirement, minor source code changes might be required.
Applications can run on a remote machine?	No. Applications must reside on the same machine as .dat files unless NFS is being used.	Yes. Applications support all engine connection options including shared memory, network, and stream pipes.

SQL-Based Applications

The following table describes the differences between the components in terms of SQL-based applications.

Issue	SQL Access	Server Storage
Data appears as a normal table to the application?	Yes	Yes
Joins with other relational tables allowed?	Yes	Yes
Communication overhead to a remote process to access data rows?	Yes. The database server must communicate with an external process to store and retrieve data rows.	No. All data is internal to the database server.

Database Server Features

The following table describes the differences between the components in terms of database server features.

Issue	SQL Access	Server Storage
Enterprise replication?	No	Yes
Backup/restore consistently with other server storage?	No	Yes
High performance load/unload utility?	No	Yes

Limitations

The C-ISAM DataBlade module has the following limitations:

- The DataBlade module supports only the following two Informix database servers:
 - Informix Dynamic Server 2000, Version 9.2
 - Informix Dynamic Server with Universal Data Option, Version 9.14

The DataBlade module does not support other Informix database servers, such as Informix Extended Parallel Server.

- Triggers created on database tables are never executed when the tables are accessed by C-ISAM applications. Triggers are executed if the tables are accessed by non-C-ISAM applications using SQL.
- Fragmentation is not supported for tables used in either the Server Storage component or the SQL Access component. Other tables in the database server may still employ fragmentation.

- Some limitations on indexing exist, based on the component used. For SQL Access, indexes in the database server are not allowed. All indexes must be created on the C-ISAM data files directly, using either the **isbuild()** or **isaddindex()** functions.
You can create indexes in the database server for Server Storage. However, the indexes must not be detached from the table. This is accomplished by creating all indexes using the IN TABLE clause to specify a storage location.
- The DataBlade module does not support the use of C-ISAM data files that use Global Language Support.
- The DataBlade module does not support the use of C-ISAM data files that contain variable-length records: only fixed-length records are supported.
- The SQL Access component of the DataBlade module does not support the use of SQL-based transactions. See [Chapter 3, “SQL Access Component,”](#) for more detailed information.
- The Server Storage component of the DataBlade module does not support the use of ANSI databases.

C-ISAM DataBlade Module Configuration

In This Chapter	2-3
Installing the DataBlade Module	2-3
Registering the DataBlade Module	2-4
C-ISAM DataBlade Module Directory Structure	2-5
Creating a New Database for C-ISAM Tables	2-7
Environment Variable Settings	2-8
CISAMDB_DIR	2-8
CISAMDB_DATABASE	2-9
INFORMIXDIR, INFORMIXSQLHOSTS, and INFORMIXSERVER	2-9
ONCONFIG	2-9
PATH	2-10
Shared Library Search Path.	2-10
Environment Variable Summary	2-11

In This Chapter

Before you can begin using the C-ISAM DataBlade module, you must install it and register it in a database. Once registration is complete, you can use either the SQL Access or the Server Storage component to access C-ISAM data.

This chapter contains the following sections:

- “Installing the DataBlade Module,” next
- “Registering the DataBlade Module” on page 2-4
- “C-ISAM DataBlade Module Directory Structure” on page 2-5
- “Creating a New Database for C-ISAM Tables” on page 2-7
- “Environment Variable Settings” on page 2-8

Installing the DataBlade Module

Installing the C-ISAM DataBlade module is different from installing other DataBlade modules. For this reason, you should *not* refer to the [DataBlade Module Installation and Registration Guide](#) for installation instructions. Instead, follow the installation instructions in the *Read Me First* sheet included with the product distribution.

One important way in which the installation of the C-ISAM DataBlade module differs from the installation of other Informix DataBlade modules is that, once installed, only part of the DataBlade module resides in the `$INFORMIXDIR/extend/cisamdb.version` directory. The other part of the DataBlade module resides in the directory you initially unpacked the product files into and ran the installation script for. You use the environment variable `CISAMDB_DIR` to point to this directory.

Once installed, standard DataBlade module files, such as the *.**blb** file and the **objects.sql** file, reside in the `SINFORMIXDIR/extend/cisamdb.version` directory. The section “[C-ISAM DataBlade Module Directory Structure](#)” on [page 2-5](#) describes the files that reside in the `CISAMDB_DIR` directory.

After you have installed the DataBlade module, read the release notes and documentation notes for additional information. See “[Documentation Notes and Release Notes](#)” on [page 8](#) of the introduction to this manual for information on the location of the release and documentation notes.



Warning: *Be sure you install the product in a separate directory from your Informix database server. If you install the DataBlade module into the same directory as existing Informix products, the DataBlade module might fail to operate, and you might introduce problems to your existing products. For example, if you have your Informix database server installed in `/usr/informix`, you might want to install the C-ISAM DataBlade module in `/usr/cisamdb`, or even `/usr/informix/cisamdb`, but not directly in the `/usr/informix` directory.*

Registering the DataBlade Module

To use the C-ISAM DataBlade module in a database, you must first register it in each database in which you want to use it. *Registration* is the process of executing the SQL statements that create the DataBlade module database objects and identify the DataBlade module shared object file to the database server. You register DataBlade modules with the BladeManager application. See the [DataBlade Module Installation and Registration Guide](#) for instructions for registering your DataBlade module.

The following example shows the registration of the C-ISAM DataBlade module in the `cisamdb_demo` database using the BladeManager command-line interface:

```
$ blademgr
fp7_914_shm>register cisamdb.1.00.UC1 cisamdb_demo
Register module cisamdb.1.00.UC1 into database cisamdb_demo?
[Y/n]y
Registering DataBlade module... (may take a while).
DataBlade cisamdb.1.00.UC1 was successfully registered in
database cisamdb_demo.
fp7_914_shm>quit
Disconnecting...
```

The name of the DataBlade module contains the exact version of the product: 1.00.UC1 in this example. Make sure that you are registering the correct version of the DataBlade module. You can view the available DataBlade modules by listing the contents of the `$INFORMIXDIR/extend` directory. Each directory name under **extend** is the name of a DataBlade module that can be registered using BladeManager.

C-ISAM DataBlade Module Directory Structure

After you install the C-ISAM DataBlade module, the DataBlade module product files are distributed between the `CISAMDB_DIR` directory and the `$INFORMIXDIR/extend/cisamdb.version` directory. This section describes the files that reside in the `CISAMDB_DIR` directory.

In the `CISAMDB_DIR` directory you should have a new directory tree containing all of the files required for the product to operate. You should have an owner and group named **informix** on your computer, and all of the files should be owned by **informix** and have a group of **informix**. If this is not the case, review the installation instructions in the *Read Me First* sheet to confirm that you followed them correctly.



Warning: The C-ISAM DataBlade module installation script creates the directory `./cisamdbtmp`. This directory is used to locate various files for the product. Never edit, rename, or remove any files in this directory, and never remove or rename the directory itself. If you do, the C-ISAM DataBlade module will fail.

The C-ISAM DataBlade module product installation includes the following directories:

- bin** The **bin** directory contains the daemon used by the SQL Access component, **cisamdbd**. For more information about this daemon, see [“Setting cisamdbd Daemon Configuration Parameters”](#) on page 3-5.
- demo** The **demo** directory contains a number of C-ISAM programs that are used to illustrate how the DataBlade module operates. Refer to [Appendix A, “The cisamdbdemo Demonstration Script,”](#) for detailed instructions on using these programs.
- etc** The **etc** directory contains files used during product installation. The use of these files is entirely transparent to the operation of the DataBlade module, and you can safely ignore this directory.
- gls** The **gls** directory contains the GLS files necessary for the DataBlade module. The use of these files is entirely transparent to the operation of the DataBlade module, and you can safely ignore this directory.
- incl** The **incl** directory contains the **isam.h** header file that is used to build C-ISAM applications using the Server Storage component. For more information about building C-ISAM applications, see [Chapter 4, “Server Storage Component.”](#)
- lib** The **lib** directory contains the shared library files necessary for C-ISAM programs that use the Server Storage component. You must configure your environment so that the operating system looks into this directory for shared libraries before executing any of these applications. For more information, see [“Shared Library Search Path”](#) on page 2-10.

- msg** The **msg** directory contains the Informix message files required for the DataBlade module. The use of these files is entirely transparent to the operation of the DataBlade module, and you can safely ignore this directory.
- release** The **release** directory contains any last-minute information on the C-ISAM DataBlade module that could not be added to this documentation. It is very important that you read the files in this directory before you continue with the DataBlade module configuration.
- udtdemo** The **udtdemo** directory contains files used to demonstrate how user-defined data types within the database server can be used with the C-ISAM DataBlade module. Refer to [“Using User-Defined Types” on page 5-7](#) for more information on this topic.

Creating a New Database for C-ISAM Tables

A database must exist for the new tables you create that reference your existing C-ISAM files. This database can be either an existing database or a new one created specifically for this DataBlade module. For example, assume that you have created a new database named **cisamdb_demo**, as shown in the following example:

```
CREATE DATABASE cisamdb_demo;
```

This creates the database in the root dbspace of the database server without logging. See the section on the CREATE DATABASE statement in the [Informix Guide to SQL: Syntax](#) for more information on the options available for the CREATE DATABASE statement.

Whether or not you create the database with logging depends on whether you need transaction semantics for the SQL statements you plan to use against the C-ISAM data. If you need transactions for your SQL statements, you should create the database with logging. Your C-ISAM applications are independent of this decision. They will or will not use logging based on their application code.

For more information on database logging, see [“Database Logging”](#) on page 3-16 and [“Creating a Database for Server Storage”](#) on page 4-7.

Environment Variable Settings

The following environment variables are required for the C-ISAM DataBlade module to operate:

- **CISAMDB_DIR**
- **CISAMDB_DATABASE**
- **INFORMIXDIR**
- **INFORMIXSERVER**
- **INFORMIXSQLHOSTS**
- **ONCONFIG**
- **PATH**
- Shared Library Search Path

These variables are described next.

CISAMDB_DIR

The **CISAMDB_DIR** environment variable contains the full path to the directory where you installed the C-ISAM DataBlade module. While this variable is not required by any of the product binaries or applications built using the DataBlade module, it provides a single place to specify the installation path referenced at several points throughout this manual. You should set this environment variable and reference it instead of hard coding the installation path.

For example, if you installed the DataBlade module in the directory **/usr/cisamdb**, you would set **CISAMDB_DIR** in the UNIX Bourne shell as follows:

```
$ CISAMDB_DIR=/usr/cisamdb
$ export CISAMDB_DIR
```

CISAMDB_DATABASE

The `CISAMDB_DATABASE` environment variable is required for C-ISAM applications that use the Server Storage component of the C-ISAM DataBlade module. It is not required for SQL applications or the SQL Access component. It specifies the name of the database to use for C-ISAM data files and indexes. In standard C-ISAM, there is no concept of a database, but the database server requires that a database be open before access to tables can occur.

For example, if your C-ISAM application has tables located in a database named **payroll**, you would set `CISAMDB_DATABASE` in the UNIX Bourne shell as follows:

```
$ CISAMDB_DATABASE=payroll
$ export CISAMDB_DATABASE
```

INFORMIXDIR, INFORMIXSQLHOSTS, and INFORMIXSERVER

The `INFORMIXDIR`, `INFORMIXSQLHOSTS`, and `INFORMIXSERVER` environment variables are used by C-ISAM Server Storage applications to locate an **sqlhosts** file and a **servername** within it. For more information on client/server communications and the **sqlhosts** file, see the [Administrator's Guide](#) for your database server.

You must set these environment variables correctly for C-ISAM applications that use the Server Storage component to connect to the appropriate database server.

ONCONFIG

The `ONCONFIG` environment variable is used by the `cisamdbd` daemon of the C-ISAM DataBlade module SQL Access component. See “[Setting cisamdbd Daemon Configuration Parameters](#)” on page 3-5 for more information on the `ONCONFIG` file settings used by the SQL Access component.

PATH

You must include the path `$CISAMDB_DIR/bin` in your existing setting of the `PATH` environment variable. This allows you to execute C-ISAM DataBlade module binaries regardless of where they are located on your computer. Here is example syntax for setting your path in the UNIX Bourne shell:

```
$ PATH=$CISAMDB_DIR/bin:$PATH
$ export PATH
```

Shared Library Search Path

You need to set the appropriate environment variable for shared library searching to include `$CISAMDB_DIR/lib`. If you do not have this setting in your environment, you will probably see an error similar to the following one when you run a C-ISAM Server Storage application:

```
ld.so.1: ./prog: fatal: libifsql.so: can't open file: errno=2
Killed
```

Each of the following platforms has its own environment variable:

- SOLARIS systems use `LD_LIBRARY_PATH` to specify directories to search for shared libraries, as follows:

```
$ LD_LIBRARY_PATH=$CISAMDB_DIR/lib:$LD_LIBRARY_PATH
$ export LD_LIBRARY_PATH
```

- AIX systems use `LIBPATH` to specify directories to search for shared libraries, as follows:

```
$ LIBPATH=$CISAMDB_DIR/lib:$LIBPATH
$ export LIBPATH
```

- HP-UX systems use `SHLIB_PATH` to specify directories to search for shared libraries, as follows:

```
$ SHLIB_PATH=$CISAMDB_DIR/lib:$SHLIB_PATH
$ export SHLIB_PATH
```

- SEQUENT systems use `LD_LIBRARY_PATH` to specify directories to search for shared libraries, as follows:

```
$ LD_LIBRARY_PATH=$CISAMDB_DIR/lib:$LD_LIBRARY_PATH
$ export LD_LIBRARY_PATH
```



Important: It is important to place `$CISAMDB_DIR/lib` at the beginning of the shared library path list when you execute C-ISAM Server Storage applications so that the appropriate shared libraries are found. If you have other Informix products installed on the same computer as the C-ISAM DataBlade module, you might already have the shared library path setting configured for these products. In this case, it is still necessary to add the `$CISAMDB_DIR/lib` directory to the beginning of the path list.

Environment Variable Summary

The following table summarizes, by activity, when the various environment variables are needed.

Task	Required Environment Variable	Optional Environment Variable	Recommended Environment Variable
Compiling C-ISAM Server Storage applications	None	CISAMDB_DIR	CISAMDB_DIR
Executing C-ISAM Server Storage applications	CISAMDB_DATABASE INFORMIXDIR INFORMIXSERVER Shared library path	INFORMIXSQLHOSTS	None
Executing cisamdbd , the SQL Access daemon	INFORMIXDIR ONCONFIG PATH	None	None
Executing non-C-ISAM SQL applications	INFORMIXDIR INFORMIXSERVER	INFORMIXSQLHOSTS Shared Library Path	None
Executing traditional C-ISAM applications	None	None	None

SQL Access Component

In This Chapter	3-3
SQL Access Architecture	3-3
Accessing C-ISAM Files in Native Format	3-4
Setting cisamdbd Daemon Configuration Parameters	3-5
CISAMDB_MAXUSERS	3-5
CISAMDB_TRANSFERSIZE	3-6
CISAMDB_WAITTIME	3-7
Sample ONCONFIG File	3-7
Running the cisamdbd Daemon Process	3-8
Starting the cisamdbd Daemon	3-8
Stopping the cisamdbd Daemon	3-8
Removing cisamdbd Shared Memory and Semaphores	3-9
Verifying That the cisamdbd Daemon Is Running	3-9
Creating Database Tables for C-ISAM Data Files.	3-10
Creating an extspace	3-10
Creating Database Tables	3-11
Table Name.	3-12
Column Types	3-12
Storage Location	3-14
Access Method	3-14
Database Considerations	3-16
Database Logging	3-16
Locking	3-17
Indexing	3-18
Table Fragmentation	3-18
C-ISAM Access Permissions	3-18

In This Chapter

This chapter describes configuration and use of the C-ISAM DataBlade module's SQL Access component.

This chapter covers the following topics:

- [“SQL Access Architecture,” next](#)
- [“Accessing C-ISAM Files in Native Format” on page 3-4](#)
- [“Setting cisamdbd Daemon Configuration Parameters” on page 3-5](#)
- [“Running the cisamdbd Daemon Process” on page 3-8](#)
- [“Creating Database Tables for C-ISAM Data Files” on page 3-10](#)
- [“Database Considerations” on page 3-15](#)

SQL Access Architecture

The SQL Access component of the C-ISAM DataBlade module provides SQL access to existing C-ISAM files, in their native format, that reside on the operating system file system. The **cisamdbd** daemon process allows you to achieve this access. This daemon process communicates with the Virtual Table Interface (VTI) in the database server over shared memory and semaphores.

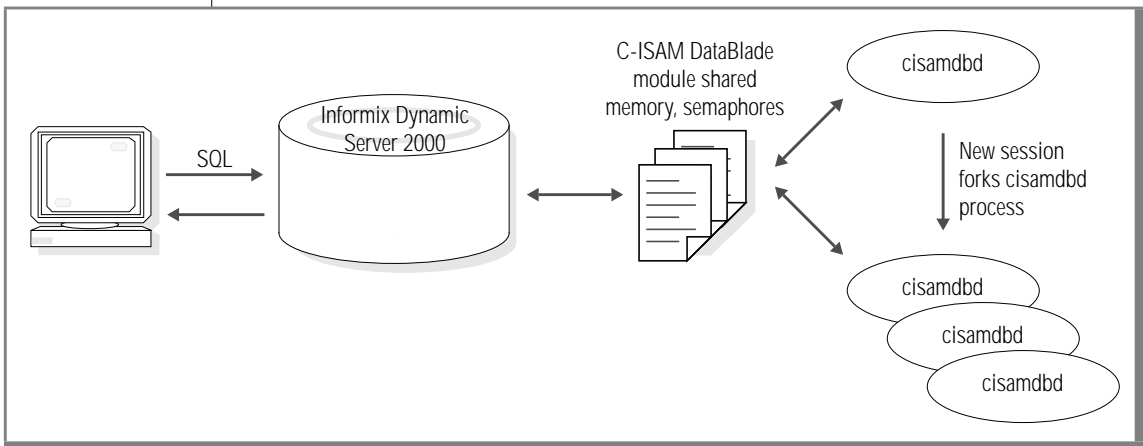
The shared memory segments and semaphores used by SQL Access are separate from those already in use by the database server. The **cisamdbd** daemon allocates and controls these interprocess communication (IPC) objects.

When a new database server session requests access to an SQL Access table, the **cisamdbd** daemon process forks itself to create a new process that handles all C-ISAM file access for that particular session. When the database session is closed, the forked process performs any necessary cleanup activities and exits. In this manner, there is always a single unique operating system process handling the C-ISAM file access for each database session. For example, if 10 database sessions reference SQL Access tables, then 11 **cisamdbd** processes will exist: one for each database session and one for the initial **cisamdbd** daemon that is always running and handles the creation of new database sessions.

Database sessions that never access a table defined using the SQL Access component will never have a daemon process associated with them.

Figure 3-1 illustrates this architecture.

Figure 3-1
The SQL Access Component



Accessing C-ISAM Files in Native Format

Follow these steps to use the SQL Access component of the C-ISAM DataBlade module to access existing C-ISAM files in their native format.

To access C-ISAM files in their native format

1. Set the **cisamdbd** daemon configuration parameters in the ONCONFIG file. See [“Setting *cisamdbd* Daemon Configuration Parameters,”](#) next.
2. Start the **cisamdbd** daemon process. See [“Running the *cisamdbd* Daemon Process”](#) on page 3-8.
3. Create a database table for each C-ISAM data file that you want to access. See [“Creating Database Tables for C-ISAM Data Files”](#) on page 3-10.
4. Address relevant database issues. See [“Database Considerations”](#) on page 3-15.

These steps are described in the following sections.

Setting *cisamdbd* Daemon Configuration Parameters

To use the **cisamdbd** daemon, you must add three configuration parameters to the ONCONFIG file for your database server:

- **CISAMDB_MAXUSERS**
- **CISAMDB_TRANSFERSIZE**
- **CISAMDB_WAITTIME**

These configuration parameters are described in the following sections. A sample ONCONFIG file on [page 3-7](#) illustrates how you should add the new parameters to the file.

CISAMDB_MAXUSERS

The **CISAMDB_MAXUSERS** parameter specifies the maximum number of client sessions that can use the SQL Access component of the database. This parameter takes a single integer, as follows:

```
CISAMDB_MAXUSERS    10
```

If any database session attempts to execute SQL statements against an SQL Access component table when this maximum number of users exists, it receives the following error message:

```
CISAMDB:MAXUSERS( # ) exceeded, access denied.
```

CISAMDB_TRANSFERSIZE

The **CISAMDB_TRANSFERSIZE** parameter specifies the size, in bytes, of shared memory to use for transferring data between the C-ISAM data files and the client's database session. This value must be at least the length of the longest record in your C-ISAM files.

If you are running your C-ISAM applications on a small computer with memory constraints, Informix recommends that you set the parameter **CISAMDB_TRANSFERSIZE** to a smaller size (such as 16 KB, 24 KB, or 32 KB) to avoid using up too much memory. If, however, you have no constraints on the amount of memory the applications use, Informix recommends that you increase the size of this parameter to 128 KB or 256 KB to increase performance. The larger the memory transfer size, the smaller the number of transfers between the database server and **cisamdb** daemon process.

The parameter takes a single integer argument, as follows:

```
CISAMDB_TRANSFERSIZE 16384
```

This would specify a transfer size of 16 KB.

The **cisamdbd** daemon allocates one transfer buffer of this size in shared memory for each user defined by the **CISAMDB_MAXUSERS** parameter. This memory is allocated when the daemon begins execution, and it exists regardless of how many database sessions are actually accessing SQL Access tables.

Use the following formula to estimate the total shared memory that the **cisamdbd** daemon allocates:

```
CISAMDB_MAXUSERS * CISAMDB_TRANSFERSIZE
```

CISAMDB_WAITTIME

The `CISAMDB_WAITTIME` parameter specifies the maximum number of seconds the database session waits for a response from the `cisamdbd` daemon before it returns an error. Database sessions make requests of a single `cisamdbd` daemon process.

In the following example, the database session waits up to 30 seconds for a response before it returns an error:

```
CISAMDB_WAITTIME    30
```

Important: *Informix recommends 30 seconds for this parameter. If you set this value too high, your users will have to wait too long for an error. If you set this value too low, your users may encounter errors if the machine is simply to heavily loaded.*



Sample ONCONFIG File

The following partial sample of an `ONCONFIG` configuration file illustrates how you should add the new parameters to the file. Although you can place the configuration parameters anywhere in the file, if you place them at the top of the file, they will be immediately visible to anyone working with the file.

```

#*****
#
#                               INFORMIX SOFTWARE, INC.
#
# Title:           onconfig.std
# Description:    INFORMIX-OnLine Configuration Parameters
#
#*****

CISAMDB_MAXUSERS      10
CISAMDB_TRANSFERSIZE 16384
CISAMDB_WAITTIME     10

SBSPACENAME sbSPACE1 # Default smartblob space name

ROOTNAME      rootdbs      # Root dbspace name
ROOTPATH      /home2/cdbdemo/dev/chunk1
ROOTOFFSET    0            # Offset of root dbspace into
device (Kbytes)
ROOTSIZE      100000       # Size of root dbspace (Kbytes)
...

```

Running the *cisamdbd* Daemon Process

You must add the ***cisamdbd*** daemon configuration parameters to your database server **ONCONFIG** file before you start the ***cisamdbd*** daemon process. Be sure you restart your database server so that the new **ONCONFIG** file parameters take effect. For detailed information on the ***cisamdbd*** daemon configuration parameters, see [“Setting *cisamdbd* Daemon Configuration Parameters” on page 3-5](#).

Starting the *cisamdbd* Daemon

You start the ***cisamdbd*** daemon process by executing the ***cisamdbd*** command with no options while logged in as user **informix**, as shown in the following example:

```
$ cisamdbd
```

The ***cisamdbd*** daemon places any status or warning messages in the database server online log file. For example, the following output from the **onstat -m** utility illustrates some of these messages:

```
16:00:08 CISAMDB: cisamdb Master Daemon (pid=25112) launched successfully
08:58:28 CISAMDB: Master daemon has forked. pid = 2762, ppid = 25112
08:58:29 CISAMDB: Child daemon has forked. pid = 2763, ppid = 1
09:01:40 CISAMDB: Master daemon has forked. pid = 2798, ppid = 25112
09:01:41 CISAMDB: Child daemon has forked. pid = 2799, ppid = 1
```

The message **Child daemon has forked. pid = ...** indicates that a database session has referenced an **SQL Access** table and that the master ***cisamdb*** daemon has created a new process to handle requests for that session.

Stopping the *cisamdbd* Daemon

You stop the ***cisamdbd*** daemon process by executing the ***cisamdbd*** command with the **-kill** option while logged in as user **informix**, as shown in the following example:

```
$ cisamdbd -kill
```

Although the **-kill** option stops the **cisamdbd** daemon process, the shared memory and semaphores allocated by the daemon process are not automatically removed because the database server is still attached to these resources. For detailed information on removing **cisamdbd** shared memory and semaphores, see [“Removing *cisamdbd* Shared Memory and Semaphores” on page 3-9](#).

When the **cisamdbd** daemon is not running, all database sessions that execute SQL statements against SQL Access component tables receive an error message indicating that the daemon is not running.

Removing *cisamdbd* Shared Memory and Semaphores

You can remove the shared memory and semaphores used by the **cisamdbd** daemon process. Two conditions must exist before these resources can be safely removed by the operating system: the **cisamdbd** daemon process must be stopped, and the database server must be shut down.

To remove shared memory and semaphores

1. Stop the **cisamdbd** daemon process. See [“Stopping the *cisamdbd* Daemon” on page 3-8](#).
2. Shut down your database server.
3. While logged in as **informix**, execute the **cisamdbd** command with the **-d** option, as shown in the following example:

```
$ cisamdbd -d
```

If successful, the command returns a message similar to the following message:

```
Shared Memory removed. Id = 103.  
Semaphores removed. Id = 2.
```

Verifying That the *cisamdbd* Daemon Is Running

Use the **-r** command-line option of the **cisamdbd** command while logged in as user **informix** to determine if the **cisamdbd** daemon is running, as shown in the following example:

```
$ cisamdbd -r
```

If the **cisamdbd** daemon is running, the command returns its process ID and the exit code of the process is set to 0.

If the daemon process is not running, the command returns following message and the exit code is set to 1:

```
No cisamdbd process is currently running.
```

Creating Database Tables for C-ISAM Data Files

You must create a database table for each C-ISAM data file that you want to access. The database table is a template for the location of columns within the C-ISAM data file record. The syntax you use to create the database table also indicates where in the operating system file system the data file resides.

It is important to remember that no data is stored in these database tables. Any SQL statements that access them are translated into appropriate calls against a C-ISAM data file and executed by a **cisamdb** daemon process.

To create SQL Access component database tables

1. Create an extspace for each C-ISAM data file that you want to access.
2. Create a database table for each C-ISAM data file that you want to access.

These tasks are described in the following sections.

Creating an extspace

An *extspace* is a logical name associated with an arbitrary string that can point to any type of file on any type of device. In the case of the SQL Access component, the arbitrary string is a full pathname to a directory where the C-ISAM data files you want to access exist.

You create an extspace with the database server **onspaces** utility, as shown in the following example:

```
onspaces -c -x cisamdb_demo_space -l /export/home/cisam/files
```


The **cisamdb_demo_space** extspace is used later in your CREATE TABLE statement. The path **/export/home/cisam/files** is the directory where your C-ISAM **.dat** and **.idx** files reside. You can override this path in your CREATE TABLE statement later, but it is more convenient from an administration perspective to have the path in a single place.

For more information on extspaces, see the [Administrator's Guide](#) for your database server.

Creating Database Tables

Each C-ISAM data file that you want to access using the SQL Access component of the C-ISAM DataBlade module requires a separate CREATE TABLE statement, such as the following:

```
CREATE TABLE employee(
    emp_number      INTEGER,
    emp_last_name   CHAR(20),
    emp_first_name  CHAR(20),
    emp_address     CHAR(20),
    emp_city        CHAR(20)
)
IN cisamdb_demo_space
USING cisamdb(directory= "/local/cisamdb",
              filename = "employee" );
```

This CREATE TABLE statement represents the layout of the **employee.dat** file created and accessed by the **ex1.c** through **ex7.c** C-ISAM programs in the C-ISAM DataBlade module **demo** directory.

There are four items of interest about this CREATE TABLE statement:

- Table name
- Column types
- Storage location
- Access method

These items are described next.

Table Name

Typically, the table name matches the name of the C-ISAM data file you want to reference with this table, without the **.dat** extension. In the previous example, the **employee** table references the **employee.dat** file.

If you want to create a database table where the name does not match the name of the **.dat** file, you can use the **filename =** argument to the access method to specify a different name for the **.dat** file. In the following example, a table named **performance** is created to reference the C-ISAM file named **001performance.dat**:

```
CREATE TABLE performance (
  emp_number    INTEGER,
  per_date      char(6),
  per_rating    CHAR(1),
  per_new_salary float,
  per_new_title CHAR(30)
)
IN cisamdb_demo_space
USING cisamdb(directory= "/local/cisamdb",
              filename = "001performance" );
```

Note that the **.dat** extension is not specified in the **filename** argument but is automatically assumed by the DataBlade module.

Column Types

The table columns provide a layout of the fields within the associated C-ISAM data file record. The columns should be listed in order, starting from byte 0 in the record and progressing to the end of the record.

For example, the record layout for the **employee.dat** file is shown in the following table.

Description	Type	Length	Offset from Beginning of Record
Employee number	long	4	0
Last name	char	20	4

(1 of 2)

Description	Type	Length	Offset from Beginning of Record
First name	char	20	24
Address	char	20	44
City	char	20	64

(2 of 2)

This corresponds to the columns specified in the CREATE TABLE statement for the **employee** table shown at the beginning of this section. The record length for this data file is 84 bytes, which must match exactly the row length of the defined database table.

The following table shows the possible C-ISAM field types and the corresponding database column type to use.

C-ISAM Type	Database Column Type	Length in Bytes
char	char	variable
short	smallint	2
long	integer	4
float	smallfloat	4
double	float	8
decimal	decimal	variable

Storage Location

The storage location for an SQL Access table must be a previously created extspace. You specify the storage location with the IN clause of the CREATE TABLE statement. In the following example, the storage location is **cisamdb_demo_space**:

```
CREATE TABLE employee(  
    emp_number      INTEGER,  
    emp_last_name   CHAR(20),  
    emp_first_name  CHAR(20),  
    emp_address     CHAR(20),  
    emp_city        CHAR(20)  
)  
IN cisamdb_demo_space  
USING cisamdb(directory= "/local/cisamdb" );
```

You must specify a storage location when you create an SQL Access table. For more information on creating extspaces, see [“Creating an extspace” on page 3-10](#).

Access Method

When you create an SQL Access table, you must specify the **cisamdb** access method in the USING clause of the CREATE TABLE statement, as shown in the following example:

```
CREATE TABLE employee(  
    emp_number      INTEGER,  
    emp_last_name   CHAR(20),  
    emp_first_name  CHAR(20),  
    emp_address     CHAR(20),  
    emp_city        CHAR(20)  
)  
IN cisamdb_demo_space  
USING cisamdb(directory= "/local/cisamdb" );
```

The keyword **cisamdb** is the name of the access method for the C-ISAM DataBlade module. This name must be used in each CREATE TABLE statement exactly as shown.

The parenthesized text after the **cisamdb** keyword contains optional arguments to the DataBlade module that specify where the C-ISAM data file resides. These arguments override the pathname associated with the extspace. If the directory argument is not present in the CREATE TABLE statement, then the pathname associated with the extspace is used. The following example shows the same CREATE TABLE statement without the **directory** argument:

```
CREATE TABLE employee(
    emp_number    INTEGER,
    emp_last_name CHAR(20),
    emp_first_name CHAR(20),
    emp_address   CHAR(20),
    emp_city      CHAR(20)
)
IN cisamdb_demo_space
USING cisamdb;
```

The **filename** argument can be used when the name of the table being created does not match the name of the **.dat** file, as follows:

```
CREATE TABLE employee(
    emp_number    INTEGER,
    emp_last_name CHAR(20),
    emp_first_name CHAR(20),
    emp_address   CHAR(20),
    emp_city      CHAR(20)
)
IN cisamdb_demo_space
USING cisamdb(filename="employee100");
```

In this case the **.dat** file is named **employee100.dat**, and the table is named **employee**. Both the directory and filename arguments are optional. They can also be used together, as in this example:

```
CREATE TABLE employee(
    emp_number    INTEGER,
    emp_last_name CHAR(20),
    emp_first_name CHAR(20),
    emp_address   CHAR(20),
    emp_city      CHAR(20)
)
IN cisamdb_demo_space
USING cisamdb(directory="/local/cisamdb", filename="employee100");
```

Here, the C-ISAM file **/local/cisamdb/employee100.dat** is used whenever the **employee** table is referenced.

Database Considerations

This section of the manual describes how various database server features relate to C-ISAM files accessed using the SQL Access component of the C-ISAM DataBlade module. The following features are discussed:

- [“Database Logging,”](#) next
- [“Locking”](#) on page 3-17
- [“Indexing”](#) on page 3-17
- [“Table Fragmentation”](#) on page 3-18
- [“C-ISAM Access Permissions”](#) on page 3-18

Database Logging

The SQL Access component of the C-ISAM DataBlade module does not currently support SQL level transactions. Although the SQL transaction statements BEGIN, ROLLBACK, and COMMIT WORK may be issued, they are ignored by the C-ISAM DataBlade module. Every SQL statement executed against an SQL Access component table is treated as an atomic transaction. It either succeeds or fails regardless of whether an SQL-level transaction is in effect.

For example, consider the following DELETE statement against the **employee_sa** table in the DataBlade module **demo** directory:

```
BEGIN WORK ;  
DELETE FROM employee_sa WHERE emp_number = 2 ;  
ROLLBACK WORK ;
```

In the example, the record where **emp_number** equals 2 will be deleted from the C-ISAM **.dat** file, and the BEGIN and ROLLBACK WORK statements are completely ignored.

All SQL statements that alter the C-ISAM **.dat** file in any way, such as INSERT, UPDATE, and DELETE, are not logged in the C-ISAM log file (if one exists). The **cisamdbd** daemon process never calls the C-ISAM **islogopen()** function. Because of this, SQL changes to the **.dat** files cannot be recovered using the **isrecover()** function of C-ISAM.

If you need to log and recover changes to your **.dat** file while using the SQL Access component of the C-ISAM DataBlade module, you should perform all file changes using C-ISAM applications instead of SQL statements.

For more information on this topic, see the *Informix C-ISAM Programmer's Manual*.

Locking

The SQL Access component of the C-ISAM DataBlade module performs manual record locking based on the isolation level of the client's database session. The following table describes the type of locking used for each isolation level.

Isolation Level	Locking Performed
Dirty Read	No locking is performed; all isread() calls are performed <i>without</i> using the ISLOCK flag.
Committed Read	Each record is locked, and then the lock is immediately released before the record is returned to the client. Locking the record verifies that it has been committed in the file and is not part of an existing transaction that could be rolled back. Locking is performed by using the ISLOCK flag for isread() .
Cursor Stability	The last row returned to the client is locked using the ISLOCK flag for isread() . Upon a read of the next row, all locks are released and the next row is locked before it is returned to the client. This always leaves a lock on the last row read.
Repeatable Read	All rows read are locked using the ISLOCK flag of isread() . No locks are released until the query completes.

For all isolation levels, locks on all records are released when a query completes.

For more information on database isolation levels, see the SET ISOLATION statement in the *Informix Guide to SQL: Syntax*.

Indexing

The SQL Access component of the C-ISAM DataBlade module does not support the creation of database indexes using SQL. This means that you cannot create an index on an SQL Access table using the CREATE INDEX statement. Indexes for C-ISAM data files should be created using the **isaddindex()** function from within a C-ISAM application.

The C-ISAM DataBlade module is aware of all C-ISAM indexes that exist on data files (as long as they reside in the same directory as the corresponding **.dat** file) and uses them as appropriate when executing SQL statements.

Table Fragmentation

SQL-level table fragmentation is not supported for tables that use the SQL Access component of the C-ISAM DataBlade module.

C-ISAM Access Permissions

C-ISAM applications rely on operating system permissions to allow or restrict access to C-ISAM data and index files. If a user running a C-ISAM application does not have appropriate operating system permissions on the files that the application attempts to open, then the **isopen()** function returns an error.

The SQL Access component directly mirrors this environment by running the forked **csamdbd** daemon process as the same user who opened the database session. In this way, SQL database sessions have permissions to C-ISAM files equivalent to those of C-ISAM application users. If the user who opened the database session does not have operating system access permissions to the C-ISAM data and index files they access, an appropriate error message is returned.

Server Storage Component

In This Chapter	4-3
Server Storage Architecture	4-3
Executing Server Storage Applications	4-4
Recompiling and Linking C-ISAM Applications	4-5
Referencing the Macro	4-5
Recompiling Existing C-ISAM Applications	4-6
Creating Databases, Tables, and Indexes	4-7
Creating a Database for Server Storage.	4-7
Creating Tables for Server Storage	4-8
Populating the Database Tables	4-8
Creating Indexes	4-9
Database Considerations	4-10
Access Permissions	4-10
Logging	4-11
Locking	4-11
Indexing	4-11
isdatabase() Function	4-12
C-ISAM Function Differences	4-13
isaddindex()	4-13
isaudit()	4-14
isbuild()	4-14
isdelcurr()	4-15
isdelete()	4-15
isdelindex()	4-16
isdelrec()	4-16

islock()	4-16
islogopen()	4-16
isopen()	4-16
isread()	4-17
isrecover()	4-17
isrename()	4-17
isrewcurr()	4-17
isrewrec()	4-18
isrewrite()	4-18
iswrcurr()	4-18
iswrite()	4-18
New Global Variables	4-19
iserrfile	4-19
iserrline	4-19
issqlcode	4-19

In This Chapter

This chapter describes in detail the configuration and use of the C-ISAM DataBlade module Server Storage component.

This chapter covers the following topics:

- “Server Storage Architecture,” next
- “Executing Server Storage Applications” on page 4-4
- “Recompiling and Linking C-ISAM Applications” on page 4-5
- “Creating Databases, Tables, and Indexes” on page 4-7
- “Database Considerations” on page 4-9
- “isdatabase() Function” on page 4-11
- “C-ISAM Function Differences” on page 4-13
- “New Global Variables” on page 4-18

Server Storage Architecture

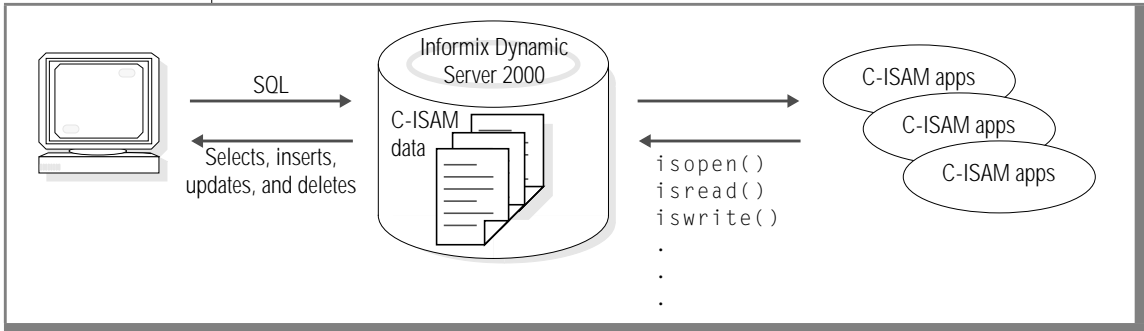
The Server Storage component of the C-ISAM DataBlade module provides SQL access to C-ISAM data that resides in database tables within the database server rather than C-ISAM data that resides in C-ISAM files within the operating system file system.

The C-ISAM data is transferred from its native format in the file system into database tables only once, typically by the database administrator or the owner of the applications that access the C-ISAM data.

The Server Storage component of the C-ISAM DataBlade module also provides a new **libisam.a** library so that you can recompile and execute your existing C-ISAM applications against those database tables.

This architecture is shown in the following illustration.

Figure 4-1
The Server Storage Component



Executing Server Storage Applications

Follow these steps to move existing C-ISAM data files into the database tables and recompile existing C-ISAM applications to access the data.

To execute Server Storage Applications

1. Recompile and link C-ISAM with the libraries provided with the C-ISAM DataBlade module. See [“Recompiling and Linking C-ISAM Applications”](#) on page 4-5.
2. Create a database, tables, and indexes for Server Storage. See [“Creating Databases, Tables, and Indexes”](#) on page 4-7.
3. Use the SQL Access component of the C-ISAM DataBlade module to populate the database tables with the data from existing **.dat** files in the file system. See [“Populating the Database Tables”](#) on page 4-8.
4. Set the following environment variables:
 - **CISAMDB_DATABASE**
 - **INFORMIXDIR**
 - **INFORMIXSERVER**
 - **INFORMIXSQLHOSTS**

- Shared Library Path variables
See [Chapter 2, “C-ISAM DataBlade Module Configuration,”](#) for more information on setting these environment variables.
- 5. Address topics described in [“Database Considerations”](#) on page 4-9.

These steps are described in the following sections.

Recompiling and Linking C-ISAM Applications

Follow these steps to recompile and link existing C-ISAM applications with the libraries provided with the C-ISAM DataBlade module instead of the **libisam.a** library provided with standard C-ISAM.

To recompile and link C-ISAM applications

1. Reference the macro that contains a list of all the libraries required for Server Storage in your makefile. See [“Referencing the Macro”](#) on page 4-5.
2. Recompile and link C-ISAM with the libraries provided with the C-ISAM DataBlade module. See [“Recompiling Existing C-ISAM Applications”](#) on page 4-6.

These steps are described in the following subsections.

Referencing the Macro

The file `SCISAMDB_DIR/lib/cisamdb_make.inc` contains an `INCLUDE` statement that you can use in your makefile to reference a macro that contains a list of all the libraries required for the Server Storage component of the C-ISAM DataBlade module.

The content of the **cisamdb_make.inc** file is shown here:

```
CISAMDB_INCL= -I$(CISAMDB_DIR)/incl
CISAMDB_LIBS= -L$(CISAMDB_DIR)/lib \
              -lisam \
              -lifsq1 \
              -lifasf \
              -lifgen \
              -lifos \
              -lifgl1 \
              -lnsl \
              -lsocket \
              -laio \
              -lm \
              -ldl \
              -lelf \
              $(CISAMDB_DIR)/lib/checkapi.o \
              -lifglx
```

You can reference the macros in the **cisamdb_make.inc** file in your make files without having to consider which libraries should be linked or where they are located. For example, a section of the makefile from the **\$(CISAMDB_DIR)/demo** directory resembles the following code:

```
include $(CISAMDB_DIR)/lib/cisamdb_make.inc

... (lines removed)...

$(PROG1): $(P1SRC)
    $(CC) $(DEFINES) -o @$ $(FLAGS) $(P1SRC) $(CISAMDB_LIBS)

$(PROG2): $(P2SRC)
    $(CC) $(DEFINES) -o @$ $(FLAGS) $(P2SRC) $(CISAMDB_LIBS)
```

Recompiling Existing C-ISAM Applications

Recompile and link your C-ISAM application with the libraries provided with the C-ISAM DataBlade module.

If you use the **make** utility to recompile and link your applications, be sure you reference the specified macros in your makefile, as described in the previous section.

If you are not using the **make** utility, compile your C-ISAM applications using your C language compiler and the libraries provided with the C-ISAM DataBlade module.

To compile your C-ISAM application, use a command link similar to the following example:

```
cc program.c -I$CISAMDB_DIR/incl -L$CISAMDB_DIR/lib -lisam -lifsq1 -lifasf \
-lifgen -lifos -lifgls -lnsl -lsocket -laio -lm -ldl -lelf \
$CISAMDB_DIR/lib/checkapi.o -lifglx
```

Command links must fit on a single line; for clarity, however, the preceding example shows the command link on three separate lines.

In the example, *program* refers to the name of your C-ISAM program. The exact list of libraries for your operating system might differ from the ones listed in the example. Refer to the `$CISAMDB_DIR/lib/cisamdb_make.inc` file the exact list of required files.

Creating Databases, Tables, and Indexes

This section of the manual describes the requirements for creating databases, tables, and indexes for use with the Server Storage component of the C-ISAM.

Creating a Database for Server Storage

You must create a database to hold the tables required for your Server Storage application. The database must have logging enabled if any of the Server Storage applications use the **islogopen()** function.

Traditional C-ISAM applications have no concept of a database; they simply open, access, and close individual files or tables. Use the environment variable `CISAMDB_DATABASE` to name the database to use for tables opened using the **isbuild()** or **isopen()** functions.

The following example shows how to set the `CISAMDB_DATABASE` environment variable in the UNIX Bourne shell:

```
$ CISAMDB_DATABASE=cisamdb_demo_db
$ export CISAMDB_DATABASE
```

An alternative to using the `CISAMDB_DATABASE` environment variable is to call the Server Storage function **isdatabase()** at the beginning of an application. See [“isdatabase\(\) Function” on page 4-11](#) for more information on this function.



Warning: *The Server Storage component of the C-ISAM DataBlade module does not support the use of ANSI databases. You must use non-ANSI databases for Server Storage tables.*

Creating Tables for Server Storage

When you create tables for the Server Storage component, the table name must match the filename argument to the **isopen()** and **isbuild()** C-ISAM functions. For example, the following function call requires that a table named `employee` exist in the currently opened database:

```
isopen( "/usr/files/payroll/employee", ISINOUT+ISEXCLLOCK ) ;
```

The Server Storage component removes any preceding path to the filename attempting to open the associated database table.

Populating the Database Tables

You might need to populate the database tables with the data from existing **.dat** files in the file system. You can use the SQL Access component of the C-ISAM DataBlade module for this task.

To populate the database tables

1. Create an SQL Access table that references existing **.dat** files in the file system. See [“Creating Database Tables for C-ISAM Data Files” on page 3-10](#).

2. Create a Server Storage database table with the same schema as the SQL Access table. Do not specify the USING CISAMDB clause and do not store the table in an extspace.

The following example shows how to create a Server Storage table called **employee_servstor** that will be used to store the SQL Access data from the **employee** table created in “[Creating Database Tables for C-ISAM Data Files](#)” on page 3-10:

```
CREATE TABLE employee_servstor (
    emp_number      INTEGER,
    emp_last_name   CHAR(20),
    emp_first_name  CHAR(20),
    emp_address     CHAR(20),
    emp_city        CHAR(20)
);
```

3. Execute the following type of SQL statement to insert all the data records into the Server Storage database table:

```
INSERT INTO server_storage_table
SELECT * FROM sql_access_table;
```

The following example shows how to insert the data from the **employee** table into the **employee_servstor** table:

```
INSERT INTO employee_servstor
SELECT * FROM employee;
```

Creating Indexes

From your C-ISAM applications, to access the data in the tables of the Server Storage component of the C-ISAM DataBlade module, you must create a special type of SQL index on the table. In particular, you must specify a dbspace storage location of **IN TABLE**, as shown in the following example:

```
CREATE INDEX i1 ON employee_servstor (emp_last_name) FILLFACTOR 90 IN TABLE;
```

Note that you must use the exact text **IN TABLE** for the storage location; this does not represent a placeholder for a dbspace name. The **IN TABLE** clause is an undocumented feature of the database server that associates the index page data with the same internal partition as the table data.



Warning: If you do not specify the `IN TABLE` clause for indexes created on Server Storage tables when creating indexes, these indexes are not updated by C-ISAM applications. This results in inconsistencies in data retrieval between C-ISAM applications and SQL access to the same table.

Database Considerations

There are subtle differences between the way standard C-ISAM applications behave and how those compiled using the Server Storage component of the C-ISAM DataBlade module behave. This section of the manual describes those differences and provides alternatives or workarounds if possible. The following topics are discussed:

- [Access Permissions](#)
- [Logging](#)
- [Locking](#)
- [Indexing](#)

Access Permissions

In standard C-ISAM applications, the permissions of the host operating system control access to data and index files. When you use the Server Storage component of the C-ISAM DataBlade module, the data is stored in tables controlled by the database server, so operating system permissions no longer apply.

To control access to these tables, you must use the permissions supplied by the database server through the SQL `GRANT` and `REVOKE` statements. See the [Informix Guide to SQL: Syntax](#) for more information on controlling access permissions.

Although database- and table-level permissions are supported by Server Storage applications, column-level permissions are not supported. For example, if a user has `SELECT` permissions on a table, they can select the entire row, regardless of any column-level permissions. This restriction must exist for C-ISAM applications to function correctly, as they always work on an entire row. Attempting to return or operate on portions of data rows from C-ISAM causes most existing applications to fail.

Logging

C-ISAM applications compiled using the Server Storage component of the C-ISAM DataBlade module have the same logging semantics as standard C-ISAM applications, with the following exceptions:

- Logging is not provided by a separate operating system file but rather by the logical logs of the database server.
- The **islogopen()** and **islogclose()** functions behave as expected, but they no longer open and close an operating system file.
- The database in which the Server Storage tables reside must have logging enabled for applications to successfully call the **islogopen()** function and support transactions.
- The **isrecover()** function is not supported. The facilities of the database server are used to recover data.

For more information on the logging facilities within the database server, see the [Administrator's Guide](#) for your database server.

Locking

Most locking semantics of traditional C-ISAM applications are preserved when using the Server Storage component of the C-ISAM DataBlade module. There are two exceptions:

- The ISWAIT option of the C-ISAM **isread()** function is ignored.
- The ISSKIPLOCK option of the C-ISAM **isread()** function is ignored.

Indexing

Indexes created using the CREATE INDEX statement must specify the IN TABLE clause as the storage location for the index. See [“Creating Indexes” on page 4-9](#).

Applications that use the C-ISAM **isaddindex()** function have some subtle differences from traditional C-ISAM applications. See [“isaddindex\(\)” on page 4-13](#).

isdatabase() Function

The Server Storage component of the C-ISAM DataBlade module contains the **isdatabase()** function to help applications integrate more easily with a database server instead of operating system files.

Use the **isdatabase()** function to open the database containing the C-ISAM tables your application will use.

Syntax

```
int isdatabase( char *dbname )
```

dbname A pointer to the name of the database to open.

Usage

You can use the **isdatabase()** function in your C-ISAM applications to open the corresponding database instead of setting the global environment variable **CISAMDB_DATABASE**. The advantage to using the **isdatabase()** function is that it can connect to the database server as any user by setting two new global variables named **isusername** and **ispassword**. Both of these variables are defined in the **isam.h** header file as follows:

```
extern char *isusername; /* user connected to database as */  
extern char *ispassword; /* password for user for connecting */
```

If either of these variables is a null pointer, the **isdatabase()** function connects to the database server as the user running the application.

Return Codes

-1	ERROR; ISERRNO contains the error code.
0	Successful

Examples

To connect to the database called **payroll@dbsrv1** as the current user, include the following code in your C-ISAM application:

```
...
ret = isdatabase( "payroll@dbsrv1" ) ;
...
```

To connect to the database called **inventory** as user **john** with a password of **new.pass**, use the following code in your C-ISAM application:

```
...
isusername = "john" ;
ispassword = "new.pass" ;
ret = isdatabase( "inventory" ) ;
...
```

C-ISAM Function Differences

This section describes the C-ISAM functions that differ from traditional C-ISAM when used with the Server Storage component of the C-ISAM DataBlade module.

isaddindex()

The **isaddindex()** function behaves the same in Server Storage applications as it does in traditional C-ISAM applications, with one difference: in C-ISAM, indexes have no name associated with them. Server Storage applications automatically generate an index name with the following format:

```
cdb<tabid>_[0..]
```

The name always begins with the prefix `cdb`, followed by the table ID of the table, finally followed by an underscore and an integer value that starts at 0 and is incremented for each new index created. The table ID corresponds to the `tabid` column of the `systables` system catalog.

For example, for a table with a table ID of 136, three indexes created using the `isaddindex()` function have the following names:

```
cdb136_0  
cdb136_1  
cdb136_2
```



Important: The `isaddindex()` function is likely to fail if used on a file created using the `isbuild()` function. This happens because the `isbuild()` function does not specify all the individual columns of the record that may be required for a `CREATE INDEX` statement to succeed. See “[isbuild\(\)](#)” on page 4-14 for more information. If this is a problem, you should create the required tables using SQL and open them using the `isopen()` function.

isaudit()

The `isaudit()` function is not currently supported. The function returns an error code of `ENOTSUPP`.

isbuild()

The `isbuild()` function differs from traditional C-ISAM only in the sense that the C-ISAM DataBlade module must create a database table instead of a file system data file. Because of this requirement, the table created by Server Storage contains only those columns required to support the primary index of the data file. The remainder of the data record is created in as few long character columns as needed.

Consider the case where your application calls the **isbuild()** function to create a file with a record length of 100 bytes and a primary key composed of two integer columns beginning at byte offsets 0 and 4 in the data file, respectively. The following CREATE TABLE statement is generated:

```
CREATE TABLE tabname (  
    col1    INTEGER,  
    col2    INTEGER,  
    col3    CHAR(92)  
);
```

You cannot create meaningful queries of this table because the remaining 92 bytes of the record are not identified as separate columns. The separation of the remaining record into table columns is not possible because the **isbuild()** function has no mechanism to provide this information.

If you want to view the record in separate columns within a table, you should use the SQL CREATE TABLE statement and open the table using the **isopen()** function instead of the **isbuild()** function.

isdelcurr()

The user executing the application that uses the **isdelcurr()** function must have delete privileges on the associated table for the DELETE statement to succeed. If the privilege is not available, an ENOPERM error code is returned.

See the [Informix Guide to SQL: Syntax](#) for more information on using the GRANT statement to grant table-level permissions.

isdelete()

The user executing the application that uses the **isdelete()** function must have delete privileges on the associated table for the DELETE statement to succeed. If the privilege is not available, an ENOPERM error code is returned.

See the [Informix Guide to SQL: Syntax](#) for more information on using the GRANT statement to grant table-level permissions.

isdelindex()

The **isdelindex()** function is not currently supported. It returns an error code of ENOTSUPP.

Use the DROP INDEX statement instead.

isdelrec()

The user executing the application that uses the **isdelrec()** function must have delete privileges on the associated table for the DELETE statement to succeed. If the privilege is not available, an ENOPERM error code is returned.

See the [Informix Guide to SQL: Syntax](#) for more information on using the GRANT statement to grant table-level permissions.

islock()

The **islock()** function has the same effect as an SQL LOCK TABLE TABLE_NAME IN SHARE MODE statement.

islogopen()

The **islogopen()** function is currently supported; however, the database must be open at the time **islogopen()** is called. The **logname** parameter to **islogopen()** is ignored.

isopen()

Use of the **isopen()** function in Server Storage applications has a number of differences from its use in traditional C-ISAM applications:

- The primary key for the opened table is found by examining the system catalogs of the currently open database. Only if a primary key constraint is found will the associated index be used for that table.
- If you do not specify either the ISNOLOG or ISTRANS parameter in the MODE flag passed to **isopen()**, the parameter defaults to ISNOLOG.

- If you specify a full or relative pathname for the table name, the pathname is truncated to use only the filename as the table name to open.

isread()

The user executing the application that uses the **isread()** function must have select privileges on the associated table for the READ statement to succeed. If the privilege is not available, an ENOPERM error code is returned.

See the [Informix Guide to SQL: Syntax](#) for more information on using the GRANT statement to grant table-level permissions.

isrecover()

The **isrecover()** function is not currently supported. It returns an ENOTSUPP error code.

isrename()

The **isrename()** function is not currently supported. It returns an ENOTSUPP error code.

Use the RENAME TABLE statement to change the name of a table.

isrewcurr()

The user executing the application that uses the **isrewcurr()** function must have update privileges on the associated table for the WRITE statement to succeed. If the privilege is not available, an ENOPERM error code is returned.

See the [Informix Guide to SQL: Syntax](#) for more information on using the GRANT statement to grant table-level permissions.

isrewrec()

The user executing the application that uses the **isrewrec()** function must have update privileges on the associated table for the WRITE statement to succeed. If the privilege is not available, an ENOPERM error code is returned.

See the [Informix Guide to SQL: Syntax](#) for more information on using the GRANT statement to grant table-level permissions.

isrewrite()

The user executing the application that uses the **isrewrite()** function must have update privileges on the associated table for the WRITE statement to succeed. If the privilege is not available, an ENOPERM error code is returned.

If there is no primary key constraint on the table that is being updated, **isrewrite()** returns an error code of ENOREC.

See the [Informix Guide to SQL: Syntax](#) for more information on using the GRANT statement to grant table-level permissions.

iswrcurr()

The user executing the application that uses the **iswrcurr()** function must have insert privileges on the associated table for the WRITE statement to succeed. If the privilege is not available, an ENOPERM error code is returned.

See the [Informix Guide to SQL: Syntax](#) for more information on using the GRANT statement to grant table-level permissions.

iswrite()

The user executing the application that uses the **iswrite()** function must have insert privileges on the associated table for the WRITE statement to succeed. If the privilege is not available, an ENOPERM error code is returned.

See the [Informix Guide to SQL: Syntax](#) for more information on using the GRANT statement to grant table-level permissions.

New Global Variables

The following new global variables are defined in the **isam.h** header file and might help to determine why particular C-ISAM functions fail.

iserrfile

The **iserrfile** global variable contains the name of the Server Storage component source code file where the last failure occurred.

Declaration

```
char *iserrfile ;
```

iserrline

The **iserrline** global variable contains the line number of the Server Storage component source code where the last failure occurred.

Declaration

```
long iserrline ;
```

issqlcode

The **issqlcode** global variable contains the SQLCODE of the last SQL statement executed by the Server Storage component.

Declaration

```
long issqlcode ;
```

Advanced Topics

In This Chapter	5-3
Setting Debugging Options	5-3
Debugging the SQL Access cisamdbd Daemon Process	5-3
CISAMDB_DUMPFILE	5-4
CISAMDB_DUMPLEVEL	5-4
Debugging SQL Access Client Database Sessions	5-5
Debugging Server Storage Applications	5-5
CISAMDB_DUMPFILE	5-6
CISAMDB_DUMPLEVEL	5-6
Using User-Defined Types	5-7

In This Chapter

This chapter provides information on some of the more advanced uses of the C-ISAM DataBlade module, including:

- [“Setting Debugging Options,”](#) next
- [“Using User-Defined Types”](#) on page 5-7

Setting Debugging Options

There are debugging options for the various components of the C-ISAM DataBlade module. You should use these debugging options only when directed to by Informix support or field personnel.

Debugging is enabled differently for each of the following DataBlade module components:

- The SQL Access **cisamdbd** daemon process
- SQL Access client database sessions
- Server Storage applications

Debugging the SQL Access **cisamdbd** Daemon Process

Debugging for the **cisamdbd** daemon process is controlled by two environment variables: **CISAMDB_DUMPFILE** and **CISAMDB_DUMPLEVEL**. You must set these variables in your environment before starting the **cisamdbd** daemon.

CISAMDB_DUMPFILE

The **CISAMDB_DUMPFILE** environment variable specifies the full path to a file that contains debugging information for the **cisamdbd** daemon process. The following example shows how to set this environment variable in the UNIX Bourne shell:

```
$ CISAMDB_DUMPFILE=/tmp/cisamdbd.dump
$ export CISAMDB_DUMPFILE
```

Data is always appended to this file, and the file is created if it does not already exist.

CISAMDB_DUMPLEVEL

The **CISAMDB_DUMPLEVEL** environment variable specifies a 4-byte hexadecimal value that determines how much debugging information to produce. The following example shows how to set this environment variable in the UNIX Bourne shell:

```
$ CISAMDB_DUMPLEVEL=0x00000043
$ export CISAMDB_DUMPLEVEL
```

The following table describes the various hexadecimal values used for the dump level.

Label	Hex. Value	Description
FUNCTION_ENTRY	0x00000001	Each function entry is logged.
FUNCTION_EXIT	0x00000002	Each function exit is logged.
LOW	0x00000004	A low amount of debugging occurs.
MEDIUM	0x00000008	A medium amount of debugging occurs.
HIGH	0x00000010	A high amount of debugging occurs.
RARE	0x00000020	A very high amount of debugging occurs, including debugging within loops, which could create considerable output.
ALWAYS	0x00000040	This bit must always be on to enable any type of debugging.

The dump value is additive; therefore, you can turn on multiple types of dump levels simultaneously by adding up the hexadecimal values of each level and setting the `CISAMDB_DUMPLEVEL` environment variable to this value.

Debugging SQL Access Client Database Sessions

To debug the code in the database server that supports SQL Access for client sessions, execute the C-ISAM DataBlade module function `cisamdb_setdebug()`.

The function takes two arguments: a pathname to a file where debugging output is written and an integer argument that indicates the level of debugging to generate.

```
EXECUTE FUNCTION cisamdb_setdebug( "/tmp/cisamdb.dump", 4 );  
SELECT * FROM employee;
```

The integer argument has a value of 0 to 4, as described in the following table.

Value	Description
0	None: disable debugging
1	Low
2	Medium
3	High
4	Rare: a large volume of debugging information is created

Debugging Server Storage Applications

Debugging for Server Storage applications is controlled by two environment variables: `CISAMDB_DUMPFILE` and `CISAMDB_DUMPLEVEL`. These variables must be set before the application is executed.

CISAMDB_DUMPFILE

The **CISAMDB_DUMPFILE** environment variable specifies the full path to a file that contains debugging information for the application. The following example shows how to set this environment variable in the UNIX Bourne shell:

```
$ CISAMDB_DUMPFILE=/tmp/appname.dump  
$ export CISAMDB_DUMPFILE
```

Data is always appended to this file, and the file is created if it does not already exist.

CISAMDB_DUMPLEVEL

The **CISAMDB_DUMPLEVEL** environment variable specifies a 4-byte hexadecimal value that specifies how much debugging information to produce. The following example shows how to set this environment variable in the UNIX Bourne shell:

```
$ CISAMDB_DUMPLEVEL=0x00000003  
$ export CISAMDB_DUMPLEVEL
```

The following table describes the various hexadecimal values used for the dump level.

Hex. Value	Description
0x00000001	All function entry points are logged.
0x00000002	All function exit points are logged.
0x00000004	All SQL statements generated are logged.
0x00000008	Values of key variables are logged.

The dump value is additive; therefore, you can turn on multiple types of dump levels simultaneously by adding up the hexadecimal values of each level and setting the **CISAMDB_DUMPLEVEL** environment variable to this value.

Using User-Defined Types

The database server allows you to create user-defined types (UDT) to handle arbitrary column values that do not map directly to the built-in data types, such as character and integer. UDTs can be used to provide an SQL interface to fields within C-ISAM files that do not have a corresponding native database type.

The directory `$CISAMDB_DIR/udtdemo` contains a set of files that demonstrates how a UDT can be created and used with the SQL Access component of the C-ISAM DataBlade module. This directory contains a README file that describes how to use the demo.



Important: Refer to the release notes of your particular version of the C-ISAM DataBlade module for information on how your version of the database server and DataBlade module support UDTs.

The cisamdbdemo Demonstration Script

You can use the script **cisamdbdemo**, located in the directory **\$CISAMDB_DIR/bin**, to create a collection of tables and C-ISAM applications that demonstrate how the C-ISAM DataBlade module operates.

To use the demonstration script

1. Set the **CISAM_LIBDIR** and **CISAM_INCDIR** environment variables to point to the directories where the standard C-ISAM header file and library exist. These variables do *not* point to the directories where the header and libraries files of the C-ISAM DataBlade module reside.

The **CISAM_LIBDIR** and **CISAM_INCDIR** variables default to **/usr/lib** and **/usr/include**, respectively. If you have your C-ISAM files installed in different locations, you should set and export these environment variables. For example, if you have the C-ISAM library installed in **/usr/informix/cisam**, you would set these variables using the UNIX Bourne shell, as shown:

```
$ CISAM_LIBDIR=/usr/informix/cisam  
$ CISAM_INCDIR=/usr/informix/cisam  
$ export CISAM_LIBDIR CISAM_INCDIR
```

2. Create a new directory to hold the files that will be built, as shown in the following example:

```
$ mkdir /usr/cisamdb/demo
```

3. Change to this directory, as shown in the following example:

```
$ cd /usr/cisamdb/demo
```

4. Make sure that the database server is online and that the **cisamdbd** daemon process is running, as shown in the following two examples:

```
$ onstat -  
INFORMIX-Universal Server Version 9.14.UC5 -- Online  
...  
$ cisamdbd -r  
13703
```

5. Execute the **cisamdbdemo** script.

With no command-line arguments, the script creates a new database called **cisamdb_demo**, and drops that database if it already exists. If you want to use a different database name, enter it as the only command-line option.

The following example shows some of the output of the **cisamdbdemo** script:

```
$ cisamdbdemo  
drop database cisamdb_demo  
  
Database dropped.  
  
create database cisamdb_demo  
  
Database created.  
  
Database closed.  
  
External space successfully dropped.  
External space successfully created.  
cbbdemo_shm>cbbdemo_shm>Registering DataBlade  
module... (may take a while).  
DataBlade cisamdb.1.00.UC1B4 was successfully  
registered in database cisamdb_de  
mo.  
cbbdemo_shm>Disconnecting...  
Copying files to current directory...  
... (remainder removed for brevity )...
```

After the **cisamdbdemo** script finishes, a directory listing shows the following list of files, which have been copied or built in your current directory:

```
$ ls
employeeLoad.c      ex1_ss* ex4_ss* ex7_ss*
employeeLoad_sa*   ex2.c  ex5.c  makefile
employeeLoad_ss*   ex2_sa* ex5_sa* makefile.sa
employee_sa.dat     ex2_ss* ex5_ss* makefile.ss
employee_sa.idx     ex3.c  ex6.c  perform_sa.dat
employee_sa.sql     ex3_sa* ex6_sa* perform_sa.idx
employee_ss.sql     ex3_ss* ex6_ss* perform_sa.sql
ex1.c               ex4.c  ex7.c  perform_ss.sql
ex1_sa*             ex4_sa* ex7_sa*
```

The directory contains seven example C-ISAM applications, **ex1_*** through **ex7_***. They have all been compiled with either an **_sa** extension for SQL Access or an **_ss** extension for Server Storage. The programs operate on two different files: **employee** and **perform**.

Four new database tables have also been created:

- **employee_ss** and **perform_ss** for Server Storage
- **employee_sa** and **perform_sa** for SQL Access

The **.sql** files in the directory contain the CREATE TABLE statements for these tables.

Finally, a program named **employeeLoad_*** has been compiled and executed to load 100 records into the **employee** and **perform** files so that there is some data for you to view.

You can now use an SQL editor to execute SQL statements against the defined tables and experiment with how the C-ISAM DataBlade module provides you with SQL access to the C-ISAM data.

Index

A

Access method, cisamdb 3-14
 AIX operating system 2-10
 ANSI compliance
 level Intro-9
 ANSI databases 1-10, 4-8

B

BladeManager 2-4
 Boldface type Intro-5

C

C-ISAM
 accessing from SQL Access 3-4
 application changes 1-8
 creating tables for 3-11
 description of 1-3
 file access permissions 3-18
 function differences when using
 DataBlade module 4-13
 global variables 4-19
 moving data into database
 tables 4-4
 recompiling applications 4-4, 4-5,
 4-6
 C-ISAM DataBlade module
 and ANSI databases 1-10
 and fragmentation 1-9
 and Global Language
 Support 1-10, 2-6
 and indexes 1-10, 3-18
 and locking 3-17
 and triggers 1-9

 and user-defined data types 5-7
 cisamdb access method 3-14
 cisamdbdemo demonstration
 script A-1
 components of 1-4
 creating a database for 2-7
 debugging 5-3
 demonstration files of 2-6
 description of 1-3
 directory structure of 2-5
 environment variables of 2-8
 function differences with C-
 ISAM 4-13
 installing 2-3
 limitations of 1-9
 registering 2-4
 Server Storage component of 1-5
 shared library files 2-6
 SQL Access component of 1-4
 supported database servers 1-9
 when to use components of 1-6
 cisamdb access method 3-14
 cisamdbd daemon
 debugging 5-3
 description of 3-3
 location of binary 2-6
 permissions 3-18
 removing shared memory and
 semaphores 3-9
 running 3-8
 setting configuring
 parameters 3-5
 starting 3-8
 stopping 3-8
 verifying that it is running 3-9
 cisamdbdemo demonstration
 script A-1

CISAMDB_DATABASE
 environment variable 2-9, 4-7, 4-12

CISAMDB_DIR environment
 variable 2-3, 2-5, 2-8, 2-11

CISAMDB_DUMPFILE
 environment variable 5-4, 5-6

CISAMDB_DUMPLEVEL
 environment variable 5-4, 5-6

cisamdb_make.inc file 4-5

CISAMDB_MAXUSERS
 configuration parameter 3-5, 3-6

cisamdb_setdebug() function 5-5

CISAMDB_TRANSFERSIZE
 configuration parameter 3-6

CISAMDB_WAITTIME
 configuration parameter 3-7

CISAM_INCDIR environment
 variable A-1

CISAM_LIBDIR environment
 variable A-1

Code, sample, conventions
 for Intro-6

Comment icons Intro-6

Committed read isolation
 level 3-17

Compliance with industry
 standards Intro-9

Configuration parameters
 CISAMDB_MAXUSERS 3-5, 3-6
 CISAMDB_TRANSFERSIZE 3-6
 CISAMDB_WAITTIME 3-7

Contact information Intro-10

Conventions,
 documentation Intro-4

Cursor stability isolation level 3-17

D

Data migration 1-7

Debugging 5-3

Default locale Intro-4

Demo directory 2-6

Dependencies, software Intro-4

directory argument of cisamdb
 access method 3-15

Dirty read isolation level 3-17

Documentation, types of
 documentation notes Intro-9

error message files Intro-8

online manuals Intro-7

printed manuals Intro-7

related reading Intro-9

release notes Intro-9

E

Environment variables
 CISAMDB_DATABASE 2-9, 4-7, 4-12
 CISAMDB_DIR 2-3, 2-5, 2-8, 2-11
 CISAMDB_DUMPFILE 5-4, 5-6
 CISAMDB_DUMPLEVEL 5-4, 5-6
 CISAMD_INCDIR A-1
 CISAM_LIBDIR A-1
 INFORMIXDIR 2-9
 INFORMIXSERVER 2-9
 INFORMIXSQLHOSTS 2-9
 LD_LIBRARY_PATH 2-10
 LIBPATH 2-10
 ONCONFIG 2-9, 3-5, 3-7, 3-8
 shared library search path 2-10
 SHLIB_PATH 2-10
 summary 2-11

en_us.8859-1 locale Intro-4

Error message files Intro-8

extspaces
 creating 3-10
 specifying for SQL Access
 table 3-14

F

filename argument of cisamdb
 access method 3-15

Find Error utility Intro-8

Fragmentation 1-9, 3-18

Functions
 cisamdb_setdebug() 5-5
 isaddindex() 1-10, 3-18, 4-11, 4-13
 isaudit() 4-14
 isbuild() 1-10, 4-8, 4-14
 isdatabase() 4-7, 4-12
 isdelcurr() 4-15

isdelete() 4-15

isdelindex() 4-16

isdelrec() 4-16

islock() 4-16

islogclose() 4-11

islogopen() 3-16, 4-7, 4-11

isopen() 3-18, 4-8, 4-16

isread() 3-17, 4-11, 4-17

isrecover() 3-16, 4-11, 4-17

isrename() 4-17

isrewcurr() 4-17

isrewrec() 4-18

isrewrite() 4-18

iswrcurr() 4-18

iswrite() 4-18

G

Global Language Support
 (GLS) Intro-4, 1-10, 2-6

Global variables 4-19

H

HP-UX operating system 2-10

I

Icons in documentation Intro-6

Important paragraphs, icon
 for Intro-6

IN clause of CREATE TABLE 3-14

IN TABLE clause 1-10, 4-9, 4-11

Indexed sequential access method
 (ISAM) 1-3

Indexes 1-10, 3-18

Industry standards, compliance
 with Intro-9

Informix C-ISAM
 See C-ISAM

informix user 2-5, 3-8, 3-9

INFORMIXDIR environment
 variable 2-9

INFORMIXSERVER environment
 variable 2-9

INFORMIXSQLHOSTS
 environment variable 2-9

Installing the C-ISAM DataBlade module 2-3

isaddindex() function 1-10, 3-18, 4-11, 4-13

ISAM 1-3

isam.h header file 2-6, 4-19

isaudit() function 4-14

isbuild() function 1-10, 4-8, 4-14

isdatabase() function 4-7, 4-12

isdelcurr() function 4-15

isdelete() function 4-15

isdelindex() function 4-16

isdelrec() function 4-16

iserrfile global variable 4-19

iserrline global variable 4-19

islock() function 4-16

islogclose() function 4-11

islogopen() function 3-16, 4-7, 4-11

ISO 8859-1 code set Intro-4

Isolation levels 3-17

isopen() function 3-18, 4-8, 4-16

isread() function 3-17, 4-11, 4-17

isrecover() function 3-16, 4-11, 4-17

isrename() function 4-17

isrewcurr() function 4-17

isrewrec() function 4-18

isrewrite() function 4-18

issqlcode global variable 4-19

iswrcurr() function 4-18

iswrite() function 4-18

J

JDBC 1-4

L

LD_LIBRARY_PATH environment variable 2-10

libisam.a library 4-3, 4-5

LIBPATH environment variable 2-10

Limitations of the C-ISAM DataBlade module 1-9

Locale Intro-4

Locking 3-17, 4-11

Logging 3-16, 4-11

M

Message file for error messages Intro-8

O

ODBC 1-4

ONCONFIG environment variable 2-9, 3-5, 3-7, 3-8

Online manuals Intro-7

onstat utility 3-8

P

Printed manuals Intro-7

R

Registering the C-ISAM DataBlade module 2-4

Related reading Intro-9

Release notes Intro-9

Repeatable read isolation level 3-17

rofferr utility Intro-8

S

Sample-code conventions Intro-6

Semaphores 3-3, 3-9

SEQUENT operating system 2-10

Server Storage component and access permissions of data

and index files 4-10

and ANSI databases 4-8

and C-ISAM application

changes 1-8

and data migration 1-7

and database locking 4-11

and database logging 4-11

and database server features 1-9

and SQL-based applications 1-8

architecture 4-3

creating database for 4-7

creating indexes for 4-9

creating tables for 4-8

database considerations 4-10

debugging 5-5

description of 1-5, 4-3

figure of 1-6

moving C-ISAM data into database tables 4-4

populating tables for 4-8

recompiling C-ISAM applications to use 4-4

referencing macro for list of libraries 4-5

when to use 1-6

Shared library files 2-6

Shared library search path environment variable 2-10

Shared memory 3-3, 3-9

SHLIB_PATH environment variable 2-10

Software dependencies Intro-4

SOLARIS operating system 2-10

SQL Access component

accessing C-ISAM data 3-4

and C-ISAM application

changes 1-8

and data migration 1-7

and database server features 1-9

and SQL-based applications 1-8

and user-defined data types 5-7

architecture 3-3

C-ISAM fields and corresponding table column types 3-13

cisamdb access method 3-14

creating extspaces for 3-10

creating tables for 3-10, 3-11

database considerations 3-16

debugging 5-5

description of 1-4

figure of 1-5

specifying a directory 3-15

specifying extspace 3-14

specifying filename 3-15

specifying maximum number of sessions 3-5

specifying transfer size of data 3-6

when to use 1-6

SQL code Intro-6

SQL-based applications 1-8

sqlhosts file 2-9

System requirements Intro-4

T

Tip icons Intro-6
Transactions 1-10, 3-16
Triggers 1-9

U

User-defined data types 5-7
Users, types of Intro-3
USING clause 3-14
Utilities
 Find Error Intro-8
 onstat 3-8
 rofferr Intro-8

V

Virtual Table Interface (VTI) 1-4

W

Warning icons Intro-6

X

X/Open compliance level Intro-9