InterBase 5

Operations Guide

InterBase®

SOFTWARE CORPORATION

100 Enterprise Way, Suite B2 Scotts Valley, CA 95066 http://www.interbase.com

1INT0055WW21002 5E4R0898

9899000102-9 8 7 6 5 4 3 2 1

D4

Table of Contents

List of Tables

List of Figures

CHAPTER

1 Introduction

The InterBase *Operations Guide* is a task-oriented reference of procedures to install, configure, and maintain an InterBase database server or Local InterBase workstation.

This chapter describes who should read this book, and provides a brief overview of the capabilities and tools available in the InterBase product line.

Who should use this guide

The Interbase *Operations Guide* is written for database administrators or system administrators who are responsible for operation and maintenance of an InterBase database server. The material is also useful for application developers who wish to understand more about the InterBase technology. The guide assumes knowledge of:

- Server operating systems including Windows NT, NetWare, and UNIX
- Networks and network protocols
- Application programming

Topics covered in this guide

- Introduction to InterBase features
- Overview of concepts of client, server, application, middleware
- General instructions on installing and licensing InterBase on Windows and UNIX platforms
- Server configuration, startup and shutdown
- Network configuration and troubleshooting guidelines
- Security configuration for InterBase servers, databases, and data; reference for the security configuration tools
- Database configuration and maintenance options; reference for the maintenance tools
- Backing up and restoring databases; reference for the backup tools
- Database and server statistics monitoring
- Interactive query profiling; reference for the interactive query tools

System requirements and server sizing

InterBase server runs on a variety of platforms, including Microsoft Window NT 4.0 and Windows 95, and several brands of UNIX operating systems. Browse the InterBase web site (**http://www.interbase.com/**) for the latest list of supported server platforms.

Windows system requirements

Operating system: Windows NT 4.0 or Windows 95

Memory: 16 megabytes minimum; 64 recommended for a server

Processor/Hardware model: 486DX2 66MHz minimum; Pentium 100MHz or greater recommended for a multi-client server

Compilers: Microsoft Visual C++ 4.2 and Borland C++ 5.0

UNIX system requirements

HP-UX

Operating system: HP-UX 10.20

HP DCE/9000 runtime support (DCE-Core) must be installed

Memory: 32 megabytes minimum; 64 recommended for a server

Processor: PA-RISC

C compiler: HP C/HP-UX Version A.10.32;

C++ compiler: HP C++/HP-UX Version A.10.22;

Fortran compiler: 10.20 release of HP Fortran/9000

Hardware Model: HP/9000 Series 7xx or 8xx

Solaris

Operating system: Solaris 2.5.*x* or 2.6.*x*

Memory: 32 megabytes minimum; 64 recommended for a server

Processor/Hardware model: SPARC or UltraSPARC

C compiler: SPARCWorks SC 4.2 C compiler

C++ compiler: SPARCWorks SC3.0.1 C++ compiler

Fortran compiler: SPARCWorks SC4.0 Fortran compiler

COBOL compiler: MicroFocus Cobol 4.0

Ada compiler: SPARCWorks SC4.0 Ada compiler

Other platforms

Information on system requirements for other operating systems supported by InterBase are not available at the time of this writing. Refer to online sources of information, including release notes included on your product media, and web pages containing platform-specific notes on the InterBase web site, http://www.interbase.com/.

Primary InterBase features

InterBase on Windows 95 and Windows NT offers all the benefits of a fully featured RDBMS. The following table lists some of the key InterBase features:

Feature	Description

Network protocol support	All platforms of InterBase support TCP/IP;
	InterBase server for Windows NT supports NetBEUI/named pipes;
	NetWare supports IPX/SPX

SQL-92 entry-level conformance	ANSI standard SQL, available through an Interactive SQL tool and Borland desktop applications
Simultaneous access to multiple databases	One application can access many databases at the same time
Multi-generational architecture	Server maintains older versions of records (as needed) so that transactions can see a consistent view of data
Optimistic row-level locking	Server locks only the individual record(s) which a client updates, instead of locking an entire database page
Query optimization	Server optimizes queries automatically, or you may manually specify query plan
Blob datatype and Blob filters	Dynamically sizeable datatypes which can contain unformatted data such as graphics and text

TABLE 1.1 InterBase features

Feature	**Description**
Declarative referential integrity	Automatic enforcement of cross-table relationships (between FOREIGN and PRIMARY KEYs)
Stored procedures	Programmatic elements in the database for advanced queries and data manipulation actions
Triggers	Self-contained program modules that are activated when data in a specific table is inserted, updated, or deleted
Event alerters	Messages passed from the database to an application; enables applications to receive asynchronous notification of database changes
Updatable views	Views can reflect data changes as they occur
User-defined functions (UDFs)	Program modules that run on the server
Outer joins	Relational construct between two tables that enables complex operations
Explicit transaction management	Full control of transaction start, commit, and rollback, including named transactions
Concurrent multiple application access to data	One client reading a table does not block others from it
Multi-dimensional arrays	Column datatypes arranged in an indexed list of elements
Automatic two-phase commit	Multi-database transactions check that changes to all databases happen before committing (InterBase Server only)
InterBase API	Functions that enable applications to construct SQL/DSQL statements directly to the InterBase engine and receive results back
gpre	Preprocessor for converting embedded SQL/DSQL statements and variables into a format that can be read by a host-language compiler
Server Manager	Windows tool for database backup, restoration, maintenance, and security
Windows ISQL	Interactive data definition and query tool for Windows
isql	Command-line version of the InterBase interactive SQL tool; can be used instead of InterBase Windows ISQL

TABLE 1.1 InterBase features (*continued*)

Feature	Description
Command-line DBA utilities	Command-line version of the InterBase database administration tools; can be used instead of Server Manager
Header files	Files included at the beginning of application programs that define InterBase datatypes and function calls
Example make files	Files that demonstrate how to invoke the makefiles to compile and link InterBase applications
Example programs	C programs, ready to compile and link, which may be used to query standard InterBase example databases on the server
Message file	*interbase.msg*, containing messages presented to the user

TABLE 1.1 InterBase features (*continued*)

SQL support

InterBase conforms to entry-level SQL-92 requirements. It supports declarative referential integrity with cascading operations, updatable views, and outer joins. InterBase Server provides libraries that support development of embedded SQL and DSQL client applications. On all InterBase platforms, client applications can be written to the InterBase API, a library of functions with which to send requests for database operations to the server.

InterBase also supports extended SQL features, some of which anticipate SQL3 extensions to the SQL standard. These include stored procedures, triggers, SQL roles, and segmented Blob support.

For information on SQL, see the *Language Reference*.

Multiuser database access

InterBase enables many client applications to access a single database simultaneously. A client applications can also access the multiple databases simultaneously. SQL triggers can notify client applications when specific database events occur, such as insertions or deletions.

You can write user-defined functions (UDFs) and store them in an InterBase database, where they are accessible to all client applications accessing the database.

Transaction management

Client applications can start multiple simultaneous transactions. InterBase provides full and explicit transaction control for starting, committing, and rolling back transactions. The statements and functions that control starting a transaction also control transaction behavior.

InterBase transactions can be isolated from changes made by other concurrent transactions. For the life of these transactions, the database appears to be unchanged except for the changes made by the transaction. Records deleted by another transaction exist, newly stored records do not appear to exist, and updated records remain in the original state.

For information on transaction management, see the *Programmer's Guide*.

Multigenerational architecture

InterBase provides expedient handling of time-critical transactions through support of data concurrency and consistency in mixed use (query and update) environments. InterBase uses a *multi-generational architecture*, which creates and stores multiple versions of each data record. By creating a new version of a record, InterBase allows all clients to read a version of any record at any time, even if another user is changing that record. InterBase also uses transactions to isolate groups of database changes from other changes.

Optimistic row-level locking

Optimistic locks are applied only when a client actually updates data, instead of at the beginning of a transaction. InterBase uses optimistic locking technology to provide greater throughput of database operations for clients.

InterBase implements true row-level locks, to restrict changes only to the records of the database that a client changes; this is distinct from page-level locks, which restrict any arbitrary data that is stored physically nearby in the database. Row-level locks permit multiple clients to update data that is in the same table without coming into conflict. This results in greater throughput and less serialization of database operations.

InterBase also provides options for pessimistic table-level locking. See the *Programmer's Guide* for details.

Database administration

InterBase provides both Windows-based and command-line tools for managing databases and servers.

You can perform database administration (DBA) on databases residing on Local InterBase or InterBase Server with Server Manager, a Windows application running on a client PC.

You can also use command-line DBA utilities on the server. For information on using the command-line DBA utilities, see the online Help.

Server Manager and the command-line utilities enable the DBA to:

- Manage server security
- Back up and restore a database
- Perform database maintenance
- View database and lock manager statistics

You can find more information on Server Manager for Windows later in this chapter, and later chapters describe individual tasks you can accomplish with Server Manager or command-line tools.

Managing server security

InterBase maintains a list of user names and passwords in a security database. The security database allows clients to connect to an InterBase database on a server if a user name and password supplied by the client match a valid user name and password combination in the security database on the server.

You can add and delete user names and modify a user's parameters, such as password and user ID.

For information about managing server security, see **Chapter 6, "Database Security."**

Backing up and restoring databases

You can back up and restore a database using Server Manager or the command-line DBA utilities. A backup can run concurrently with other processes accessing the database because it does not require exclusive access to the database.

Database backup and restoration can also be used for:

- Erasing obsolete versions of database records
- Changing the database page size
- Changing the database from single-file to multifile

- Transferring a database from one operating system to another

  Server Manager and the command-line backup tool also have an option for backing up only a database's metadata to recreate an empty database.

  For information about database backup and recovery, see **Chapter 8, "Database Backup and Restore."**

Maintaining a database

You can prepare a database for shutdown and perform database maintenance using Server Manager or command-line DBA utilities. If a database incurs minor problems, such as an operating system write error, these tools enable you to sweep a database without taking the database off-line.

Some of the tasks that are part of database maintenance are:

- Sweeping a database
- Shutting down the database to provide exclusive access to it
- Validating table fragments
- Preparing a corrupt database for backup
- Resolving transactions "in limbo" from a two-phase commit
- Validating and repairing the database structure

For information on database maintenance, see **Chapter 7, "Database Configuration and Maintenance."**

Viewing statistics

Server Manager enables the database administrator (DBA) to monitor the status of a database by viewing statistics from the database header page, and an analysis of tables and indexes. For more information, see **Chapter 9, "Database and Server Statistics."**

InterBase Specifications

This section defines the limits of a number of InterBase characteristics. The values the following table lists are design limits, and in most cases are further restricted by finite resource restrictions in the operating system or computer hardware.

Characteristic	Value
Maximum number of clients connected to one server	There is no single number for the maximum number of clients the InterBase server can serve—it depends on a combination of factors including capability of the operating system, limitations of the hardware, and the demands that each client puts on the server. Assuming a “normal” type of client application that executes database operations from human interaction, and a modern server platform (Pentium 150MHz+, 64Mb RAM), expect the InterBase server to comfortably handle up to 150 clients. This is a guideline, not a guarantee. Applications that engage in high levels of contention or which perform complex or high-volume operations may cause the practical number of clients to be fewer. Note also that some operating systems do not have the technology to serve 150 incoming network connections.
Maximum database size	The maximum addressable file size for a single file is 2Gb on Windows 95, 4Gb on Windows NT and most UNIX brands. Refer to your operating system documentation to verify file size limits. Combined with the multifile database feature of InterBase, this allows many terrabytes of addressable file space.
Maximum number of files per database	By design, 2^{16} (65,536), because the files are enumerated with an unsigned 16-bit integer. Shadow files count toward this limit. This is a design parameter of InterBase, but most operating systems have a much lower limit on the number of files that a single process can have open simultaneously. In some cases, the OS provides a means to raise this limit. Refer to your OS documentation for the default open files limit, and the means to raise this limit.
Maximum number of databases open in one transaction	No restriction. The parameters in a transaction parameter buffer comprise a linked list, so there is no limit except that imposed by system resources.

TABLE 1.2 InterBase specifications

Characteristic	Value
Maximum number of tables per database	By design, 2^{16} (65,536), because tables are enumerated with a 16-bit unsigned integer.
Maximum row size	64Kb. Each Blob and array contributes eight bytes to this limit in the form of their Blob handle. Systems tables (tables maintained by the InterBase engine for system data) have a row size limit of 128Kb.
Maximum number of rows and columns per table	By design, 2^{32} rows, because rows are enumerated with a 32-bit unsigned integer per table. Number of columns in a row depends on datatypes used. One row can be 64K. For example, you can define 16,384 columns of type INTEGER (four bytes each) in one table.
Maximum number of indexes per table	By design, 2^{16} (65,536), because indexes per table are enumerated with a 16-bit unsigned integer.
Maximum number of indexes per database	By design, 2^{32}, because there you can create 2^{16} tables per database, and each table can have 2^{16} indexes.
Maximum index key size	Starts at 256 bytes for a single-column key, and 200 for multicolumn keys; subtract four bytes for each additional column. Example: a single-column CHAR key can be up to 256 – 4 = 252 bytes; a three-column key must add up to 200 – 12 = 188 bytes. Note that multibyte character sets must fit within the key by counting bytes, not by counting characters. For example, a single-column key using 3-byte UNICODE_FSS characters can have a maximum of (256 – 4) / 3 = 84 characters.
Maximum number of events per stored procedure	No restriction by design, but there is a practical limit, given that there is a limit on the length of code in a stored procedure or trigger (see below).
Maximum stored procedure or trigger code size	48Kb of BLR, the bytecode language compiled from stored procedure or trigger language.

TABLE 1.2 InterBase specifications

Characteristic	**Value**
Maximum Blob size	The size of the largest single Blob datum depends on the database page size: 1Kb page size => 64Mb 2Kb page size => 512Mb 4Kb page size => 4Gb 8Kb page size => 32Gb The maximum Blob segment size is 64Kb.
Maximum tables in a JOIN	There is no restriction by design, though the task of joining tables is exponential in relation to the number of tables in the join. The largest practical number of tables in a JOIN is about 16, but experiment with your application and a realistic volume of data to find the most complex join that has acceptable performance.
Maximum levels of nested queries	There is no restriction by design. The practical limit depends on the type of queries you nest. Experiment with your application and a realistic volume of data to find the deepest nested query that has acceptable performance.
Maximum number of columns per one composite index	Sixteen.
Levels of nesting per stored procedure or trigger	750 on Windows 95 and Windows NT. 1000 for UNIX platforms.
Maximum size of key in SORT clause	32Kb.
Range of date values	January 1, 100 a.d. to February 29, 32768 a.d.

TABLE 1.2 InterBase specifications

Overview of Windows tools

The InterBase products on Windows 95 and Windows NT include several graphical tools with which you can perform every task necessary to configure and maintain an InterBase server, and to create and administer databases on the server.

Server Manager

This chapter introduces the InterBase Server Manager, a Windows application for monitoring and administering InterBase databases and servers. Server Manager runs on Windows, but can manage databases on any server on the local network.

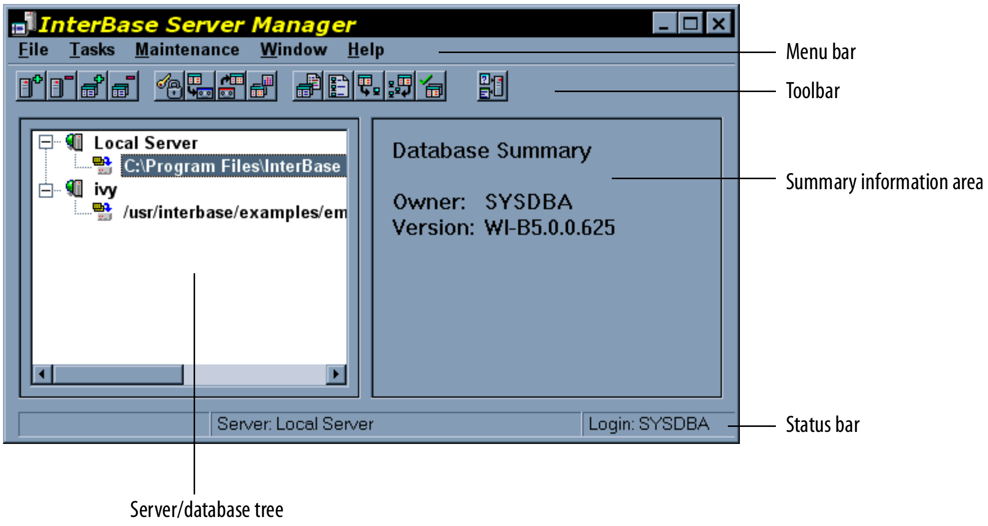
Server Manager enables you to:

- Manage server security
- Back up and restore a database
- View database and server statistics
- Perform database maintenance, including:
  - Validating the integrity of a database
  - Sweeping a database
  - Recovering transactions that are “in limbo”

The Server Manager window

To start Server Manager, choose **InterBase Server Manager** from the InterBase 5 Start menu. The Server Manager window opens:

FIGURE 1.1 Server Manager main window



Database Summary

Owner: SYSDBA
Version: WI-B5.0.0.625

Elements in the Server Manager dialog

- **Menu bar** Commands for performing DBA tasks with Server Manager
- **Toolbar** Shortcut buttons for menu commands
- **Server/database tree** Displays the server's name and the databases to which Server Manager is currently connected.
- **Summary information area** Displays information about the server or database that is selected in the server/database tree.
- **Status line** Shows the current server and user login and help for menus and the toolbar

Server Manager menus

The Server Manager menus are the basic way to perform tasks with Server Manager. There are five pull-down menus:

- File menu: enables you to log in to a server and log out, connect to a database, disconnect from a database, and exit Server Manager.

- Tasks menu: enables you to manage user security, perform backup and restoration, view database and lock manager statistics, and start Windows ISQL.
- Maintenance menu: enables you to view database connections, perform a database sweep, transaction recovery, and database shutdown, restart and validate a database, and display database properties.
- Window menu: enables you to close or minimize Server Manager windows and display the toolbar.
- Help menu: provides online Help.

Server Manager toolbar

The toolbar is a row of buttons which are shortcuts for menu commands. Table 1.3 describes these buttons:

Button	Description
	Server login: opens the login dialog, enabling you to log in to a remote InterBase server. The local server is already connected.
	Server logout: enables you to logout from the local server, and disconnect from any databases on that server to which you are currently connected.
	Database connect: opens a dialog that enables you to connect to a database on the current server.
	Database disconnect: disconnects Server Manager from the current database.
	Configure users: opens the User Configuration dialog for administering server security.
	Database backup: opens the Database Backup dialog.

TABLE 1.3 Buttons on the Server Manager toolbar

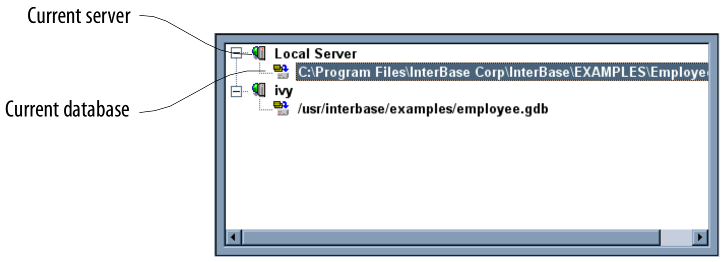
Button	Description
	Database restore: opens the Database Restore dialog.
	Database statistics: displays relevant information from the database header page.
	Database properties: opens the Database Properties dialog.
	Database connections: shows active connections.
	Database shutdown: prevents clients from connecting to a database.
	Database restart: restarts a database that has been shut down.
	Database validation: opens the Database Validation dialog.
	Start ISQL: opens the Interactive SQL Window and automatically connects to the current database.

TABLE 1.3 Buttons on the Server Manager toolbar

Server/database tree pane

When you open the Server Manager window, you must log in to a local or remote server and connect to the server's databases to display the server/database tree.

FIGURE 1.2 Server Manager tree pane


Current server
Current database
Local Server
C:\Program Files\InterBase Corp\InterBase\EXAMPLES\Employe
ivy
/usr/interbase/examples/employee.gdb

To expand or retract the server/database tree:

Tasks	Commands
Display a server's databases	Left-click (plus) + to the left of the server icon
	Double-click the server icon
	Press the plus (+) key
	Press the right arrow key
Retract a server's databases	Left-click (minus) – to the left of the server icon
	Double-click the server icon
	Press the minus (–) key
	Press the left arrow key

TABLE 1.4 Server/database tree commands

In an expanded tree, click a database name to highlight it. The highlighted database is the one on which Server Manager operates, referred to as the *current database*. When a database is highlighted, the server on which the database resides becomes the *current server*. Any actions of Server Manager then affect that server.

When you right-click the mouse on a server, a popup menu is displayed listing actions that can be performed on the server:

Popup command	**Description**
Server Login/Logout	Log in or out of a server
Database Connect	Connect to a database
User Security	Authorize users
Backup	Backup a database
Restore	Restore the database
Lock Manager Statistics	View lock manager statistics report

TABLE 1.5 Server Manager popup menu from server

When you right-click the mouse on a database, a popup menu is displayed listing actions that can be performed on the database:

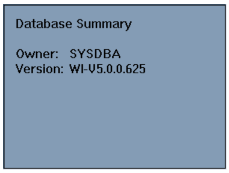
Popup command	**Description**
Database Connect/Disconnect	Connect to or disconnect from a database
Database Statistics	View database header file information
Database Connections	View active users on a database
Database Sweep	Perform a database sweep
Transaction Recovery	Recover limbo transactions
Database Shutdown	Shutdown a database
Database Restart	Restart a database
Database Validation	Verify integrity of database structures
Database Properties	View database information, adjust database sweep, enable forced writes.
Database Backup	Back up a database
Interactive SQL	Run Windows ISQL

TABLE 1.6 Server Manager popup menu from database

Summary information area

The summary information area in the right side of the Server Manger window displays information about the server or database currently selected in the server/database tree.

FIGURE 1.3 Server Manager summary information area



Database Summary

Owner: SYSDBA
Version: WI-V5.0.0.625

Standard text display window

The standard text display window is used to monitor database backup and restoration and to display database and lock manager statistics. Although these operations all display output in a standard text display window, each has menu commands specific to the particular operation.

The standard text display window contains a menu bar, a toolbar with icons for often-used menu commands, and a scrolling text display area.

InterBase Windows ISQL

The Windows ISQL tool is a graphical, interactive application that enables you to quickly and easily enter SQL statements to execute with respect to a database.This tool uses InterBase's Dynamic SQL mechanism to submit a statement to the server, prepare it, execute it, and retrieve any data from statements with output (for example, from a SELECT or EXECUTE PROCEDURE). Windows ISQL manages transactions, displays metadata information, and can produce and execute scripts containing SQL statements.

See **Chapter 10: "Interactive Query"** for full documentation and reference on **isql** and Windows ISQL.

InterBase Communication Diagnostics

Network configuration of a client/server system involves several different software and hardware layers and proper configuration of each of these layers. When one or more layers are misconfigured, it is not always evident where the problem lies. The Communications Diagnostic tool, also called ComDiag, helps to identify the source of the problem by testing each layer progressively.

See **Chapter 5: "Network Configuration"** for more guidelines on configuring and troubleshooting a client/server InterBase systems, as well as details about using ComDiag.

InterBase Server Configuration

The InterBase server has a number of configurable properties that you can change. On Windows 95 and Windows NT, you can access the graphical property sheet for the InterBase server by one of the following methods:

- Right-click the icon that appears in the task bar
- Right-click the server icon in Server Manager for Windows
- Run **InterBase Configuration** in the InterBase 5 Start menu

See **Chapter 4, "Server Configuration"** for more details on the server configuration tool.

License Registration tool

An InterBase server requires certificate IDs and keys to operate. At the time you purchase InterBase, you also purchase certificates for the type and amount of functionality you require (such as number of users). You enter one valid ID/key pair as part of the InterBase installation. To enter additional ID/key pairs, you use the InterBase License Registration Tool, located on the Start menu at **Programs | InterBase 5 | InterBase License Registration Tool** or <*ib_install_dir*>*bin**iblicense.exe*. This tool is a graphical interface to the text file (*iblicense.dat*) that contains the license strings.

The InterBase client does not require any software activation certificates. This is a change from previous versions of InterBase, in which both client and server required certificate keys.

See **"Licensing" on page 67** for more details on licensing options and tools.

Overview of command-line tools

For each task that you can do with Windows ISQL or Server Manager for Windows, there is a command-line tool that you can run in a command window or console to perform the same task.

The UNIX versions of InterBase 5 include all of the following command-line tools. The graphical Windows tools do not run on a UNIX workstation, though you can run most of the tools on Windows to connect to and operate on InterBase databases that reside on UNIX servers.

An advantage of non-interactive, command-line tools is that you can use them in batch files or scripts to perform common database operations. You can automate execution of scripts through your operating system's scheduling facility (**cron** on UNIX, **AT** on Windows NT). It is more difficult to automate execution of graphical tools.

isql

The **isql** tool is a shell-type interactive program that enables you to quickly and easily enter SQL statements to execute with respect to a database.This tool uses InterBase's Dynamic SQL mechanism to submit a statement to the server, prepare it, execute it, and retrieve any data from statements with output (for example, from a SELECT or EXECUTE PROCEDURE). **isql** manages transactions, displays metadata information, and can produce and execute scripts containing SQL statements.

See **Chapter 10: "Interactive Query"** for full documentation and reference on **isql** and Windows ISQL.

gbak and gsplit

You can run a database backup and restore on a local or remote database with **gbak**. This is the same feature that you access in Server Manager for Windows. Additionally, there is a command-line tool **gsplit** that filters the output of **gbak** and writes to multiple files on disk. This permits you to back up a multifile database when the backup file is larger than the operating system maximum.

See **Chapter 8: "Database Backup and Restore"** for full documentation and reference on use of **gbak** and **gsplit**.

gfix

You can use **gfix** to configure several properties of a database:

- Database active/shutdown status
- Default cache allocation for clients
- Sweep interval and manual sweep
- Synchronous or asynchronous writes
- Detection of some types of database corruption
- Recovery of unresolved distributed transactions

You can also use the graphical perform all the same configuration tasks with Server Manager for Windows.

See **Chapter 7: "Database Configuration and Maintenance"** for descriptions of these properties, and a reference of the **gfix** tool.

gsec

You can configure authorized users to access InterBase servers and databases with **gsec**. You can also perform the same manipulations on the security database with Server Manager for Windows.

See **Chapter 6: "Database Security"** for full details and reference.

gstat

You can use **gstat** to view some database statistics related to transaction inventory, data distribution within a database, and index efficiency. You can also view these statistics with Server Manager for Windows.

See **Chapter 9: "Database and Server Statistics"** for more information on retrieving and interpreting database statistics.

iblockpr (gds_lock_print)

You can view statistics from the InterBase server lock manager to monitor lock request throughput and identify the cause of deadlocks in the rare case that there is a problem with the InterBase lock manager. The utility is called **gds_lock_print** on the UNIX platforms, and **iblockpr** on the Windows platforms.

See **Chapter 9: "Database and Server Statistics"** for more information on retrieving and interpreting lock statistics.

ibmgr

On UNIX servers, use the **ibmgr** utility to start and stop the InterBase server process. See the section **"UNIX daemon" on page 80** for details on using this utility.

Migrating from InterBase 4 to InterBase 5

This section describes issues that affect upgrades of InterBase applications and databases designed for pre-Version 5 server software to the new InterBase 5 version.

Backward compatibility

The InterBase 5 software can access InterBase 4.0, 4.1, or 4.2 databases transparently. You do not need to modify the database for it to work with InterBase 5. However, certain new features in InterBase 5 are not active except on a newly created or restored database.

ODS changes

The On-Disk Structure (ODS) for InterBase 5 has been updated to ODS version 9.1, which supports cascading referential integrity, index garbage collection, and SQL roles.

InterBase 5 supports ODS 8.2 and ODS 9.1. Newly created databases are always ODS 9.1. Databases created by InterBase 4.0, 4.1, and 4.2 are in ODS 8.0 format are automatically converted to ODS 8.2 the first time a Version 5 engine accesses them. An ODS 8.2 database is backward-compatible with an InterBase 4.*x* server. ODS 8.*x* databases are not transparently upgraded to ODS 9.1.

To use the new ODS 9.1 features on an ODS 8.*x* or older database, use the Version 4.*x* or Version 5 **gbak** to back up the database. Then use the Version 5 **gbak** to restore it. The process of restoring a database is akin to creating one, so the database is restored as an ODS 9.1 file.

Accessing Version 3 databases

The Bridge was a feature of InterBase V4.0 server that allowed that software to access Interbase Version 3.3 databases (ODS 7.*x*). The Bridge is not included in InterBase 5. If you have a Version 3.3 server that you are upgrading to InterBase5, you *must* use **gbak** to back up your databases before you install the new software. After the install, use **gbak** *from the new version* to restore the databases.

New keywords

Do not use keywords to name metadata objects. Such names conflict with SQL syntax. For example, if you name a table CASCADE, the statement:

```
SELECT * FROM CASCADE
```

would confuse the InterBase 5 SQL parser, because CASCADE is a new keyword.

The following reserved keywords are new since InterBase 4.0:

ACTION	CASCADE	ROLE
ADMIN	RESTRICT	FREE_IT

TABLE 1.7 New keywords

If your current database contains a metadata object that has any of these names, you cannot use these objects by name in SQL statements, because their names are now reserved words. You need to drop the objects or change their names. To change the names without using their name in an SQL statement, you need to alter the names by issuing UPDATE statements against the InterBase system tables. Refer to the *Language Reference*.

Runtime environment changes

There are a number of runtime changes in InterBase 5.

- The file *interbase.cfg* no longer exists. Configuration that was placed in that file is now found in *isc_config* (UNIX) or *ibconfig* (Windows).
- The format for temporary file configuration has been changed. See **"Temporary file management" on page 92**.
- There is a new environment variable, INTERBASE_TMP. It has priority over the TMP environment variable, but not over *isc_config* (UNIX) or *ibconfig* (Windows) entries for TMP_DIRECTORY. See**"Environment variables" on page 91** for more information.
- On UNIX, InterBase itself now runs as a daemon on the server instead of using the turnkey daemon *inetd* run the server processes. InterBase 5 needs no entry in the */etc/inetd.conf* file. Because multiple daemons may not listen on the same port, you *must* not run the InterBase 5 daemon and an **inetd** daemon based on the InterBase 4.0 entry in */etc/inetd.conf* at the same time.

Preprocessor changes

The **gpre** preprocessor now generates code with a directive to include *ibase.h* instead of *gds.h*. If embedded SQL programmers use functions with prefixes of the form "**gds_**", these are not recognized by the *ibase.h* header file, and compilation fails. The fix is to put a directive in your embedded SQL code file:

```
#include '/usr/interbase/include/gds.h'
```

TIP The "**gds_**" nomenclature is obsolete and continues to phase out of the InterBase product in future versions. You should plan to change occurrences of such names of functions and other InterBase identifiers in your code. In almost all cases, a given function name such as **gds_$attach_database( )** has changed only in its prefix, becoming **isc_attach_database( )**. The one exception is **gds_$event_wait( )**, which is now **isc_wait_for_event( )**.

Below is an example **sed** script to change these identifiers:

```
s/\<gds_\$event_wait/isc_wait_for_event/g
s/\<gds_[_$]?/isc_/g
```

Licensing changes

- The requirement for an international license has been removed. International components no longer require licensing separate from the InterBase license.
- C and C++ language license options for the **gpre** preprocessor are now separate licenses from the InterBase license.
- The license registration utility on UNIX, **isc_install**, has been replaced by a similar program, **iblicense**. See **"License registration utility for UNIX" on page 69**.

Unsupported features

The following utilities have changed or are no longer supported in InterBase 5:

- **gdef**, **qli**, **fred**, and **pyxis** are not supported, but are included in the UNIX distribution kit for backward compatibility. They behave exactly as they did in InterBase V3.3 and V4.0. They have not been updated to support newer InterBase features, such as stored procedures and SQL roles.
- **gbak** no longer backs up triggers from Version 2 databases.
- **gdef** no longer extracts triggers from Version 2 databases.
- **gdef** does not output views from Version 3 databases in the optimal order for reuse in an input script, so the script may need editing to reorder view definitions.
- **gdef** requires that secondary files (external files, shadow files, and multifile databases) be fully qualified during definition in order to be created properly. In the absence of fully qualified path name, the behavior is undefined.
- The GDML Security Classes feature for assigning data access privileges to groups of users does not work in InterBase 5. In InterBase 5, SQL features assist in migrating InterBase users from GDML to SQL. You need to re-implement group privileges using the InterBase 5 features. For more information on SQL roles, see the *Language Reference* and *Data Definition Guide*.

Obsolete components

The following components have been removed from SuperServer (multi-threaded) versions of the InterBase product:

- **gds_drop** has been removed because it is not necessary with the InterBase 5 SuperServer architecture. In InterBase V4.0 and earlier, **gds_drop** deallocated the shared memory and semaphores that were used for communication between the database server processes, such as lock header. Since there is only one database server process in InterBase 5, there is no interprocess communication necessary and hence no allocated semaphores to deallocate.

  The preferred way to shut down a running server and release server resources is:

  ```
  ibmgr -shut
  ```

  See the section **"UNIX daemon" on page 80**.

  Note Shutting the server down does not release shared memory. You can use the UNIX command **ipcrm** to release these shared memory segments. Rebooting the server also releases shared memory.

- **gds_relay** has been removed for reasons similar to those for **gds_drop**. **gds_relay** was used in Classic architecture to relay signals between processes that couldn't signal each other directly. Since SuperServer runs only a single instance of the engine, there's no need for client processes to signal each other.
- **gds_inet_server** has been removed. The SuperServer process directly listens to and handles incoming TCP/IP connections. Therefore, you do not need an entry in */etc/inetd.conf*, but you still need an entry in */etc/services*. The InterBase 5 installation script performs these edits, including removing any "gds" entry in */etc/inetd.conf*.

  These components are still present on versions of InterBase that use the "Classic" architecture, such as those for SCO and Linux. Refer to the *Release Notes* for your platform if you are running the Classic implementation of InterBase.

CHAPTER 2

Client/Server Concepts

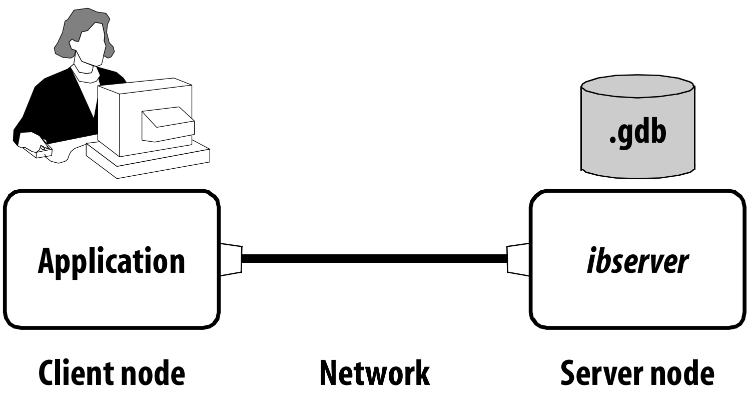
This chapter describes in high-level language the architecture of client/server systems using InterBase and InterClient. The chapter covers topics including definition of client and server, and options for application development.

Definition of a client

An InterBase client is an application, typically written in C, C++, Delphi or Java, that accesses data in an InterBase database.

In the more general case, an InterBase client is any application process that uses the InterBase client library, directly or via a middleware interface, to establish a communication channel to an InterBase server. The connection can be *local* if the application executes on the same node as the InterBase server, or *remote* if the application must use a network to connect to the InterBase server.

FIGURE 2.1 Basic client/server relationship



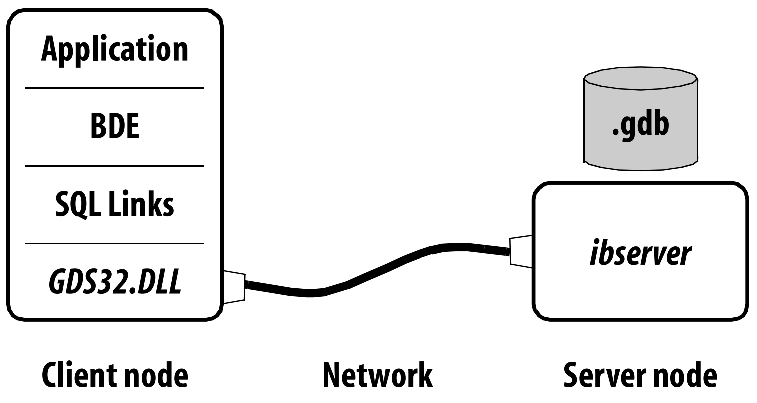
Note InterBase is designed to allow clients running one operating system to access an InterBase server on a different platform and operating system from the client. For example, a common arrangement is to have inexpensive Windows 95 PCs acting as client workstations to concurrently access a departmental server running Windows NT, NetWare, or any of several vendors of UNIX.

The InterBase client library

The InterBase client library is a library that developers of client applications use to initiate connections to a server and to programmatically perform database operations. The library uses the operating system client network interface to communicate with one or more InterBase servers, and implements a special InterBase client/server application protocol on top of a network protocol (see **"Network protocols" on page 97**).

The client library provides a set of high-level functions as an Application Programmer's Interface (API) for communication with an InterBase server. Any client application or middleware must use the API to access an InterBase database. The *API Guide* provides reference documentation and guidelines for using the API to develop high-performance applications.

FIGURE 2.2 Role of the InterBase client library



Definition of a server

The InterBase server is a software process that executes on the node that hosts the storage space for databases. The server process is the only process on any node that can perform direct I/O to the database files.

Clients send to the server process requests to perform several different types of actions on the database, including:

- Search the database based on criteria
- Collate, sort and tabulate data
- Return sets of data
- Modify data values
- Insert new data into the database
- Remove data from the database
- Create new databases or data structures
- Execute procedural code on the server
- Send messages to other clients currently connected

The server process is fully network-enabled; it services connection requests that originate on another node. The server process implements the same InterBase application protocol that the client uses.

Many clients can remain connected to the multithreaded server process simultaneously. The server regulates access to individual data records within the database, and enforces exclusive access to records when clients request to modify the data in the records.

InterClient and InterServer

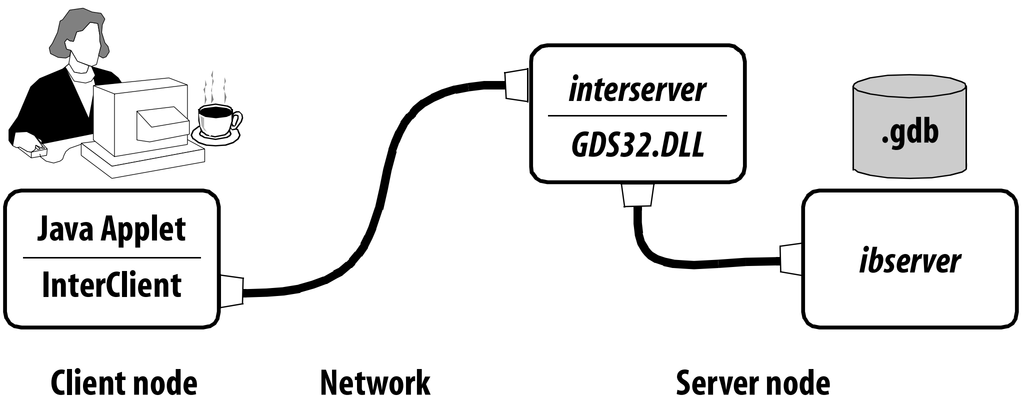
InterClient is a networked driver for Java development. It incorporates a JDBC remote protocol for exchanging and caching data between client and server. This allows for a browser enabled client with *no* preinstalled client libraries to access InterBase data across the net.

The InterClient driver is comprised of two pieces:

- The InterClient client-side driver is a thin, JDBC-compliant, all-Java class package. Applets can download the InterClient classes as a part of the applet, thus enabling dynamically distributed applets.
- The InterServer daemon runs on an NT or UNIX server. The InterClient client-side driver communicates over a network to the InterServer daemon. In turn, the InterServer process acts like an application, accessing the InterBase server via the InterBase client API. Typically, InterServer runs locally on the same node as the InterBase server, but it can also run on a separate node from the actual database server. InterServer needs the InterBase client API to function.

For more information on installing the InterClient classes and developing Java applications and applets, see InterClient's online documentation. InterClient is included with the InterBase 5 product, and updates to the InterClient classes and documentation are on InterBase's web site (http://www.interbase.com/).

FIGURE 2.3 InterClient and InterServer client/server relationship



Application development

Once you create and populate a database, you can access the information through an application. If you use one of Borland's client tools, you can access information through your existing application. You can also design and implement a new application by embedding SQL statements or API calls in an application written in a programming language such as C or C++.

Borland client tools applications

Client/server versions of Borland client tools such as Delphi, Borland C++, Paradox, and Visual dBASE can access InterBase databases using Borland SQL Links. Server query reporting is built into the client tool providing Windows application support. This enables you to build sophisticated, user-friendly database applications with minimal programming effort.

The Borland Database Engine

Most Borland application development tools use middleware technology based on the *Borland Database Engine* (BDE). The BDE is a library that provides a unified API for applications to interface programmatically with the database client library of any database vendor for which there is an *SQL Links* driver available. For instance, a C++ application programmer uses the BDE functions to access data from a BDE *alias*. The programmer configures the BDE alias to use the InterBase driver for SQL Links, and this configuration leads BDE to dynamically load the appropriate library that implements BDE functions with equivalent functions in the InterBase API.

The most important advantage is that application engineers can write code that is independent from a given proprietary database product API, and thereby reduce porting expense if project requirements call for the engineer to change database server technology. For instance, porting an application from using Paradox tables to an InterBase database can be accomplished in large part simply by reconfiguring the BDE alias to use the appropriate SQL Links driver, and specifying the path of the InterBase server and database.

The BDE technology has an internal architecture that implements features to accommodate database technologies that do not offer those features.

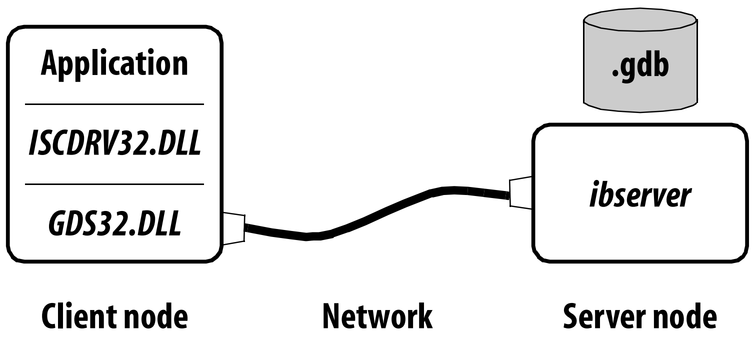
- The interaction between BDE's caching and InterBase's own caching (see **"Configuring the SuperServer cache" on page 140**) is confusing. Client-side caching gives a lot of benefit with little associated cost when the database resides on the same machine as the client, and the volume of data is low. Applying client-side caching in a client/server system with datasets that are greater in size by orders of magnitude can result in poor network performance as the client refreshes its cache over a network.
- The differences between the BDE's Local SQL interpreter and InterBase's server-side SQL interpreter are also subtle. For consistency's sake, you should configure applications to pass SQL statements through the BDE and on to the server's SQL interpreter.

Browse the InterBase web site (http://www.interbase.com/) where over time you can find more suggestions for configuration of BDE to interact with InterBase in the most efficient manner.

ODBC

Microsoft's standard, similar in intent to the BDE, is called Open Database Connectivity (ODBC). One standard API provides a unified interface for applications to access data from any data source for which an ODBC driver is available. The InterBase client for Windows NT and Windows 95 includes a 32-bit client library for developing and executing applications that access data via ODBC. The driver is in the file *iscdrv32.dll*.

FIGURE 2.4 Role of the ODBC driver



Similarly to BDE, you configure a data source using the ODBC Administrator tool. If you need to access InterBase databases from third party products that do not have InterBase drivers, you need to install this ODBC driver. The install program then asks you if you want to configure any ODBC data sources. "Configuring" means providing the complete path to any databases that you know you will need to access from non-InterBase-aware products, along with the name of the ODBC driver for InterBase.

ODBC is the common language of data-driven client software. Some software products make use of databases, but do not yet have specific support for InterBase. In such cases, they issue data queries that conform to a current SQL standard. This guarantees that these requests can be understood by any compliant database. The ODBC driver then translates these generic requests into InterBase-specific code. Other ODBC drivers access other vendors' databases.

Microsoft Office, for example, does not have the technology to access InterBase databases directly, but it can use the ODBC driver that is on the InterBase CDROM.

You do not need to install an ODBC driver if you plan to access your InterBase databases only from InterBase itself or from products such as Delphi, C++Builder, and JBuilder that use either native InterBase programming components or Borland SQL-Links components to query InterBase data.

Configuring an ODBC driver

To access the ODBC Administrator on Wintel machines, display the Control Panel and choose ODBC (in some cases, it appears as “32-Bit ODBC Administrator”).

Embedded applications

You can write your own application using C or C++, or another programming language, and embed SQL statements in the code. You then preprocess the application using gpre, the InterBase application development preprocessor. gpre takes SQL embedded in a host language such as C or C++, and generates a file that a host-language compiler can compile.

The preprocessor matches high-level SQL statements to the equivalent code that calls functions in InterBase’s client API library. Therefore, using embedded SQL affords the advantages of using a high-level language, and the runtime performance and features of the InterBase client API.

For more information about compiling embedded SQL applications, see the *Programmer’s Guide*.

Predefined database queries

Some applications are designed with a specific set of requests or tasks in mind. These applications can specify exact SQL statements in the code for preprocessing. The **gpre** preprocessor translates statements at compile time into an internal representation. These statements have a slight speed advantage over dynamic SQL, since they do not need to incur the overhead of parsing and interpreting the SQL syntax at runtime.

Dynamic applications

Some applications may need to be able to handle *ad hoc* SQL statements entered by users at run time; for example, allowing a user to select a record by specifying criteria to a query. This requires that the program construct the query based on user input.

InterBase uses Dynamic SQL (DSQL) for generating dynamic queries. At run time, your application passes DSQL statements to the InterBase server in the form of a character string. The server parses the statement and executes it.

BDE provides methods for applications to send DSQL statements to the server and retrieve results. ODBC applications rely on DSQL statements almost exclusively, even if the application interface provides a way to visually build these statements. For example, Query By Example (QBE) or Microsoft Query provide convenient dialogs for selecting, restricting and sorting data drawn from a BDE or ODBC data source, respectively.

You can also build templates in advance for queries, omitting certain elements such as values for searching criteria. At run time, supply the missing entries in the form of *parameters* and a buffer for passing data back and forth.

For more information about DSQL, see the *Programmer's Guide*.

API applications

The InterBase API is a set of functions that enables applications to construct and send SQL statements to the InterBase engine and receive results back. All database work can be performed through calls to the API.

While programming with the API requires an application developer to allocate and populate underlying structures commonly hidden at the SQL level, the API is ultimately more powerful and flexible. Applications built using API calls offer the following advantages over applications written with embedded SQL:

- Control over memory allocation
- Simplification of compiling procedure—no precompiler
- Access to error messages
- Access to transaction handles and options

API functions can be divided into seven categories, according to the object on which they operate:

- Database attach and detach
- Transaction start, prepare, commit, and rollback
- Blob calls
- Array calls
- "Database Security" handling
- Informational calls
- Date and integer conversions

The *API Guide* has complete documentation for developing high-performance applications using the InterBase API.

Multi-database applications

Unlike many relational databases, InterBase applications can use multiple databases at the same time. Most applications use only one database, but others may need to use several databases which have the same or a different structure.

For example, each project in a department may have a database to keep track of its progress, and the department may need to produce a report of all the active projects. Another example where more than one database would be used is where sensitive data is combined with generally available data. One database could be created for the sensitive data with access to it limited to a few users, while the other database could be open to a larger group of users.

With InterBase you can open and access any number of databases at the same time. You cannot join tables from separate databases, but you can use cursors to combine information. See the *Programmer's Guide* for information about multi-database applications programming.

CHAPTER 3

Installation

This chapter gives detailed instructions for installing InterBase and InterClient on Windows 95, Windows NT, Solaris, and HP-UX.

InterBase on other platforms may install differently due to operating system requirements. Refer to installation notes included in the software package, on the media, or on the InterBase web site (http://www.interbase.com/) for platform-specific installation instructions that supersede the content in this chapter.

Installation on Windows NT and Windows 95

Preparing to install on Windows

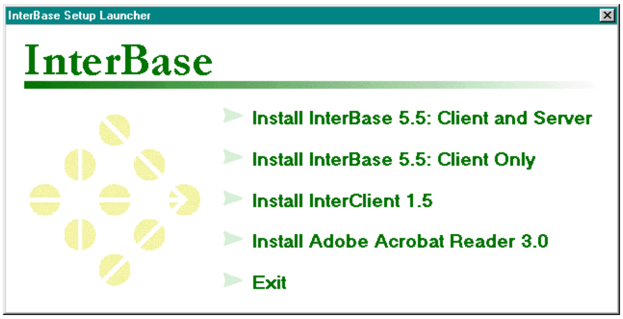
It is recommended that you uninstall any previous versions of InterBase that are on the server where you are installing the InterBase 5 product. With this software, as with any software, it is recommended to remove any field test versions prior to installing. If InterBase 4.1 or earlier is present on your server, you *must* uninstall it before proceeding with the Version 5 install.

- There must be one and only one copy of *gds32.dll* on the machine where InterBase 5 client or server is running. If this file exists on the server, you must remove it before installing InterBase 5.
- You cannot install InterBase onto a network (mapped) drive.
- You must be logged in as a user with Administrator privileges to install or uninstall InterBase on Windows NT.

Installing InterBase on Windows

Windows system requirements: Windows NT 4.0 with Service Pack 3 or Windows 95 with Service Pack 1. Installation of the Client and Server without the PDF document set requires approximately 11MB of disk space. Installation requires 36 MB of disk space if you include Adobe Acrobat Reader and the full doc set in PDF format.

InterBase Setup Launcher


InterBase Setup Launcher
InterBase
Install InterBase 5.5: Client and Server
Install InterBase 5.5: Client Only
Install InterClient 1.5
Install Adobe Acrobat Reader 3.0
Exit

When you place the InterBase CD-ROM is the drive and close the drive, the InterBase Setup Launcher displays on your screen if you have Autoplay enabled. You can always install by executing *Setup.exe* at the root of the CD-ROM. The Launcher is a menu of different software packages you can install from the CD-ROM. You install only one element at a time. When the install of that element is complete, the menu screen reappears and you can choose the next element or choose to exit. The options are:

- InterBase 5.5: Client and Server

  The fully functional InterBase Server, which you can run on Windows NT or Windows 95. You can select optional components later in the wizard, including database client and server tools, ODBC driver, example database, client application development files, and online documentation.

- InterBase 5.5: Client Only

  As above, but without the server component.

- InterClient 1.5

The high-performance JDBC driver for InterBase, with InterServer the accompanying server-side component, Java application development classes, and web server deployment kit.

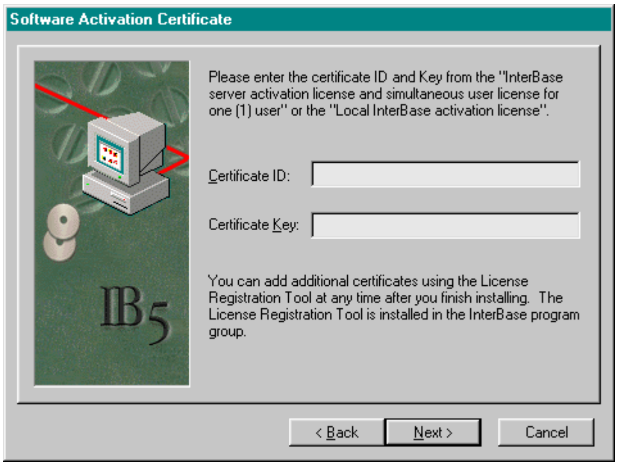
- Adobe Acrobat Reader 3.0.1

  Acrobat Reader With Search for using the online InterBase documentation and performing full-text searches on the complete doc set.

InterBase Client and Server Setup

1. To install the client and server, choose "InterBase 5.5 Client and Server" from the Launcher. This displays a welcome to the InterBase Server installation wizard. Read the screen and then choose Next.
2. Read the *Install.txt* document that displays in the Important Installation Information window. Choose Next to proceed to the next step.
3. You must read and accept the terms of the Software License Agreement. Choose Yes to continue the installation or No to exit.
4. In the Software Activation Certificate window, enter a valid Certificate ID and its corresponding Certificate Key in the appropriate fields to activate the InterBase server. These numbers are on the Software Activation Certificates that you purchased through your sales representative.

   If you are installing an evaluation copy, type `eval` in the Certificate ID field.


Software Activation Certificate

Please enter the certificate ID and Key from the "InterBase server activation license and simultaneous user license for one (1) user" or the "Local InterBase activation license".

Certificate ID:

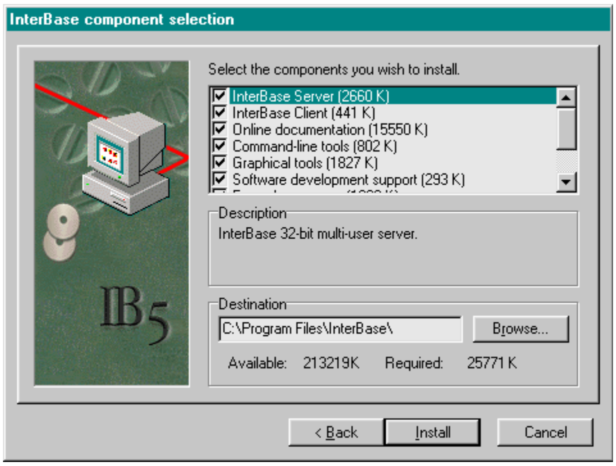
Certificate Key:

You can add additional certificates using the License Registration Tool at any time after you finish installing. The License Registration Tool is installed in the InterBase program group.

   Note Use the License Registration tool to enable additional functionality by adding other software activation keys. See **The InterBase License Registration Tool for Windows on page 68** for more information about this topic.

   Choose Next to display the InterBase Component Selection window.

5. Select components from the list in this window to customize your InterBase installation to your needs. To get more information about a choice, highlight it and read the Description panel.


InterBase component selection
Select the components you wish to install.
☑ InterBase Server (2660 K)
☑ InterBase Client (441 K)
☑ Online documentation (15550 K)
☑ Command-line tools (802 K)
☑ Graphical tools (1827 K)
☑ Software development support (293 K)
Description
InterBase 32-bit multi-user server.
Destination
C:\Program Files\InterBase\
Browse...
Available: 213219K Required: 25771 K
< Back
Install
Cancel
IB5

   Specify a destination directory for installing the software. It must be on a drive local to the machine: it cannot be on a networked drive.

   Choose Install. InterBase Setup installs both the InterBase client and your selected server components. It then displays a final window where you can choose to display the Readme file.

6. Choose Finish. The Setup Launcher redisplays so that you can install another element, such as InterClient or Acrobat Reader. The complete InterBase document set with full-text search is available online in Adobe Acrobat format. If you choose Online Documentation as an install component, they are present on your hard drive in the <*ib_install_dir*>*doc* directory. If you do not install them, you can still read them from the *doc* directory of the InterBase CD-ROM.

Installing only the InterBase Client

To install only the InterBase client, choose "Install InterBase 5.5 Client Only" from the launcher. The install procedure is identical to that for the Client and Server with one exception: Setup does not ask you for a Certificate ID/key.

Configuring InterBase

You can control many aspects of how InterBase runs, as well as choosing to configure an ODBC driver or adding software activation certificates. See **Chapter 4, "Server Configuration"** in this manual for information on these topics.

Logging on

To attach to any database, you must have a user name and password. When you first install InterBase, there is one user defined. That user is SYSDBA and the password is *masterkey*. The SYSDBA user has special privileges that override normal SQL security and there are a number of database maintenance tasks that only SYSDBA can perform. You should change the SYSDBA password immediately after installing InterBase.

See **"Modifying user configurations"** for how to change a password and**"Adding a user"** for how to create additional users.

Installing InterClient on Windows

Windows system requirements: Windows NT 4.0 or Windows 95 operating system and approximately 2,216Kb of disk space for a full install.

InterClient installation

This screen appears after you select "InterClient 1.*x*" from the InterBase Setup Launcher. This is a welcome to the InterClient installation wizard.

Software license agreement

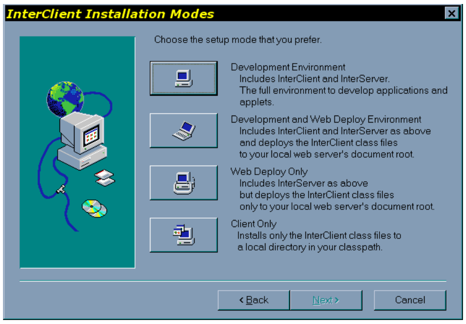
You must read and accept the terms of the license agreement, or else the setup wizard will close.

Installation notes

This screen contains important information on installing InterClient. It is recommended to read this file carefully.

InterClient installation modes

During installation you are prompted to choose one of the following installation modes:



Choose the setup mode that you prefer.

Development Environment
Includes InterClient and InterServer.
The full environment to develop applications and applets.

Development and Web Deploy Environment
Includes InterClient and InterServer as above
and deploys the InterClient class files
to your local web server's document root.

Web Deploy Only
Includes InterServer as above
but deploys the InterClient class files
only to your local web server's document root.

Client Only
Installs only the InterClient class files to
a local directory in your classpath.

1. Development Environment

   This selection installs the full development environment.

   - InterClient classes are installed for local JDBC development

- InterServer is installed for local or remote access to an InterBase server

2. Development and Web Deploy Environment

   This selection installs the full development environment as above. In addition, the InterClient classes and documentation are copied to the document root directory of your web server.

   - InterClient classes are installed for local JDBC development
   - InterServer is installed for local or remote access to an InterBase server
   - In addition, InterClient classes are copied to the document root directory of your Web server

3. Deploy to Web Server Only

   This selection installs the InterClient classes and documentation only to the document root directory of your Web server.

   - InterServer is installed for use by local or remote clients
   - InterClient classes are copied to the document root directory of the Web server

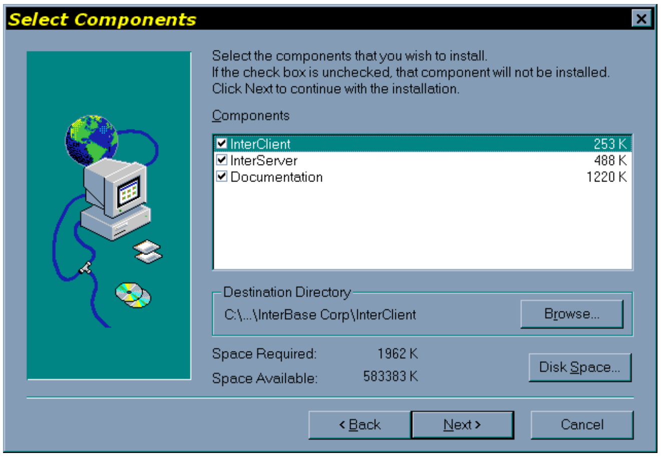
4. Client Development Only

   This selection installs the InterClient classes and documentation on the local client for use in local Java application development against a remote server with a separately installed InterServer.

   - InterClient classes are installed for local development.

Select components

This screen allows you to customize your InterClient installation by selecting components of the product to install. Refer to the Components window in this screen for descriptions of the components.



Select the components that you wish to install.
If the check box is unchecked, that component will not be installed.
Click Next to continue with the installation.

Components

☑ InterClient	253 K
☑ InterServer	488 K
☑ Documentation	1220 K

Destination Directory
C:\...\InterBase Corp\InterClient

Space Required: 1962 K
Space Available: 583383 K

Choose HTTP server class directory

If you chose web server deployment as part of installation mode 2 or 3 above, this screen permits you to specify the document root directory for your web server.

InterClient installation

This screen reminds you that you need the InterBase client installed in order to use InterServer.

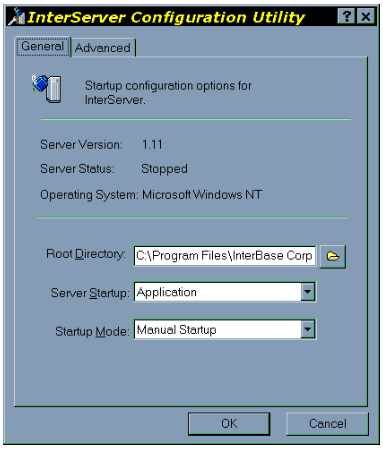
Start copying files

This screen reviews the settings you have chosen in previous screens. If you are satisfied with the choices, click **Next**. If you wish to change any of the choices, click **Back**.

After you click **Next**, InterClient begins to install to your hard disk. Included in this process, if you had chosen the components, are the InterClient Java class files, the **interserver** executable, and InterClient documentation.

InterServer configuration

This screen allows you to run the InterServer configuration utility, to set the InterServer process to run as an application or service, and to start manually or at Windows startup.



Startup configuration options for InterServer.

Server Version: 1.11
Server Status: Stopped
Operating System: Microsoft Windows NT

Root Directory: C:\Program Files\InterBase Corp
Server Startup: Application
Startup Mode: Manual Startup

View README file

It is recommended to read the introductory notes about InterClient that appear in the README file.

Setup complete

This screen notifies you that the setup wizard has completed. Which you click **Finish**, you return to the InterBase Setup Launcher.

Uninstalling InterBase on Windows

To uninstall InterBase and InterClient, use the **Control Panel | Add/Remove Programs** applet. Choose InterBase Server 5.*x* or InterBase InterClient or both in turn and click **Add/Remove** button.

- The InterBase Server, Guardian, and InterServer processes must not be running when you uninstall the software. To stop one of these applications, right-click its icon in the Task Tray and selecting Shutdown. To stop one of these services (Windows NT), use the **Control Panel | Services** applet.
- You must be logged in as a user with Administrator privileges to install or uninstall InterBase on Windows NT.
- Uninstall never removes *isc4.gdb* or files created by the server process, including *interbase.log*, *<host>.evn*, *<host>.lck*.
- The Windows InterBase installation allows you to choose between performing a complete install or selecting individual components. If you choose to install components at different times, the uninstall program removes only the components selected for the last install.
- The ODBC driver or the ODBC driver manager must be removed manually; there is no uninstall available through the Windows control panel.

Installation on UNIX

This section provides instructions for installing InterBase 5 on Solaris and HP-UX. For platform-specific instruction on installing InterBase on other UNIX brands, see the InterBase web site (http://www.interbase.com/).

Preparing to install on UNIX

1. Save older databases

   InterBase 5 uses a new On-Disk Structure (ODS), ODS 9.1. Databases created with InterBase 4.0 used ODS 8.0. To take advantage of new InterBase 5 features, you must use **gbak** to back up any databases that you intend to use with the Version 5 software. V3 databases must be backed up using the V3 **gbak**. V4 databases can be backed up using V4 **gbak** or with the InterBase 5 **gbak** if you are performing the backup after the install. Once a database has been converted to ODS 9.1, it cannot be converted back to earlier versions of the ODS.

2. Save customization files

   If you have InterBase 4.0 installed on the server and you want to preserve the customization files, copy them to a safe place, for example:

   ```
   gbak –b /usr/interbase/isc4.gdb /var/tmp/isc4.gbak
   cp /usr/interbase/isc_license.dat /var/tmp
   cp /usr/interbase/isc_config /var/tmp
   ```

   You can skip this step if you haven’t customized these files in a previous installation.

   Note When you use **pkgrm** remove InterBase V4.0 for Solaris, these files are automatically saved in */usr/tmp*.

3. Save older versions

   If InterBase 4.0 is running on your server, shut it down. To save the current version, rename the directory, for example:

   ```
   gds_drop –a
   mv /usr/interbase /usr/interbase.save
   ```

4. Point to the install directory

   If you install into a directory other than */usr/interbase*, you can point to the install directory either by defining the INTERBASE environment variable before installation, or through the use of a symbolic link. The install script looks to see if the INTERBASE variable is defined. If it is, the installer does not create any symbolic link. If it does not find the INTERBASE variable defined, it creates a symbolic link from */usr/interbase* to the install directory.

   Note If you choose to define the INTERBASE variable, be sure to define it in a location where it can be read by all users of the software. The global shell initialization files (*/etc/profile* and */etc/.login* on Solaris, for example) are appropriate places to put this setting.

5. Platform-specific issues

   HP-UX Installation requires the HP-UX 10.20 operating system. Furthermore, the following patches are strongly recommended to correct operating system defects:

   - PHKL_8376 (s700), PHKL_8377 (s800)
   - PHSS_10565 (s700_800)
   - PHNE_13265 (s700), PHNE_13266 (s800)
   - PHNE_13469 (s700), PHNE_13468 (s800)
   - PHKL_13611 (s700), PHKL_13612 (s800)
   - PHCO_13626 (s700_800)

The HP DCE/9000 runtime support (DCE-Core) must be installed.

Solaris InterBase requires Solaris versions 2.5.*x* or 2.6.*x*.

Installing on Solaris

1. Log in to your database server as *root*.
2. Load the CD-ROM with the InterBase 5 product. Mount the CD-ROM at an appropriate mount point. If you have the volume manager running, this is done for you, and the CD-ROM mounts according to its label. For instance */cdrom/interbase_sos_V511* for the CD-ROM of InterBase 5 for Solaris.
3. It is important to set the following environment variable for the proper function of the InterBase postinstall script:

```
# NONABI_SCRIPTS=TRUE
# export NONABI_SCRIPTS
```

4. Use the **pkgadd** utility to install InterBase from the CD-ROM:

```
# pkgadd –d /cdrom/interbase_SOS_V511
The following packages are available:
   1  interbase InterBase RDBMS Software
                (sparc) InterBase Version 5.5.1

Select package(s) you wish to process (or ‘all’ to process
all packages). (default: all) [?,??,q]:
```

5. Press *Enter* to choose the Install InterBase default.

```
Processing package instance <interbase> from
   </cdrom/interbase_SOS_V511>

InterBase RDBMS Software
(sparc) InterBase Version 5.5.1

Enter the absolute pathname of the install directory
   (default /usr) [?,q]
```

6. Press *Enter* to accept the default or type in the pathname in which to create the *interbase* directory.

```
Using </usr> as the package base directory.
## Processing package information.
## Processing system information.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.

This package contains scripts which will be executed with superuser
permission during the process of installing this package.

Do you want to continue with the installation of <interbase> [y,n,?]
```

7. Choose *y*. The installation script must have superuser privilege to modify system files and create symbolic links.

   ```
   Installing InterBase RDBMS Software as <interbase>
   ## Installing part 1 of 1.
   /usr/interbase/install.txt
   /usr/interbase/Release_Notes.pdf
   . . .
   [ verifying class <none> ]
   ## Executing postinstall script.
   . . .
   Please enter the license certificate ID:
   ```

8. Enter the ID from your InterBase Server software activation certificate.

   ```
   Please enter the license certificate key:
   ```

9. Enter the corresponding KEY from your InterBase Server license ID/key card.

   You can add more license keys later. See **"License registration utility for UNIX" on page 69**.

   At this step, the InterBase install script looks for the InterClient installation script on the CD-ROM, and runs it if possible. See **"Installing InterClient on UNIX" on page 64**.

   Thereafter, the install script returns to **pkgadd**:

   ```
   Installation of <interbase> was successful.
   The following packages are available:
      1  interbase InterBase RDBMS Software
                   (sparc) InterBase Version 5.5.1

   Select package(s) you wish to process (or 'all' to process
   all packages). (default: all) [?,??,q]:
   ```

10. Choose *q*. There is only one package to install using **pkgadd**.
11. If you saved customization files, restore them now with the ones you backed up before installing InterBase 5. For example:

    ```
    # cp /tmp/isc_config /usr/interbase
    ```

12. If you intend to run the **ibserver** daemon as a user other than root, create an UNIX user account named “interbase” on your machine. Log in as this user before starting the SuperServer with the **ibmgr** utility.
13. Execute the following command to start the InterBase server:

```
# echo “/usr/interbase/bin/ibmgr –start –forever” | su interbase
```

    This starts the SuperServer daemon (**ibserver**) and a guardian (**ibguard**) program that keeps track of **ibserver**.
14. Now that the server is running, you can restore the security database with the ones you backed up before installing InterBase 5.

```
# gbak –r /tmp/isc4.gbak jupiter:/usr/interbase/isc4.gdb
```

15. To test your existing databases with InterBase 5, use **gbak** to restore the backup files you created before upgrading.

Installing on HP-UX

1. Log in to your database server as *root*.
2. Load the CD-ROM with the InterBase 5 product. Mount the CD-ROM at an appropriate mount point. For example, */cdrom/interbase*.
3. **cd** to */usr*, the directory that will be the parent of the *interbase* directory, and extract the distribution file. For HP-UX, the distribution file is in tar format, and is named *InterBase511_HUS.tar*. For example:

```
# cd /usr
# tar –xvf /cdrom/interbase/Interbase511_HUS.tar
```

4. From the parent of the interbase directory (*/usr*), run the install script:

```
# ./interbase/install
```

    This script creates some symbolic links in */usr/lib* for InterBase libraries, and modifies the */etc/services* files to configure the InterBase 5 installation. It also unconfigures a V4.0 entry in */etc/inetd.conf* if one exists on this server node.
5. If you saved customization files, restore them now with the ones you backed up before installing InterBase 5. For example:

```
# cp /tmp/isc_config /usr/interbase
```

6. If you intend to run the **ibserver** daemon as a user other than root, create an UNIX user account named “interbas” on your machine. Log in as this user before starting the SuperServer with the **ibmgr** utility.

7. Execute the following command to start the InterBase server:

   ```
   # echo “/usr/interbase/bin/ibmgr –start –forever” | su interbase
   ```

   This starts the SuperServer daemon (**ibserver**) and a guardian (**ibguard**) program that keeps track of **ibserver**.

8. Now that the server is running, you can restore the security database with the one you backed up before installing InterBase 5.

   ```
   # gbak –r /tmp/isc4.gbak jupiter:/usr/interbase/isc4.gdb
   ```

9. To test your existing databases with InterBase 5, use **gbak** to restore the backup files you created before upgrading.

Running multiple versions on UNIX

You can run only one version of the InterBase server per host: you cannot run multiple versions of the InterBase server simultaneously. The InterBase 5 server process is called **ibserver**. There is no longer any **gds_inet_server** as there was in the Classic architecture.

If you want to install InterBase 5 but retain the ability to go back to a previous version, you should install InterBase5 in a location other than */usr/interbase*, and create a symbolic link from */usr/interbase* that points to the directory in which you extracted the tar file.

To switch between versions of InterBase, follow these steps:

1. Shut down the current server and point the symbolic link to a directory that contains the version you want to run.
   - Use `ibmgr -shut` to stop the 5.*x* server process
   - Use `gds_drop -a` to stop all instances of the V4.0 or V3.3 server process
2. Run the install script for the version you want to run. This makes the necessary change to the */etc/inetd.conf* configuration file.

   ```
   cd /usr ; sh ./interbase/install
   ```

3. Use `kill –HUP` to refresh the **inetd** daemon with the new configuration.

   ```
   ps | grep inetd
   kill -HUP process-ID
   ```

4. Start the server in the directory to which the symbolic link points.
   - No further action is necessary to start the V4.0 server; **inetd** starts the server process automatically.

- To start the InterBase 5 server daemon, follow the steps in **"UNIX daemon" on page 80**.

Installing InterClient on UNIX

InterClient 1.*x* installs as a tar file on both Solaris and HP-UX. The instructions below are relevant for both platforms. The examples given below are for Solaris, but the installation procedure for HP-UX has very similar output.

1. Extract the InterClient tar file. This creates a temporary directory that you can remove after installing InterClient.

```
# tar xvf /cdrom/interbase_SOS_V511
x interclient_install_temp_dir, 0 bytes, 0 tape blocks
x interclient_install_temp_dir/interclient.tar, 743936 bytes
x interclient_install_temp_dir/install.sh, 27604 bytes
```

2. Run the install script.

```
# cd interclient_install_temp_dir; sh ./install.sh

Installation script for product: InterClient
                    revision   : 1.x
                    date       : Sun Nov 9 14:10:22 PST 1997

The installation is logged in the file
   directory/interclient_install_temp_dir/install.log.

Press Enter to continue
```

3. Press *Enter*. The install script issues a warning about the port number used in InterClient prior to release 0.72. Read this, and press *Enter* again.

```
Executing : tar xpvf directory/interclient_install_temp_dir/
   interclient.tar ...

   INTERCLIENT 1.x LICENSE AGREEMENT
. . .
(EOF) Press enter to continue
```

4. The install script pages through the license agreement for InterClient. Read this carefully for information on the terms under which you can use and distribute InterClient. When you are done, press *Enter*.

```
Select products to install [Q to quit] :
   1-  Development Environment:
      InterServer and InterClient
   2- Development and Web Server Environment:
      InterServer, InterClient,
      and deployment of InterClient to Web server document root
   3- Deploy to Web Server Only:
      Deployment of InterClient to Web server document root
   4- Runtime Class Files Only:
      InterClient only

Enter choice [default = 1] : 2
```

5. Enter your choice. Refer to **"InterClient installation modes" on page 55** for a description of the different modes. This example follows choice 2, Development and Web Server Environment.

```
   You have selected the Development and Web Server Environment
      installation
   Enter the Company Name [Q to quit] : ACME Systems, Inc.

   Enter your Name [Q to quit] : John Q. Javaprogrammer

   Enter installation directory as an absolute path starting from
root.
   If the default directory /usr/interclient is not used,
   a soft link /usr/interclient will be created to point to
   your installation directory.
   Enter the root directory [/usr/interclient] :
```

6. Press *Enter* to accept the default, or enter a preferred pathname.

```
   The directory /usr/interclient does not exist.
   Select "Q" to quit, or \
          "Y" to create the directory, or
          "N" to select a new directory.
   Enter selection [Y] :
```

7. Press *Enter*.

```
   Creating root directory /usr/interclient ...
   Directory /usr/interclient created.
   Executing : zcat icserver.tar.Z | tar xpvf - ...
   Executing : zcat icdriver.tar.Z | tar xpvf - ...
   Do you want to copy CommDiag and the InterClient class files to your
   Web server's document root directory now?
   Select "Y" to copy the files, or
```

```
         “N” to skip this step.
   Enter selection [N] : n

   Copying Install.txt, Readme.txt, Relnotes.txt, and License.txt
      to /usr/interclient ...

   Copying ic_license.dat to /usr/interclient ...

   Checking /etc/services...
   interserver 3060/tcp

   If the above services entry is not identical to
   interserver 3060/tcp
   Then this services entry must be modified manually with an editor

   Checking /etc/inetd.conf...
   interserver stream tcp nowait root
/usr/interclient/bin/interserver
      interserver

   If the above inetd configuration entry is not identical to
   interserver stream tcp nowait root
/usr/interclient/bin/interserver
      interserver
   Then the entry must be modified manually and inetd restarted with
      kill -HUP
```

8. The install script displays information about how to run Java applications with InterClient:

```
   To run local InterClient Java applications you must
   append your CLASSPATH environment variable as follows
      /usr/interclient/interclient.jar:$CLASSPATH

   From the HotJava browser, test your web server installation:
      http://localhost/CommDiag.html

   From the Netscape browser, test your web server installation:
      http://localhost/CommDiagForNetscape.html

   Test your installation with the CommDiag Java application:
      java interbase.interclient.utils.CommDiag
   From the appletviewer, unset your CLASSPATH and invoke
      appletviewer /usr/interclient/CommDiag.html

   Browse the InterClient API documentation from the local URL:
      file:/usr/interclient/docs/Package-interbase.interclient.html
      file:/usr/interclient/docs/customHierarchy.html
```

```
The above installation notes and a record of the installation
can be found in the file directory/interclient_install_temp_dir/
   install.log.

InterClient Installation is complete.
```

Uninstalling InterBase on UNIX

- On Solaris, use **pkgrm** to remove the InterBase 5 package:

  ```
  # pkgrm interbase
  ```

- On HP-UX, remove the InterBase installation directory and all links to it:

  ```
  # /bin/rm –rf /usr/interbase
  # /bin/rm -i /usr/lib/*gds*
  # /bin/rm /usr/include/ibase.h
  ```

  Edit the */etc/services* file to remove the entry for *gds_db*.

Licensing

This section describes:

- The InterBase licensing mechanism
- The components and options of an InterBase license
- Tools available on Windows and UNIX for license registration

Software activation certificates

To install and use the InterBase software, you must have one or more software activation certificates (also sometimes called “license certificates”). Each certificate bears a unique ID and key pair that enables a specific functionality. During installation, you are required to enter exactly one valid certificate ID and key. Use the InterBase License Registration tool for your platform to enter additional certificate ID/key pairs—and therefore functionality—to InterBase.

The InterBase Media Kit contains the CD-ROM and documentation for your platform. You order software activation certificates as needed to acquire key to enable InterBase functionality. You can purchase additional functionality—such as a greater number of concurrent users—at any time through your sales representative.

Your initial InterBase kit includes licensing for the InterBase client at no additional cost.

License registration tools

On Windows platforms, InterBase provides a GUI tool, the InterBase License Registration Tool, for adding software activation certificates to InterBase. On UNIX, run **iblicense** at the system prompt.

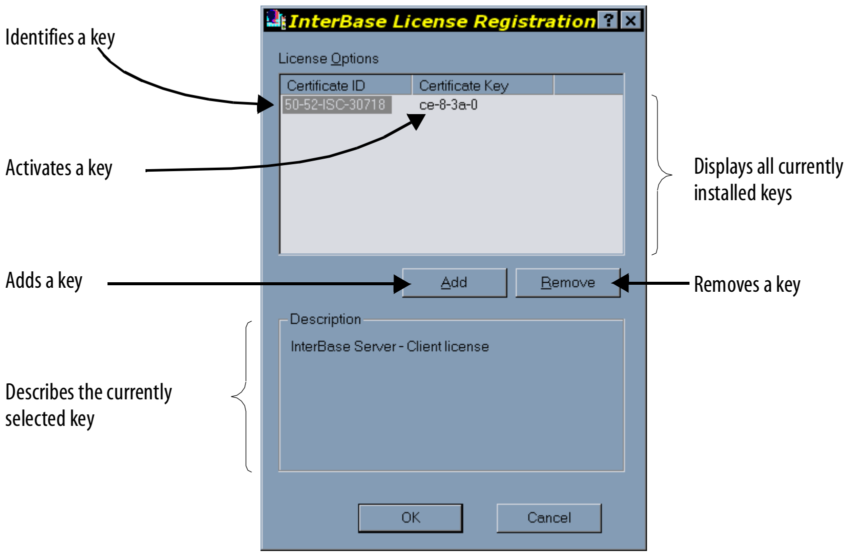
▸ *The InterBase License Registration Tool for Windows*

This GUI tool permits you to add, remove, and view certificate ID and key numbers on Windows platforms.

1. If you are adding certificate ID/keys, have these numbers handy.
2. Display the InterBase License Registration Tool by choosing **Programs | InterBase 5 | InterBase License Registration Tool** from the Start Menu.

   Note You must run this application on the machine where the license file resides. You can also invoke it by running <*ib_install_dir*>\bin\iblicense.exe.

FIGURE 3.1 License registration tool dialog



Certificate ID	Certificate Key
50-52-ISC-30718	ce-8-3a-0

Description: InterBase Server - Client license

3. Perform one of the following three actions:

To add a new ID/key Click Add to display the Add License Certificate dialog. Type the ID and key numbers in the appropriate fields. Click OK.

To remove an ID/key Highlight the Certificate ID that you want to delete and click the Remove button. When you remove a key, InterBase saves the current state of the license file in a backup file, named *ib_bckup.dat*

To view information about an ID/key Highlight the Certificate ID in the list; the License Tool displays information about it in the Description area.

4. To add, delete, or view another ID/key, repeat Step 4.
5. When you have completed your changes, click OK.
6. If you add or delete ID/keys, you must restart the server process for the changes to take effect.

▸ *License registration utility for UNIX*

On UNIX platforms, run the **iblicense** utility to add certificate ID/keys. If you are adding certificate ID/keys, have the numbers handy.

iblicense replaces the **isc_install** script that was used in InterBase V4.0 to add license keys to *ib_license.dat* on UNIX servers.

You must run **iblicense** on the machine where the license file resides.

Command-line syntax

To add only one ID/key, use the following syntax:

```
iblicense -command [-option parameter ...]
```

For example:

```
iblicense -add -id IDNUMBER -key KEYSTRING
```

To add several ID/keys in one session, invoke **iblicense** with no arguments.

```
iblicense Enter
IB_LICEN> command [-option parameter …]
IB_LICEN> command [-option parameter …]
…
IB_LICEN> Quit
```

IMPORTANT After an add or remove command succeeds, you must restart the server to ensure that the new license is being used.

Removing keys When you remove a key, InterBase saves the current state of the license file in a backup file, named *ib_bckup.dat*.

Options Commands are:

Command/option	Description
`add -key` *`keynumber`* `-id` *`idnumber`*	Add a new license key; key is the license key value, -id is the
`display`	Display all current keys
`remove -key` *`key`*	Remove a key
`help`	Prints this list
`quit`	Exit prompt mode
`z`	Display version

TABLE 3.1 **iblicense** commands and their options

Arguments to the **add** and **remove** commands listed above are:

Option	Description
`key`	License key value
`id`	License id value

TABLE 3.2 **iblicense** options

You can abbreviate a command or option by issuing the initial letters until you have uniquely identified it.

Once the command has been issued, the utility reports whether or not the operation succeeded.

Examples The following are examples of using the **iblicense** utility.

- Adding a key:

```
iblicense –add –key 00-b-90-21 –id 60-60-84200
```

- Removing a key:

```
iblicense –r –k 00-b-90-21
```

- Displaying keys:

```
iblicense –display
```

- Displaying help:

```
iblicense -h
```

Note The example key and ID values above are fictional and do not enable a server.

Available certificates

Each software activation certificate enables a specific piece of functionality, which is identified in the box that contains the Certificate ID and key numbers.

Certificate	Functionality
Server Activation	Activates local access to the server for one (1) user
Metadata	Allows database metadata manipulation for the server; that is, SQL statements CREATE, ALTER, and DROP
Remote Access	Enables the server to accept requests for database access from remote clients
Client Capability	Allows the software to act as a client and to connect to other InterBase servers
Local interBase	Activates local access to the server for one user, with metadata and client capabilities
Pre-Compilers	Allows for GPRE using various languages
Simultaneous Users	Regulates the number of clients that can connect to this server at the same time; clients can be local or remote
Internet Access	Allows use of the InterBase server with Web servers

TABLE 3.3 Certificate keys available for InterBase 5

A note on simultaneous connections

The USERS component of an InterBase license string indicates the maximum number of clients that can simultaneously connect to databases on the server host. The number following the word USERS determines the number of distinct users that can connect at one time. Each user can have up to four connections. See **"The attachment governor" on page 91**.

The lines in the license file must be unique. InterBase ignores duplicate instances of a certificate ID.

TIP In Borland Delphi and other VCL-based Borland tools, each **TTable** and **TQuery** component establishes a separate connection to the database by default, and therefore counts toward the connection limit. You should instead use a **TDatabase** component to connect to the database and associate each **TTable** and **TQuery** with the **TDatabase**. This way, the application uses only one database connection per **TDatabase**.

The InterBase license file

Information about what functionality has been enabled is stored in the *ib_license.dat* file in the InterBase install directory on both Windows and UNIX. The InterBase License Registration Tool (Windows) and **iblicense** utility (UNIX) both write to this file. Users do not have to write to this file directly. This section describes the contents of this file.

ELEMENTS IN THE IBLICENSE.DAT FILE

Each line of *iblicense.dat* contains four or more components, listed in the table below:

Component	Description
COMMENT *string*	Descriptive comments
ID *string*	Unique identification for a key
KEY *string*	Encoded key you need to activate a server capability
OPTIONS *string*	Set of single-character codes for enabling specific InterBase server or client functionality; see below
PRODUCT *name*	"InterBase" or "InterClient"
UNTIL *date*	Expiration date of the license
USERS *n*	User limit for the attachment governor; see below
VERSION *string*	Code for InterBase platform; for example, "WI" for Windows NT and Windows 95, "SO" for Solaris, and "HU" for HP-UX

TABLE 3.4 InterBase components

The KEY component is a form of password, encoded from the combination of all the other components present in the license key string. You need the KEY in order to activate the capabilities specified in OPTIONS. Any change to the components, such as adding to the OPTIONS or altering the number of USERS, invalidates the KEY because the KEY is based on the specific values of the components.

OPTIONS IN THE IBLENCESE.DAT FILE

The capabilities of an InterBase installation are determined by the options present in the license file. Each license line in the file contains some number of single-letter option codes in the OPTIONS component of the line. The options and the respective functions they enable are listed below:

Option	Enables functionality
A	**gpre** preprocessor for Ada compilers
C	**gpre** preprocessor for C compilers
D	Metadata changes with CREATE, ALTER or DROP statements
E	External table access
F	**gpre** preprocessor for Fortran compilers
I	Internal table access
J	Listening server capability (like the "S" option) for the InterServer component of InterClient; appears in InterServer's license file, not InterBase's
L	**gpre** preprocessor for Fortran compilers
Q	Query tool (**isql**)
R	Client capability; required for clients
S	Server capability; required for servers
W	Internet access license with unlimited users
2	**gpre** preprocessor for COBOL compilers
3	**gpre** preprocessor for C++ compilers

TABLE 3.5 InterBase license options

CHAPTER

4

Server Configuration

This chapter describes operation and configuration of the InterBase server process. The chapter includes topics: starting and stopping the server process on each platform, environment configuration, managing temporary files, monitoring client connections, and reading log files.

Windows NT service

On Windows NT, a service is a program that runs outside of the context of a given user login session. A service runs even when there is no user logged in on the desktop of the Windows NT system. This section details how to configure and run InterBase as a service on Windows NT.

Starting and stopping the service

The Local InterBase Server and InterBase Server must be started prior to enabling database connections.

Selecting service startup options

You can configure InterBase Server services or change its startup options by using the Registry Configuration Utility (**regcfg.exe**). This utility enables you to:

- Choose how InterBase starts:
  - Start InterBase automatically as a service (recommended). InterBase starts automatically when you start Windows NT.
  - Start InterBase manually as a Service. InterBase must be started using the Services Control Panel or the Registry Configuration Utility (*regcfg.exe*).
  - Start InterBase automatically as an icon. The enabled server appears as an icon on the desktop each time you start Windows NT.
  - Start InterBase manually as an icon. To start the server, run InterBase Server from the InterBase 5 Start menu. The server appears as an icon on the desktop.
- Choose the process priority. The InterBase server process is a background process that by default runs at normal priority. If there are other processes running at high priority, you may wish to increase the priority of the InterBase process. You can run InterBase in either normal or high-priority.
- Determine the location of the InterBase Server or change the directory for the InterBase license file, message file, and security database.
- Remove InterBase Information from the Registry and Services database.
- Start and Stop NT service. Used to start up and stop the InterBase Server.

If the status of the InterBase Server Service is started, then stop it before changing the status. Once the service has been stopped, you can change how InterBase starts and then restart the NT Service Control.

If you want to change the process priority, you must first stop the InterBase Server Service using the NT Service Control, then choose a different service priority and the InterBase Server Service. Thereafter, the InterBase server process starts with higher priority.

If InterBase is not getting the CPU share that it needs, and for some reason you do not want to boost the priority of the InterBase process, you can change the default priority of foreground and background processes. Invoke the System application in the **Control Panel**.

In the System dialog that opens, choose the **Tasking** button, then select the option "Foreground and Background Applications Equally Responsive." This ensures that the InterBase process (that runs in the background) receives the same priority as foreground processes. Note that all other background processes have their priority boosted as well.

TIP Disable any CPU-intensive screen savers on your server, or set it to “blank screen.” A screen saver can use an inordinate amount of CPU cycles, and Windows NT does not provide a way to reduce the priority of screen savers or to favor services. Screen savers are not generally necessary to prevent phosphor burn on modern monitors.

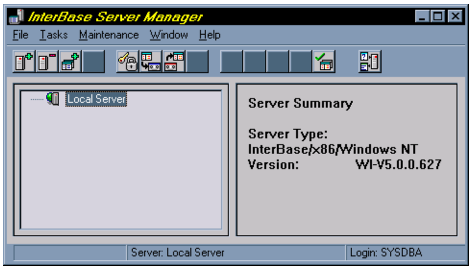
For more information on foreground and background tasking, see the Microsoft Windows NT documentation.

Windows NT server properties

For servers running on Windows NT platforms, Server Manager provides server information and an interface for configuring server-wide cache page size defaults, client map size, and InterBase process RAM.

Server information

For information on server properties such as version number and capabilities, display the InterBase Server Manager and connect to a server. Highlight the server icon in the left panel of the Server Manager to display its properties in the right panel.



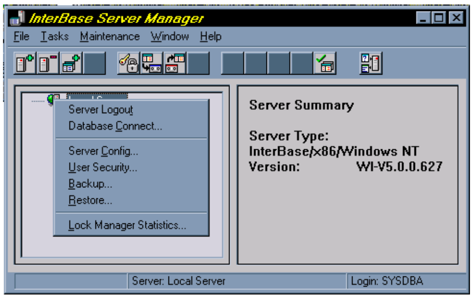
Server Summary

Server Type: InterBase/x86/Windows NT
Version: WI-V5.0.0.627

Server: Local Server
Login: SYSDBA

Server configuration

To configure server-wide cache page size, client map size, and working set properties on a Windows NT server, follow these steps:



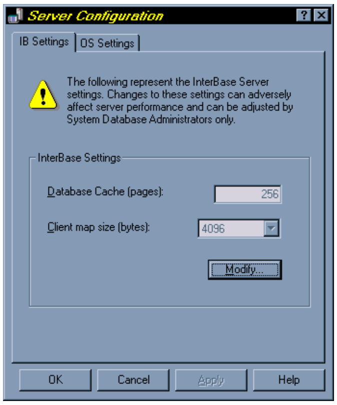
Server Summary

Server Type: InterBase/x86/Windows NT
Version: WI-V5.0.0.627

Server: Local Server
Login: SYSDBA

1. Run the InterBase Server Manager by choosing it from the InterBase folder on the Start Menu.
2. Connect to the server that you want to configure.
3. Right-click the server icon in the left panel of the Server Manager dialog, and choose Server Config from the mouse menu to display the Server Configuration dialog.

SERVER-WIDE CACHE PAGE SIZE


Server Configuration

IB Settings | OS Settings

The following represent the InterBase Server settings. Changes to these settings can adversely affect server performance and can be adjusted by System Database Administrators only.

InterBase Settings

Database Cache (pages): 256

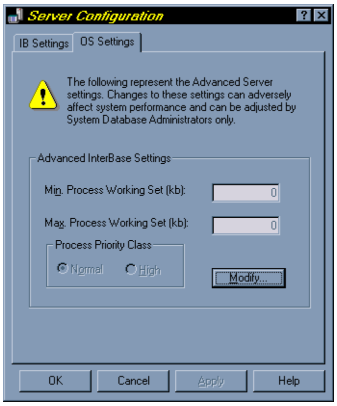
Client map size (bytes): 4096

The Database Cache (Pages) field permits you to change the server-wide cache page size default. This is not a recommended practice, since it is likely to result in inappropriate cache page sizes for some databases. The **gfix** utility changes cache page size for individual databases, and **isql** connections can specify a cache page size at connection time. See **Configuring the SuperServer cache on page 140** for a discussion of setting cache page sizes.

CLIENT MAP SIZE

Interprocess communication on the server uses a memory-mapped file. Each client attachment gets a segment of the file equal to the client map size in bytes. The file is initially ten times the client map size. Additional clients attaching and detaching cause the file to grow and shrink by 1 increment of the client map size. You can tune this with the SERVER_CLIENT_MAPPING keyword in *isc_config*. This is part of the IB Settings of the Properties dialog of the InterBase 5 service. By default, it is 4096 bytes. This configuration takes effect when the InterBase service restarts.

WORKING SET


Server Configuration

IB Settings | OS Settings

The following represent the Advanced Server settings. Changes to these settings can adversely affect system performance and can be adjusted by System Database Administrators only.

Advanced InterBase Settings

Min. Process Working Set (kb): 0

Max. Process Working Set (kb): 0

Process Priority Class: [✓ Normal] High

This refers to the set of RAM dedicated for the InterBase process. The minimum working set size specifies the amount of physical RAM that the operating system guarantees for the InterBase process. The system may swap out memory in excess of this figure. The maximum working set size specifies a threshold above which the operating system swaps out memory used by the InterBase process.

Using the Windows NT Performance Monitor, a database administrator can watch for excessive Page Faults. Raise the minimum working set size in this case. If memory resources permit, set the minimum working set size to at least the amount of cache allocated for the InterBase server. Certainly the maximum should be greater than the size of the InterBase cache, so NT does not force the cache to swap.

The default values are **zero** for both minimum and maximum. This is a special case in which the system determines the working set for the InterBase server process. Both values must be less than the amount of physical RAM on the machine. The minimum must be less than the maximum. These parameters are specified by the keywords SERVER_WORKING_SIZE_MIN and SERVER_WORKING_SIZE_MAX in *isc_config*. This is part of the OS Settings of the Properties dialog of the InterBase 5 service. This configuration takes effect when the InterBase service is restarted.

Shutting down the server on Windows NT

To shut down the InterBase Server running as an icon, left-click the server icon to display a popup menu and select Shutdown Server from the menu. A message appears indicating the number of open database connections. If the message indicates zero (0) active connections, click OK to shut down the server.

To shut down the InterBase Server running as an NT Service, use the Stop NT Service Control in the Registry Configuration Utility or the NT Service Control.

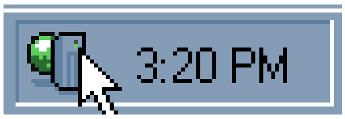
If you have open connections, it is recommended that you close them before shutting down the server. You also must close all client application you are running.

Windows 95 peer-to-peer server application

You must start Local InterBase or InterBase Server prior to enabling database connections.

When you install the Local InterBase Server or the InterBase Server and reboot your system, the enabled server appears as an icon on the Task Tray, located on the right side of the Task Bar.

FIGURE 4.1 InterBase Server icon on the task tray



To start a server that has been shut down, run **InterBase Server** from the InterBase 5 Start menu. The enabled server icon appears on the Task Tray.

Selecting startup options

You can configure how InterBase Server starts by using the Registry Configuration Utility (**regcfg.exe**). This utility enables you to:

- Start a server each time you start Windows 95. The enabled server appears as an icon on the Task Tray.

  Start a server manually by running **InterBase Server** from the Interbase 5 Start menu. The enabled server then appears as an icon on the Task Tray.

Shutting down InterBase on Windows 95

To shut down the Local InterBase or InterBase server, right-click the InterBase server icon located in the Task Tray to display a popup menu and select Shutdown. A message appears indicating the number of open database connections. If the message indicates zero (0) active connections, click OK to shut down the server.

If you have open connections, it is recommended that you close them before shutting down the server. You also must close all client applications you are running.

UNIX daemon

Syntax

```
ibmgr -command [-option [parameter] ...]
```

or

```
ibmgr Enter
IBMGR> command [–option [parameter]]
```

Description On UNIX, the InterBase server process runs as a daemon. A daemon runs even when no user is logged in to the console or a terminal on the UNIX system.

ibmgr is a utility for managing the InterBase server process on UNIX systems. You must be logged in into the machine on which the server is running to use **ibmgr**.

Note The *ibmgr32.exe* file that is present in Windows installations is an older client-side utility whose functions are entirely different than **ibmgr** on UNIX. The name is coincidental.

Options

`start [–once | –forever]`	Starts server; the *–forever* switch causes the server to restart if it crashes; default is *–forever*
`shut`	Rolls back current transactions, terminates client connections, and shuts down server immediately
`show`	Shows host and user
`user` *`user_name`*	Supply SYSDBA
`passwd` *`password`*	Supply SYSDBA password
`help`	Prints help text
`quit`	Quits prompt mode

TABLE 4.1 **ibmgr** *commands*

The command switches *–user* and *–passwd* can also be used as option switches for commands like *-start* or *-shut*. For example, you can shut down a server in any of the following ways:

```
ibmgr –shut –passwd password
```

or

```
ibmgr Enter
IBMGR> shut –passwd password
```

or

```
imbgr Enter
IBMGR> passwd password
IBMGR> shut
```

Starting the server

To start the InterBase server, log in as the “root” or “interbase” user. (“interbas” is a synonym for “interbase,” to accommodate operating systems that do not support nine-character account names.) Execute the following command:

```
ibmgr –start
```

Note Once you have started **ibserver** using one login, such as "root," be aware that all objects created belong to that login. They are not accessible to you if you later start **ibserver** as one of the other two ("interbas" or "interbase"). It is highly recommended to run the InterBase Server as "interbase."

Stopping the server

To stop the InterBase server, execute the following command:

```
ibmgr -shut -password SYSDBA_password
```

You do not need to log on as "root" to do this.

Note Currently, the *–shut* command rolls back all current transactions and shuts down the server immediately. If you need to allow clients a grace period to complete work and detach gracefully, use shutdown methods for individual databases. See **"Database shutdown and restart" on page 146**.

Starting the server automatically

To configure a UNIX server to start the InterBase 5 Server automatically at boot-time of the server host, you need to install a script that the *rc* initialization scripts can run. Refer to */etc/init.d/README* for more details on how UNIX runs scripts at boot-time.

Example initialization script

```
#!/bin/sh
# ibserver.sh script - Start/stop the InterBase 5 daemon

# Set these environment variables if and only if they are not set.
: ${INTERBASE:=/usr/interbase}
: ${ISC_USER:=SYSDBA}
: ${ISC_PASSWORD:=masterkey}
# WARNING: in a real-world installation, you should not put the
# SYSDBA password in a publicly-readable file. To protect it:
# chmod 700 ibserver.sh; chown root ibserver.sh
export INTERBASE
export ISC_USER
export ISC_PASSWORD
```

```
ibserver_start() {
    # This example assumes the InterBase server is
    # being started as UNIX user ’interbase’.
    echo '$INTERBASE/bin/ibmgr -start -forever' | su interbase
}

ibserver_stop() {
    # No need to su, since $ISC_USER and $ISC_PASSWORD validate us.
    $INTERBASE/bin/ibmgr -stop
}

case $1 in
’start’) ibserver_start ;;
’start_msg’) echo 'InterBase Server starting...\c' ;;

’stop’) ibserver_stop ;;
’stop_msg’) echo 'InterBase Server stopping...\c' ;;

*) echo 'Usage: $0 { start | stop }' ; exit 1 ;;
esac

exit 0
```

Example initialization script installation on Solaris

1. Log in as root.

```
$ su
```

2. Enter the example script above into the initialization script directory.

```
# vi /etc/init.d/ibserver.sh
```

3. Enter text
4. Link the initialization script into the *rc* directories for the appropriate run levels for starting and stopping the InterBase server.

```
# ln /etc/init.d/ibserver.sh /etc/rc0.d/K30ibserver
# ln /etc/init.d/ibserver.sh /etc/rc2.d/S85ibserver
```

Example initialization script installation on HP-UX

1. Log in as root.

```
$ su
```

2. Enter the example script above into the initialization script directory.

```
# vi /sbin/init.d/ibserver.sh
<Enter text>
```

3. Link the initialization script into the *rc* directories for the appropriate run levels for starting and stopping the InterBase server.

```
# ln -s /sbin/init.d/ibserver.sh /sbin/rc1.d/K500ibserver
# ln -s /sbin/init.d/ibserver.sh /sbin/rc2.d/S500ibserver
```

InterBase Guardian process

The InterBase Guardian manages the InterBase server under both Windows NT and Windows 95. By default, Guardian runs as a service under Windows NT and as an application under Windows 95.

Users normally do not need to interact with Guardian in any way; it operates as an invisible process. When Guardian is configured for “Windows Startup” and “Start Always”—the default settings—it starts the InterBase server whenever Windows starts, monitors the server, and restarts it if it stops due to anything other than a normal shutdown by a user.

A number of options are available for starting and configuring Guardian and the InterBase server. See **“Starting Guardian” on page 85**, **“Starting the server without Guardian” on page 85**, and **“Configuring Guardian and the server” on page 86**.

The following options are available under Windows NT:

- Guardian can be started automatically as a service (default behavior)
- Guardian can be started manually as a service
- Guardian can be disabled, and the server can run as either a service or an application, started automatically or manually

The following options are available under Windows 95:

- Guardian can be started automatically as an application (default behavior)
- Guardian can be started manually rather than starting automatically with Windows
- Guardian can be disabled and the server can be run independently as an application, started either automatically or manually

On Windows 95, you can access the InterBase Server properties by right-clicking the Guardian icon in the Task Tray.

The following options are available under UNIX:

- Guardian can be started as a daemon automatically with **ibmgr -start -forever** (this is also the default behavior if you do not specify *-forever* or *-once*)
- Guardian is not started if you run **ibmgr -start -once**

Starting Guardian

To start Guardian manually, Startup Mode must be set to either Windows Startup or Manual Startup (see **page 89**). Startup always fails if either Guardian or the server is already running.

- To start Guardian manually as an application, run **InterBase Guardian** from the Interbase 5 Start menu. Administrative privileges are not required to start Guardian.
- Any attempt to either start the server while the Guardian is running or to start the guardian twice results in an error message.
- To start Guardian manually as a service under Windows NT, go to the **Control Panel | Services**, highlight InterBase Guardian, and click Start.

Starting the server without Guardian

This section describes how to run the server directly, without using Guardian.

Running the server as an application

To run the InterBase Server without running Guardian, run **InterBase Server** from the Interbase 5 Start menu. Manual startup detects whether the server is already running and will not start up if that is the case.

Running the server as a service

On Windows NT, you can choose to run the InterBase server as a service rather than as an application. To do this, you must disable the Guardian.

1. Shut down the Guardian, or shut down SuperServer if it is running without the Guardian.
2. Run **InterBase Configuration** from the Interbase 5 Start menu. and in the Guardian page, set the Startup Mode to Disabled.

3. Move to the General page of the dialog and set Server Startup to Service. Set the Startup Mode to Windows Startup if you want SuperServer to start automatically when Windows starts, or to Manual if you want to start it manually.
4. Click OK.
5. Go to **Control Panel | Services**, highlight InterBase Server, and click Start.

To shut down the server when it is running as a service, go to **Control Panel | Services**, highlight InterBase Server, and click Stop.

Configuring Guardian and the server

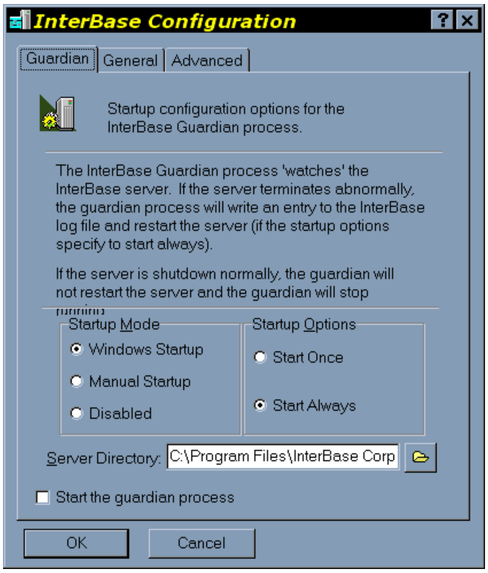
The Startup Configuration dialog provides a number of ways to configure the behavior of both the InterBase Guardian and the InterBase server. Under Windows NT, only a user with administrative privileges can access the InterBase Configuration dialog. It is available to all users under Windows 95.

There are two ways to invoke the InterBase Configuration dialog:

- Run **InterBase Configuration** from the Interbase 5 Start menu
- If the icon for either the InterBase server or Guardian is present in the Task Bar, you can right-click the icon and choose Startup Configuration from the mouse menu

Either method displays the InterBase Configuration dialog:

FIGURE 4.2 Interbase Guardian configuration dialog



InterBase Configuration

Guardian | General | Advanced

Startup configuration options for the InterBase Guardian process.

The InterBase Guardian process 'watches' the InterBase server. If the server terminates abnormally, the guardian process will write an entry to the InterBase log file and restart the server (if the startup options specify to start always).

If the server is shutdown normally, the guardian will not restart the server and the guardian will stop running.

Startup Mode: [✓ Windows Startup] Manual Startup Disabled

Startup Options: Start Once [✓ Start Always]

Server Directory: C:\Program Files\InterBase Corp

Start the guardian process: ☐

The General page configures the startup behavior of the new InterBase Guardian.

Startup Mode

Windows Startup Whenever Windows starts, Guardian starts and invokes the InterBase server.

Manual Startup No automatic startup of Guardian or the InterBase server.

- *To start Guardian as a service*, run **Control Panel | Services**, highlight InterBase Guardian, and click Start
- *To start Guardian as an application*, run **InterBase Guardian** from the Interbase 5 Start menu

Disabled The Guardian cannot be started. To re-enable the Guardian, choose Windows start or Manual start.

Startup Options

Start Once If the InterBase server exits unexpectedly, Guardian does not restart it.

Start Always Guardian monitors the InterBase server and restarts it if it exits unexpectedly.

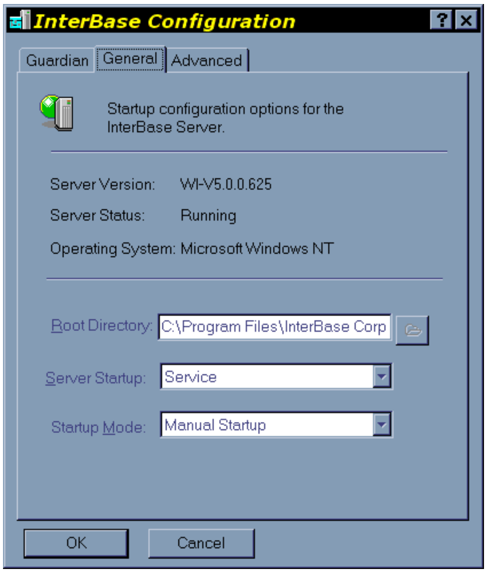
Server Directory

This should point to the directory that contains the InterBase server executable (*ibserver.exe*). Unless you have put the server executable in a different directory than the Guardian executable after installation, you do not need to change this. Guardian assumes that *ibguard.exe* and *ibserver.exe* are in the same directory.

Start the guardian process or service

If Guardian is not already running, you can start it immediately by checking this and clicking OK. If you do this when the InterBase server is already running, you receive an error message.

FIGURE 4.3 Interbase general configuration dialog



Startup configuration options for the InterBase Server.

Server Version: WI-V5.0.0.625

Server Status: Running

Operating System: Microsoft Windows NT

Root Directory: C:\Program Files\InterBase Corp

Server Startup: Service

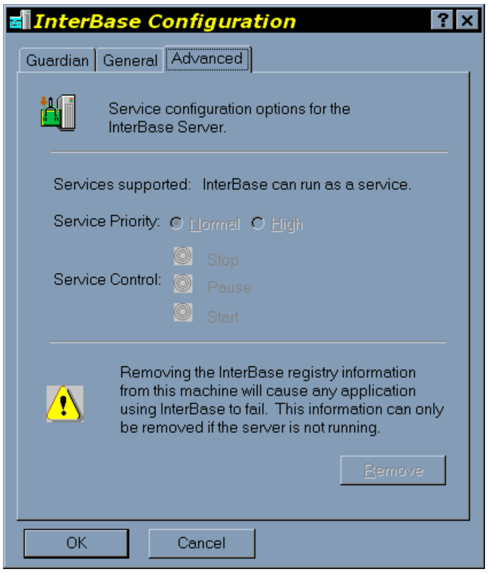
Startup Mode: Manual Startup

The General page of the Startup Configuration dialog configures the startup behavior of the InterBase server, not of the Guardian.

Option	Description
Root Directory	If you have more than one version of InterBase installed, you can choose which one to run by pointing the Root Directory field to the desired install directory. This is the location of InterBase server files, including *ib_license.dat*, *interbase.msg*, and *isc4.gdb*. This option is grayed out if **ibserver** is running.
Server Startup	This choice is available only when Guardian is disabled. (See **"Startup Mode" on page 89** for how to disable Guardian.) Under Windows NT, you can choose to run the server either as an application or as a service. For more information, see **"Running the server as a service" on page 85**. **ibserver** always runs as an application on Windows 95, since services are not available on that system. This option is grayed out unless Guardian is disabled.
Startup Mode	This choice is available only when Guardian is disabled. (See **"Startup Mode" on page 89** for how to disable Guardian.) This field determines whether the InterBase server starts automatically or whether it must be started manually. This option is grayed out unless Guardian is disabled. -*To start the server as an application*, run InterBase Server from the Interbase 5 Start menu -*To start the server as a service*, see **"Running the server as a service" on page 85**.

TABLE 4.2 Options on the General page of the Startup Configuration dialog

FIGURE 4.4 Interbase advanced configuration dialog



Service configuration options for the InterBase Server.

Services supported: InterBase can run as a service.

Service Priority: Normal High

Service Control:

Removing the InterBase registry information from this machine will cause any application using InterBase to fail. This information can only be removed if the server is not running.

The settings on the Advanced page are available only when Guardian is disabled and the InterBase server is running as a service under Windows NT. See page 85 for how to run the InterBase server as a service. These settings can also be made from the Services Control Panel.

Option	**Description**
Service Priority	Sets the startup priority for the InterBase server relative to other processes.
Service Control	Use these settings to start, stop, pause, and continue the InterBase server when it is running as a service under Windows NT.
Remove	If you are running more than one version of InterBase, you may wish to clear the Registry information when switching between versions. This option is enabled only if **ibserver** is not running and Guardian is disabled.

TABLE 4.3 Options on the Advanced page of the Startup Configuration dialog

The attachment governor

The InterBase server has an attachment governor that regulates the number of attachments to the server. Multiply the value of the USERS field in the license file by four (4) to determine the total number of permitted concurrent attachments.

All successful attempts to create or connect to a database increment the number of current attachments. Both local and remote connections count toward the connection limit. Connections are permitted by the governor until the maximum number of concurrent attachments is reached. All successful attempts to drop or disconnect from a database decrement the number of current attachments.

Once the maximum number of attachments is reached, the server returns the error constant *isc_max_att_exceeded* (defined in *ibase.h*), which corresponds to the message:

```
Maximum user count exceeded. Contact your database administrator.
```

Environment variables

This section describes the usage of environment variables that InterBase recognizes.

ISC_USER and ISC_PASSWORD

In lieu of providing a user and password for validation to connect to a database, you can use the environment variables ISC_USER and ISC_PASSWORD, respectively. The InterBase client library reads these environment variables only if you omit the user and password in a database connect string.

The INTERBASE environment variable

INTERBASE

The INTERBASE variable is used both during installation and during runtime. During installation, it defines the path where the InterBase product is installed. If this path is different from */usr/interbase*, all users must have the correct path set at runtime. During runtime, use the INTERBASE variable to set the InterBase home directory.

INTERBASE_TMP

The INTERBASE_TMP variable can be used to set the location of InterBase's sort files on the server. There are other options for defining the location of these files. See **"Configuring sort files" on page 93**.

INTERBASE_LOCK AND INTERBASE_MSG

INTERBASE_LOCK sets the location of the InterBase lock file and INTERBASE_MSG sets the location of the InterBase message file. These two variables are independent of each other and can be set to different locations.

IMPORTANT The environment variables must be in the scope of the **ibserver** process. On Windows NT, define the variables as *System Variables* in the NT **Control Panel | System | Environment** dialog. On UNIX, the easiest way to do this is to add the variable definition to the system-wide default shell profile.

The TMP environment variable

The TMP environment variable defines where InterBase stores temporary files, if the INTERBASE_TMP variable is not defined.

Temporary file management

InterBase 5 creates two types of temporary files: sort files and history list files.

The InterBase server creates sort files when the size of the internal sort buffer isn't big enough to perform the sort. Each request (for example, CONNECT or CREATE DATABASE) gets and shares the same list of temporary file directories. Each request creates its own temporary files (each has its own I/O file handle). Sort files are released when sort is finished or the request is released. If space runs out in a particular directory, InterBase creates a new temporary file in the next directory from the directory list. If there are no more entries in the directory list, it prints an error message and stops processing the current request.

The InterBase clients **isql** and Windows ISQL create the history list files to keep track of the input commands. Each instance creates its own temporary files, which can increase in size until they run out of disk space. Temporary file management is not synchronized between clients. When a client quits, it releases its temporary files.

Configuring history files

To set the location for history files, define the TMP environment variable on your client machine. This is the *only* way to define the location of history files. If you do not set the location for the history files by defining the TMP environment variable, an InterBase client uses whatever temporary directory it finds defined for the local system. If no temporary directory is defined, it uses */tmp* on a UNIX system or *C:\temp* on a Windows system.

Configuring sort files

There are two ways to configure the sort files, which are located on the server:

- You can add an entry to the $INTERBASE*/isc_config* (UNIX) or *ibconfig* (Windows) file to enable directory and space definition for sort files. The syntax is:

  ```
  TMP_DIRECTORY size pathname
  ```

  This defines the maximum size in bytes of each sort directory. You can list several directories, each on its own line with its own size specification and can specify a directory more than once with different size configurations. InterBase exhausts the space in each specification before proceeding to the next one.

  For example, if you specify *dir1* with a size of 5MB, then specify *dir2* with 10MB, followed by *dir1* with 2 MB, InterBase uses *dir1* until it reaches the 5MB limit, then uses *dir2* until it has filled the 10MB allocated there, and then returns to *dir1* where it has another 2 MB available.

- You can use the INTERBASE_TMP and TMP environment variables to define the location.

  If temporary files are configured in *isc_config* (UNIX) or *ibconfig* (Windows), the server uses those values for the sort files and ignores the server environment variable values. If configuration of temporary files has not been defined in *isc_config* or *ibconfig*, then the server picks a location for a sort file based on the following algorithm:

  1. Use the directory defined in INTERBASE_TMP environment variable
  2. If INTERBASE_TMP isn't defined, use directory defined in TMP environment variable
  3. If TMP isn't defined, use the */tmp* directory (UNIX) or *C:\temp* (Windows)

Configuration parameters in isc_config

The *isc_config* file (*ibconfig* on Windows 95 and Windows NT) is a text file with configuration information for the InterBase server. Entries are in the form:

```
PARAMETER <whitespace> VALUE
```

Where `PARAMETER` is a string containing no whitespace, naming a property of the server to which the line specifies a configuration. `VALUE` is a number or string which is the specific configuration of the respective property.

Note In InterBase 5, each line in *isc_config* is limited to 80 characters, including the `PARAMETER` and any whitespace.

The following is a summary of the legal entries in *isc_config* (UNIX) or *ibconfig* (Windows) file:

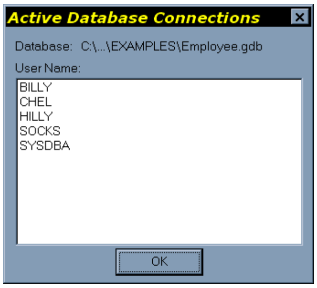
Parameter	Description
CONNECTION_TIMEOUT	Seconds to wait before concluding an attempt to connect has failed. Default: 180.
DATABASE_CACHE_PAGES	Server-wide default for the number of database pages to allocate in memory per database. This can be overridden by clients. See **"Configuring the SuperServer cache" on page 140** for more information on database cache configuration. Default: 256.
DEADLOCK_TIMEOUT	Seconds before an ungranted lock causes a scan to check for deadlocks. Default: 10.
DUMMY_PACKET_INTERVAL	Seconds to wait on a silent client connection before the server sends dummy packets to request acknowledgment. Default: 60.
LOCK_ACQUIRE_SPINS	Number of spins during a busy wait on the lock table mutex. Only relevant on SMP machines. Default: 0.
LOCK_HASH_SLOTS	Tune lock hash list. More hash slots means shorter hash chains. Not necessary except under very high load. Prime number values are recommended. Default: 101.
SERVER_CLIENT_MAPPING	Size in bytes of one client's portion of the memory mapped file used for interprocess communication. Default: 4096.
SERVER_PRIORITY_CLASS	Priority of the InterBase service on Windows NT. The value 1 is low priority, 2 is high priority. Relevant on Windows NT only. Default: 1.

SERVER_WORKING_SIZE_MAX	Threshold above which Windows NT is requested to swap out all memory. Relevant on Windows NT only. Default: 0 (system-determined).
SERVER_WORKING_SIZE_MIN	Threshold below which Windows NT is requested to swap out no memory. Relevant on Windows NT only. Default: 0 (system-determined).
TMP_DIRECTORY	Directory to use for storing temporary files. Specify number of bytes available in the directory, and the path of the directory. You may list multiple entries, one per line. Each directory is used according to the order specified. Default: Value of the INTERBASE_TMP environment variable, otherwise */tmp* on UNIX or C:\temp on Windows NT.
V4_EVENT_MEMSIZE	Bytes of shared memory allocated for event manager. Default: 32768.
V4_LOCK_GRANT_ORDER	1 means locks are granted first come, first served. 0 means emulate InterBase V3.3 behavior, where locks are granted as soon as they are available, which can result in lock request starvation. Default: 1.
V4_LOCK_MEM_SIZE	Bytes of shared memory allocated for lock manager. Default: 98304.
V4_LOCK_SEM_COUNT	Number of semaphores for interprocess communication. Classic architecture only.
V4_LOCK_SIGNAL	UNIX signal to use for interprocess communication. Classic architecture only.
V4_SOLARIS_STALL_VALUE	Number of seconds a server process waits before retrying for the lock table mutex. Relevant on Solaris only. Default: 60.

Monitoring client connections with Server Manager for Windows

To view active database connections, invoke the Server Manager for Windows, and choose **Maintenance | Database Connections** from the Main Menu, toolbar, or popup menu. The Active Database Connections dialog appears.

FIGURE 4.5 Active database connections dialog



Database: C:\...\EXAMPLES\Employee.gdb
User Name:
BILLY
CHEL
HILLY
SOCKS
SYSDBA

The dialog shows the name of the current database and active users connected to that database.

Diagnostic Log files

- InterBase Server logs diagnostic messages in the file *interbase.log* in the InterBase install directory. Any messages generated by **ibserver** are sent to this file. This can be a very important source of diagnostic information if your server is having configuration problems. Refer to the *Language Reference* for a list of error messages that can appear in this file.
- On Windows NT, the Event Viewer application contains many warnings and notifications pertaining to operating system problems, including memory, I/O, and networking failures. Some of these operating system problems can affect the InterBase server.

CHAPTER

5

Network Configuration

This chapter details issues with configuring InterBase in a networked client/server environment. Topics include network protocols supported by InterBase, remote connection specifiers, and network troubleshooting tips.

Network protocols

InterBase supports TCP/IP for all combinations of client and server platforms. Additionally, InterBase supports NetBEUI for NT servers and Windows clients, and a *local* connection mode (involving interprocess communication but no network interface) for Windows 95 and Windows NT clients.

InterBase is designed to allow clients running one operating system to access an InterBase server that is running on a different platform and operating system than the client. For example, a common arrangement is to have several inexpensive Windows 95 PCs acting as client workstations concurrently accessing a departmental server running Windows NT, NetWare, or any of several brands of UNIX.

	Server platform			
Client platform	**Windows 95 server**	**Windows NT server**	**UNIX server**	**NetWare server**
Windows 95	TCP/IP, Local	TCP/IP, NetBEUI	TCP/IP	TCP/IP, IPX/SPX
Windows NT	TCP/IP	TCP/IP, NetBEUI, Local	TCP/IP	TCP/IP, IPX/SPX
UNIX	TCP/IP	TCP/IP	TCP/IP	TCP/IP

TABLE 5.1 Matrix of connection supported protocols

Note The InterBase 5 client does not support IPX/SPX, though it can use TCP/IP to connect to a NetWare server. To use IPX/SPX, you must use the InterBase 5.1 or higher client.

Connection specification

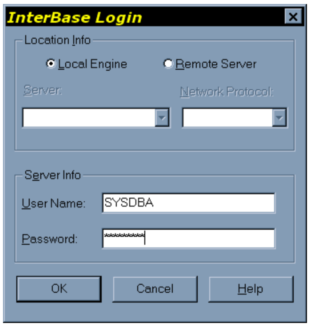
Before performing any database administration tasks, you must log in to a server. Once you log in, you can connect to databases residing on the server. You can switch context from one connected database to another by choosing the desired database in the Server Manager's database/server tree.

Logging in to a server

In the Server Manager for Windows, you can log in to a database server by clicking the Login toolbar button or by choosing **File | Server Login**. You can also choose Server Login by right-clicking an active server in the server/database tree pane.

The following dialog appears:

FIGURE 5.1 Server login dialog



User Name: SYSDBA

Password: ********

To log on to a local or remote server:

1. Open the Server Manager for Windows' Server Login dialog and select Local Engine or Remote Server.
2. If you chose Remote Server, type the name of the server in the Server text field, or click the drop-down arrow for a list of servers to which Server Manager has previously logged in, then choose a server name. Server Manager maintains a list of the last ten servers accessed in the file *C:\windows\interbas.ini*.

   Note The InterBase server name is the name of the database server machine. There is not a specific name for the InterBase server process itself. For example, if the server is running on the NT server *venus*, you enter this name in the Server field.
3. If you chose Remote Server, select a network protocol: one of TCP/IP, IPX/SPX, or NetBEUI. Protocols are valid only when the client and the server both support the protocol.
4. For either a local or remote connection, enter a username and password in the corresponding text fields. These must be the InterBase usernames and passwords that are stored in the security database *isc4.gdb* on the server. For convenience, Server Manager displays the last username and network protocol used.

   The username is significant to 31 characters and is *not* case-sensitive. Password is significant to eight characters and *is* case-sensitive.

All users must enter their username and password to log in to a server. The password is encrypted for transmission over the network. The username and password are verified against records in the security database. If a matching record is found, the login succeeds.

5. Click the OK button to log in to the server. Once you are logged in, Server Manager displays the server type and the version of the InterBase server in the summary information area.

IMPORTANT Initially, every server has only one authorized user with username SYSDBA. The SYSDBA must log on and add other authorized users. For more information about how to add new users, see **"User administration with Server Manager for Windows" on page 116**.

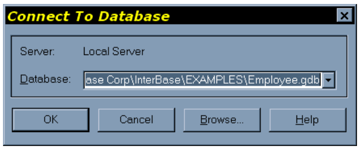
Connecting to a database

You can connect to a database in Server Manager using one of the following methods:

- Click the Database Connect toolbar button
- Choose **File | Database Connect**.
- Right click a database in the display tree and choose Database Connect from the menu.

The Connect to Database dialog appears:

FIGURE 5.2 Connect to Database dialog


Connect To Database

Server: Local Server

Database: ase Corp\InterBase\EXAMPLES\Employee.gdb

The Database text field shows the last database accessed on the current server. To choose a database, click the arrow to the right of the text field to display a drop-down list and select a database from the list, or enter a different database name. Choose OK to connect to that database.

If you enter a different name, Server Manager adds it to the list of saves the last ten databases to which it connected, maintained in the *C:\windows\interbas.ini* file. The database file name and path must be appropriate for the type of server it is on.

TIP There is no distinction between forward slashes ("/") and backslashes ("\") as directory separators. InterBase translates either type of slash to the proper directory separator for the type of server on which the database resides. As a result, C or C++ programmers may find it convenient to use forward slashes when writing code containing database specification strings, since this symbol does not need to be escaped.

After the database connection is established, the Server Manager toolbar and menus are active, and any actions you take apply to the selected database.

Connection specification examples

Here are some examples of connecting to databases on various types of servers.

- For a Windows NT or Windows 95 server, the database path name must contain the appropriate drive letter designation. For example:

  ```
  D:\users\accting\fin\accrec.gdb
  ```

- To connect to a database on a remote server by TCP/IP, use a colon to separate the server name from the path:

  ```
  venus:D:\users\accting\fin\accred.gdb
  ```

- To connect via NetBEUI (Windows NT servers only), use UNC notation:

  ```
  \\venus\D:\users\accting\fin\accred.gdb
  ```

- To connect via IPX/SPX (NetWare servers only) use the following notation:

  ```
  mars@vol2:\accting\fin\accred.gdb
  ```

- For a UNIX server, you must enter the complete and absolute directory path for the database. You must also provide the hostname; the SuperServer architecture necessarily disables the *local* database attachment configuration. For example:

  ```
  jupiter:/usr/accting/fin/accrec.gdb
  ```

Connection troubleshooting

This section describes some troubleshooting guidelines for issues related to network configuration and client/server connections. If you are having trouble connecting client to server over a network, use the steps listed below to diagnose the cause. On Windows, you can perform some of these tests using the Communications Diagnostic Tool (ComDiag). See **"Communication Diagnostic tool for Windows" on page 106** for more information on using ComDiag.

Connection Refused errors

If the client fails to reach the server host at all, or the *gds_db* service fails to answer, you may get a “connection refused” error. Below is a checklist that you can use to diagnose the source of this error.

Is there low-level network access between the client and server?

You can quickly test whether the client cannot reach the server because of a physically disconnected network or improper network software configuration, by using the **ping** command. Usage is:

```
ping servername
```

Error messages from **ping** indicate that there is a network problem. Check that the network is plugged in, that the network wires are not damaged, and that the client and server software is properly configured.

Test connectivity from the client in question to another server; if it succeeds, this may rule out improper network configuration on the client.

Test connectivity from another client to the InterBase server host; if it succeeds, this may rule out improper network configuration on the server.

Can the client resolve the server’s hostname?

InterBase clients must specify the server by name, not by IP address. Therefore, the client must be able to resolve the server’s hostname. For TCP/IP, this is done either by maintaining a *hosts* file on the client with the mappings of hostnames to IP addresses, or by the client querying a DNS server or WINS server to resolve this mapping. Make sure the name server has a correct entry for the server host in question.

Is the server behind a firewall?

If the database server is behind a software or hardware firewall, all network traffic may be restricted and the client may not be able to reach the server at all. Some firewalls permit or restrict traffic based on the port to which the client attempts to connect. Because of this, it is not conclusive whether a given service can reach the server. Neither is it an indication of connectivity if the client can resolve the IP address; that merely indicates that the client can reach a name server that resolves the InterBase server host’s name.

If the client is separated from the server by a firewall, the client cannot connect.

Are the client and server on different subnets?

NetBEUI cannot route network traffic between subnets. Other protocols can also be configured to restrict traffic between subnets. If the client and server are on a complex network with multiple subnets, ask your network administrator if the network configuration allows you to route network traffic between the client and server in question using a given protocol.

Can you connect to a database locally?

To confirm that the **ibserver** process is running on the server and able to attach to your database, try a local database connection:

1. Log in to the console of the database server host, and run an application such as **isql** or Windows ISQL.

2. Attempt to connect to a database without specifying a hostname: list just the path. In Windows ISQL, the Local Engine option is grayed out in the Database Connect dialog if the **ibserver** process is not running.

ComDiag also has a local database attachment test. See **“DB Connection tab” on page 107** for details.

Note Local connection mode is not available on UNIX servers or NetWare servers.

Can you connect to a database loopback?

You can simulate a client/server connection and test the server’s configuration without the additional variable of the client configuration and intervening network by connecting in a *loopback* mode.

1. Log in to the console of the database server host and run an application such as **isql** or Windows ISQL.

2. Attempt to connect to the database using a remote connection specification, even though the server named is also the client host.

Whether this test fails or succeeds, it helps to narrow the focus of further diagnostic tests. If it fails, you can infer that the server’s configuration is at fault. If it succeeds, you can infer that the server is not at fault and you can concentrate further tests on the client.

Note Loopback tests cannot be performed when using NetWare, because client applications run only on a remote client, not on the NetWare server.

Is the server listening on the InterBase port?

If the **ibserver** process on the server has not started, there is no answer to attempts to connect to the *gds_db* service (port 3050).

Start the **ibserver** process on the server. Use **ibmgr -start** on UNIX, or the Services control panel on NT. See **Chapter 4, "Server Configuration."**

Is the *services* file configured on client and server?

The services file must have correct entries to indicate the port number associated with the named service *gds_db*. This configuration must be accessible on the client as well as the server.

```
gds_db        3050/tcp          # InterBase Server
interserver   3060/tcp          # InterClient InterServer
```

On Windows NT, this file is in *C:\windows\system32\drivers\etc\services*.
On Windows 95, this file is in *C:\windows\services*.
On UNIX, this file is in */etc/services*.

In a UNIX environment with NIS, the NIS server can be configured to supply the *services* file to all NIS clients on UNIX workstations.

Is the UNIX inetd daemon configured for InterBase Classic architecture?

When running a version of InterBase that has the Superserver architecture (for instance, InterBase 5.1.1 for Solaris and HP-UX), you should check the */etc/inetd.conf* file to make sure that the **inetd** daemon is not configured to listen on the `gds_db` service (port 3050). In InterBase 5, the **ibserver** process takes over the task of listening on the port, and if both **inetd** and **ibserver** attempt to listen, then there is a conflict and the result is that neither can successfully accept connection requests.

Make the following change:

- Use a text editor to remove the line in */etc/inetd.conf* that mentions the `gds_db` service
- Restart **inetd** by sending it a HUP signal

The installation script in InterBase 5 for UNIX is supposed to perform this task, but if something goes wrong, or the */etc/inetd.conf* file is restored to its configuration for InterBase Classic, you need to correct the configuration.

For InterBase versions that run in Classic mode (for instance, SCO OpenServer and Linux), check to make sure the */etc/inetd.conf* file does have an entry for `gds_db`, and restart **inetd** with `kill -HUP` to make sure **inetd** is using the current configuration in */etc/inetd.conf*.

Connection Rejected errors

If the client reaches the server host and the *gds_db* service answers but you still cannot attach to a database, it can result in a “connection rejected” error. Below is a checklist that you can use to diagnose the source of this error.

Did you get the correct path to the database?

Verify that you supplied the correct path to the database file. Keep in mind:

- On NT, you must supply the drive letter with the path.
- On UNIX, paths are case-sensitive.
- Slash (“/”) vs. backslash (“\”) does not matter, unless you need to use double-backslashes in string literals in C or C++ code.

Is UNIX host equivalence established?

To use the UNIX user-equivalence feature, there must be a *trusted host* relationship between the client and the server. See **“Users on UNIX” on page 113**.

Is the database on a networked filesystem?

A database file must not reside on an NFS filesystem or a mapped drive. When the **ibserver** process finds such a case, it either denies the connection or passes the connection request on to the InterBase service running on the file server. See **“Networked filesystems” on page 129** for more details.

To correct this situation, move your database to a filesystem on a hard disk that is physically local to the database server.

Are the user and password valid?

The client application must use a valid user and password combination that matches an entry in *isc4.gdb*. Make sure you are using a valid user and password for that server.

Does the server have permissions on the database file?

The **ibserver** process must have permission to read and write the database file at the operating system level. Check the permissions on the database file, and the uid of the **ibserver** process. (On UNIX, you have the option of running **ibserver** as user *interbase*, a non-superuser uid.)

The *isc4.gdb* database that contains users and passwords must also be writable by the **ibserver** process.

Does the server have permissions to create files in the InterBase home directory?

The **ibserver** process must have write permission in the InterBase directory (by default, */usr/interbase* on UNIX, *C:\Program Files\InterBase Corp\InterBase* on Windows). The server process must be able to write to, and perhaps create, the *interbase.log* file and other temporary files.

Other errors

Unknown Win32 error 10061

This error is often attributed to missing server-access license for the InterBase software on the server host. Make sure you have licensed InterBase server to allow clients to connect from the network. See **"Licensing" on page 67**.

When all else fails...

If these troubleshooting guidelines have not helped you to correct your networking issues, contact InterBase Software Corp. technical support.

Communication Diagnostic tool for Windows

The InterBase Communication Diagnostic Tool (also called ComDiag) lets you troubleshoot your communications setup by testing for existing or potential network problems.

Winsock tab

Use this property sheet to test your Winsock TCP/IP connectivity.

To run a Winsock test

1. Enter the name of an InterBase server in the Host field.
2. Choose a service name or number from the dropdown Service list.
3. Click Test, then review the output in the Results window.

Sample results:

```
Path Name = C:\WINNT40\System32\wsock32.dll
Size      = 20240 Bytes
File Time = 23:00:00
File Date = 08/02/1996
Version   = 4.0.1371.1
This module has passed the version check.

Initialized Winsock.

Attempting connection to: jupiter.
Socket for connection obtained.
Connected to host ’jupiter’, port 21.
Socket closed successfully.
Winsock deinitialized successfully.

Winsock Communication Test Passed!
```

If the error message is...	Then check...
Failed to find named port	• Your *services* file to be sure there is an entry for *gds_db* in the form: **gds_db 3050/tcp**
Failed to connect to host	• Hostname, port 3050 • The InterBase Server to make sure it is installed properly, is running, and is configured for TCP/IP
Failed to resolve hostname	• Hostname • Your *hosts* file or DNS to be sure it has an entry for the server • That you used a hostname and not an IP address
Unavailable database	• Whether the InterBase server is running; the server must be running before attempting a database connection

TABLE 5.2 Using ComDiag to diagnose connection problems

DB Connection tab

This test lets you connect to an InterBase database using the InterBase client libraries. It is the most basic test of InterBase operation and is generally used only after confirmation that the underlying network is working correctly.

To run a DB Connection test

1. Choose a server type—Local or Remote—to which you want to connect. If you choose Remote, you have to enter the name of the Server and the Network Protocol (choose from the dropdown listbox). If you choose Local, server and protocol aren't needed, so the fields are grayed.

   Note InterBase Server must be running on your machine before you can connect to a local database.

2. In the Database Info group, enter the name of a Database on the selected server, along with the required User Name and Password.

   Notes Include the full drive and directory path to the database.
   NT share names or mapped drive letters are not allowed.

3. Click Test, then review the output in the Results window.

Sample output (local connection)

```
Path Name = C:\WINNT40\System32\gds32.dll
Size      = 385024 Bytes
File Time = 04:00:00
File Date = 09/10/1997
Version   = 5.5.0.589
This module has passed the version check.

Attempting to attach to C:\Program Files\InterBase Corp
      \InterBase\Examples\employee.gdb
   Attaching ...Passed!
   Detaching ...Passed!

InterBase versions for this connection:
InterBase/x86/Windows NT (access method), version “WI-V5.5.5”
on disk structure version 9.1

InterBase Communication Test Passed!
```

NetBEUI tab

Use this property sheet to test NetBEUI connectivity between the client and the server. This test checks the network, but does not use the InterBase client library.

To run a NetBEUI test

1. Enter the name of a Windows NT server on which InterBase has been installed
2. Click Test, and review the output in the Results window.

Sample output (NetBEUI connection):

```
Path Name = C:\WINNT40\System32\netapi32.dll
Size      = 224528 Bytes
File Time = 23:00:00
File Date = 08/02/1996
Version   = 4.0.1373.1
This module has passed the version check.

Attempting to attach to venus using NetBEUI.

Attached successfully to venus using Named Pipes.
Closing the Pipe to venus.

NetBEUI Communication Test Passed!
```

Notes

- NetBEUI is supported on Windows NT and Windows 95 clients, but only Windows NT supports NetBEUI as a server.
- The connection may fail if a Microsoft Windows network is not the default network for the client. You should also be logged into the MS Windows network with a valid NT user name and password.

CHAPTER

6

Database Security

InterBase provides several methods to configure and enforce security. This chapter gives an overview of these options. The user administration tools are covered here, but SQL statements for configuring privileges are in other InterBase books; these passages are referenced where appropriate.

Security model

Security for InterBase relies on a central database for each server host. This database contains legitimate users who have permission to connect to databases and InterBase services on that host. The database also contains an encrypted password for the user. This user and password applies to any database on that server host.

Before performing any database administration tasks, you must first log in to a server. Once you log in to a server, you may then connect to databases residing on the server.

The username is significant to 31 characters and is not case-sensitive. Password is significant to eight characters and is case-sensitive.

All users must enter their username and password to log in to a server. The password is encrypted for transmission over the network. The username and password are verified against records in the security database. If a matching record is found, the login succeeds.

The SYSDBA user

Every InterBase server has a SYSDBA user, with default password “masterkey”. SYSDBA is a special user that can bypass normal SQL security, and perform special tasks such as database shutdowns.

Initially, SYSDBA is the only authorized user on a server; the SYSDBA must authorize all other users on the server. Only the SYSDBA user can update the security database to add new users, delete users, or modify user configurations. SYSDBA can use the tools **gsec** and Server Manager for Windows to authorize a new user by assigning a username and password in the security database.

IMPORTANT It is *strongly* recommended that you change the password for SYSDBA as soon as possible after installing InterBase. If you do not alter the SYSDBA password, unauthorized users can easily guess it and none of your databases are secure.

Other users

You can create other users on a per-server basis. Use **gsec** or Server Manager for Windows to create, modify, or remove users from the *isc4.gdb* security database. These users are authorized to connect to any database on that database server host. It is a common design strategy to create a distinct InterBase user for each human who uses the databases on your server. However, other strategies are also legitimate. For example:

- Create one InterBase user for an entire group of people to use, in order to simplify password administration. For example, a user FINANCE may satisfy the access needs for any and all staff in a financial analysis team. This team only needs to remember one password between them.
- Create one InterBase user for a group of people to use, as warranted by requirements of distinct privilege configurations. For example, if Erin and Manuel have identical access to the data within a database, they can easily use the same InterBase user.

Users on UNIX

If both the client and the server are running UNIX, you can allow UNIX usernames access to databases by configuring the server host to treat the client host as a *trusted host*.

To establish a trusted host relationship between two hosts, add an entry in */etc/hosts.equiv* or */etc/gds_hosts.equiv* on the server. The former file establishes trusted host status for any service (for example, **rlogin**, **rsh**, and **rcp**); the latter file establishes trusted host status for InterBase client/server connections only. The format of entries in both files is identical; see your operating system documentation on *hosts.equiv* for details.

The login of the client user must exist on the server. In addition to the *hosts.equiv* method of establishing a trusted host, the you can also use the *.rhosts* file in the home directory of the account on the server that matches the account on the client.

The InterBase client library defaults to using the current client's UNIX login as the InterBase login only when the client specifies no username through any of the following methods:

- Database parameter buffer (*dpb*) parameters—see the *API Guide*
- Command-line options—for example, *-user* options of **isql** or another utility
- Environment variables—see **"ISC_USER and ISC_PASSWORD" on page 91**

Notes

- This feature is not implemented in InterBase 5 on a Windows NT server, because NT does not implement a trusted host mechanism as UNIX does.
- Windows clients cannot be treated as trusted hosts by UNIX servers.

Security database isc4.gdb

Every user of an InterBase server requires an entry in the *isc4.gdb* security database. The security utility, **gsec**, lets you display, add, modify, or delete information in *isc4.gdb*. Server Manager for Windows provides a graphical interface for the same functionality. The following table describes the contents of *isc4.gdb*:

Column	Required?	Description
User name	Yes	The name that the user supplies when logging in. Maximum length is 31 characters.
Password	Yes	The user's password. Case sensitive. Only the first eight characters are significant. Maximum length is 32 characters.
UID	No	An integer that specifies a user ID
GID	No	An integer that specifies a group ID
Full name	No	User's real name (as opposed to login name)

TABLE 6.1 Format of the *isc4.gdb* security database

SQL privileges

Connecting to a database does not automatically include privileges to modify or even view data stored within that database. Privileges must be granted explicitly; users cannot access any database objects until they have been granted privileges. Privileges granted to the special "user" PUBLIC apply to all users.

For full description of syntax of SQL privileges, see the *Language Reference* and *Data Definition Guide*.

Groups of users

InterBase 5 implements features for assigning SQL privileges to groups of users. SQL roles are implemented on a per-database basis. UNIX groups are implemented on a server-wide basis, using the UNIX group mechanism; this feature is not available on the Windows 95/NT or NetWare platforms.

ANSI SQL 3 roles

InterBase 5 supports SQL group-level security as described in the *ISO-ANSI Working Draft for Database Language*. For syntax of SQL ROLEs, see the *Language Reference* and *Data Definition Guide*.

Implementing roles is a four-step process. The role must first be declared with CREATE ROLE. Then privileges on specific tables and columns are assigned to the role using the GRANT statement. Next, the role is granted to users, again with the GRANT statement. Finally, the role must be stated in the connection. To acquire the privileges assigned to a role, a user must first have been granted the role and then must specify that role at connection time.

A user can belong to only one role per connection to the database, and cannot change role while connected. To change role, the client must disconnect and reconnect, specifying a different role name.

You can adopt a role when connecting to a database by any one of the following means:

- To specify a role when attaching to a database through Windows ISQL, display the Database Connect dialog and type a rolename in the Role field.
- To specify a role programmatically upon connection using the InterBase API, use the dpb parameter *isc_dpb_sql_role_name*. See chapter 4 of the *API Guide*.
- To specify a role for a connection made by an embedded SQL application or **isql** session, use the ROLE *rolename* clause of the CONNECT statement. See the statement reference for CONNECT in the *Language Reference*.
- To specify a role for the attachment made by **gbak** or **gsec**, using the **-role** *rolename* option for either of those command-line utilities.

Note Other methods of connecting to a database do not currently provide the means to specify a role name. Notably, applications using the BDE, including Delphi, JBuilder, and C++Builder, have no property by which they can specify a role name. Also, the ODBC driver that currently ships with InterBase does not recognize roles.

UNIX groups

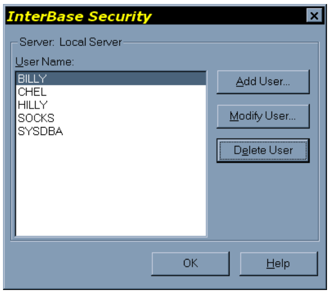
Operating system-level groups are implicit in InterBase security on UNIX, similarly to the way UNIX users automatically supplement the users in *isc4.gdb*. For full description of usage and syntax of using UNIX groups with InterBase security, see the *Language Reference* and *Data Definition Guide*.

Note Integration of UNIX groups with database security is not an SQL standard feature.

User administration with Server Manager for Windows

Choose **Tasks | User Security** or the popup menu from the server/database tree to open the InterBase Security dialog:

FIGURE 6.1 InterBase security dialog



The usernames of all authorized users are displayed in the list.

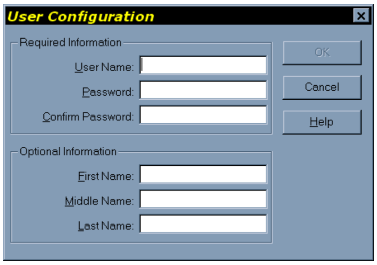
This dialog enables you to:

- View the list of authorized users for the server.
- Add a user (if you are SYSDBA).
- Modify users' passwords and optional information (if you are SYSDBA).
- Delete a user (if you are SYSDBA).

Adding a user

To add a new user, click the Add User button. This opens the User Configuration dialog:

FIGURE 6.2 User configuration dialog



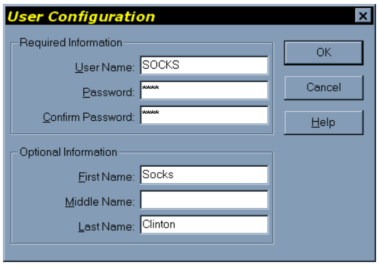
Type the new username in the username text field, and then type the user's password in both the Password and the Confirm Password text fields. Do not create usernames containing spaces. Add any desired optional information in the corresponding text fields, and then click OK.

Note Usernames can be up to 31 characters long and are not case sensitive. Passwords *are* case sensitive. Only the first eight characters of the password are significant.

Modifying user configurations

To change a user's password, select the username to highlight it, and then click the Modify button. Alternatively, double-click the username. This opens the User Configuration dialog for that user. For example:

FIGURE 6.3 Example user configuration



User Name: SOCKS
Password: *****
Confirm Password: *****
First Name: Socks
Middle Name:
Last Name: Clinton

Change any of the entries except username and then click OK. If you change the password, you must enter the same password in the Password text field and the Confirm Password text field.

You may not modify a username. The only way to change a username is to delete the user and then add a user with the new name.

Deleting a user

To remove a user's InterBase privileges, select the username and click the Delete button. A confirmation dialog inquires, "Do you wish to delete user *username*?" If you choose OK, the user is no longer be authorized to access databases on the current server.

The SYSDBA may delete the SYSDBA user. In this case, no new users may be added to the database, and existing user configurations may not be modified. This is not recommended. If you do delete SYSDBA, you must reinstall InterBase to restore the *isc4.gdb* security database.

User administration with the InterBase API

Authors of InterBase applications can now add, delete, and modify users programmatically using three new API functions:

Function	Description
isc_add_user( )	Adds a user record to the password database
isc_delete_user( )	Deletes a user record from the password database
isc_modifiy_user( )	Modifies a user record in the password database

TABLE 6.2 InterBase API functions for user administration

There is no function for displaying a list of current users authorized for the server, since this can be accomplished by querying the security database.

See the *API Guide* for details and examples of using these API functions. See the *Language Reference* for new InterBase error codes associated with user configuration tasks.

IMPORTANT Even though **isc_delete_user()** allows SYSDBA to delete the SYSDBA account, do not do that. If you do delete SYSDBA, you must reinstall InterBase to restore the *isc4.gdb* security database.

gsec command-line tool

The InterBase command-line security utility is **gsec**. This utility is used in conjunction with the security database *isc4.gdb*, to specify user names and passwords for an InterBase server. This tool duplicates the functionality of **Tasks | User Security** in Server Manager for Windows.

The security database, *isc4.gdb*, resides in the InterBase install directory. To connect to a database on the server, users must specify a user name and password, which are verified against information stored in *isc4.gdb*. If a matching row is found, the connection succeeds.

IMPORTANT Only the superuser can run **gsec**. To do this, use one of the following methods:

- Invoke the command as:

```
gsec -user sysdba -password masterkey
```

- Define the ISC_USER and ISC_PASSWORD environment variables for sysdba before you invoke the command.
- Run **gsec** when you are logged in as root on UNIX or Administrator on Windows NT.

To use **gsec** interactively, type **gsec** at the NT command prompt. The NT prompt changes to `GSEC>`, indicating that you are in interactive mode. To quit an interactive session, type `QUIT`.

Running gsec remotely

You can use **gsec** on a client host to administer users in a security database on a remote server. Use the *-database* option with a remote database specification to connect to a remote *isc4.gdb*. For example:

```
gsec -database jupiter:/usr/interbase/isc4.gdb
```

Security utility commands

The following table summarizes **gsec** commands. The initial part of each command is required. The part in brackets is optional.

Command	Description
di[splay]	Displays all rows of *isc4.gdb*.
di[splay] *name*	Displays information only for user *name*.
a[dd] *name* -pw *passwd* [*option argument*] [*option argument* ...]	Adds user *name* to *isc4.gdb* with password *string*. Each *option* and corresponding *argument* specifies other data associated with the user, as shown in **Table 6.4, "gsec options."**
mo[dify] *name* [*options*]	Like add, except that *name* already exists in *isc4.gdb*.
de[lete] *name*	Deletes user *name* from *isc4.gdb*.
h[elp] or ?	Displays **gsec** commands and syntax.
q[uit]	Quits the interactive session.

TABLE 6.3 Summary of **gsec** commands

Displaying the security database

To see the contents of *isc4.gdb*, enter the DISPLAY command at the `GSEC>` prompt. All the rows in the security database are displayed:

```
GSEC> display
user nameuid     gid full name
----------------------------------------------
FRED       123    345 Fred Flintstone
BARNEY     123    345 Barney Rubble
BETTY      123    345 Betty Rubble
```

Note that passwords are never displayed.

Adding entries to the security database

To add users to the security database, use the *add* command:

```
a[dd] name -pw passwd [options]
```

followed by a user name, the *-pw* option followed by a password, and any other options, as shown in the following table. The password is case sensitive. None of the other parameters are case sensitive.

For each option, the initial letter or letters are required and optional parts are enclosed in brackets. Each option must be followed by a corresponding argument, a string that specifies the data to be entered into the specified column in *isc4.gdb*.

Option	Meaning
-password or -pa *string*	Password of user who is performing the change
-user *string*	User who is performing the change
-pw *string*	Target user password
-uid *integer*	Target user ID
-gid *integer*	Group ID for target user
-fname *string*	First Name for target user
-mname *string*	Middle Name for target user
-lname *string*	Last Name for target user

TABLE 6.4 **gsec** options

Note The `-pa` switch specifies the root or SYSDBA password; `-pw` specifies the password for a user being added or modified.

For example, to add user *jones* and assign the password *welcome*, enter:

```
GSEC> add jones -pw welcome
```

Use *display* to verify the entry. An unassigned UID or GID defaults to 0:

```
GSEC> display
user name     uid     gid     full name
----------------------------------------------
JONES         0       0
```

For example, to add authorization for a user named Cindi Brown with user name *cbrown* and password “coffee2go”, use the following **gsec** command:

```
GSEC> add cbrown –pw coffee2go –fname cindi –lname brown
```

To verify the new entry, display *isc4.gdb*:

```
GSEC> display
```

```
user name     uid     gid     full name
----------------------------------------------
JONES         0       0
CBROWN        0       0       CINDI  BROWN
```

gsec stores the user name in uppercase regardless of how it is entered.

Modifying the security database

To change existing entries in the security database, use the *modify* command. Supply the user name for the entry to change, followed by the option indicating the items to change and the corresponding values to which to change them.

For example, to set the user ID of user *cbrown* to 8 and change the first name to *Cindy*, enter the following commands:

```
GSEC> modify cbrown –uid 8 –fname cindy
```

To verify the changed line, use *display* followed by the user name:

```
GSEC> display cbrown
user name    uid    gid    full name
----------------------------------------------
CBROWN       8      0      CINDY BROWN
```

Note To modify a user name, first delete the entry in *isc4.gdb*, then enter the new user name and re-enter the other information.

Deleting entries from the security database

To delete a user's entry in *isc4.gdb*, use *delete* and specify the user name:

```
GSEC> delete cbrown
```

You can confirm that the entry has been deleted with the *display* command.

Using gsec from the command prompt

To use **gsec** from the NT command prompt, precede each command with **gsec** and prefix each **gsec** command with a hyphen (-). For example, to add user *aladdin* and assign the password, "sesame", enter the following at the command line:

```
C:> gsec –add aladdin –pw sesame
```

To display the contents of *isc4.gdb*, enter:

```
C:> gsec –display
```

gsec error messages

Error Message	**Causes and Suggested Actions to Take**
Add record error	The *add* command either specified an existing user, used invalid syntax, or was issued without appropriate privilege to run **gsec**. Change the user name or use *modify* on the existing user.
<*string*> already specified	During an *add* or *modify*, you specified data for the same column more than once. Retype the command.
Ambiguous switch specified	A command did not uniquely specify a valid operation.
Delete record error	The *delete* command was not allowed. Check that you have appropriate privilege to use **gsec**.
Error in switch specifications	This message accompanies other error messages and indicates that invalid syntax was used. Check other error messages for the cause.
Find/delete record error	Either the *delete* command could not find a specified user, or you do not have appropriate privilege to use **gsec**.
Find/display record error	Either the *display* command could not find a specified user, or you do not have appropriate privilege to use **gsec**.
Find/modify record error	Either the *modify* command could not find a specified user, or you do not have appropriate privilege to use **gsec**.
Incompatible switches specified	Correct the syntax and try again.
Invalid parameter, no switch defined	You specified a value without a preceding argument.
Invalid switch specified	You specified an unrecognized option. Fix it and try again.
Modify record error	Invalid syntax for *modify* command. Fix it and try again. Also check that you have appropriate privilege to run **gsec**.

TABLE 6.5 **gsec** security error messages

Error Message	**Causes and Suggested Actions to Take**
No user name specified	Specify a user name after *add*, *modify*, or *delete*.
Record not found for user: <*string*>	An entry for the specified user could not be found. Use *display* to list all users, then try again.
Unable to open database	The *isc4.gdb* security database does not exist or cannot be located by the operating system.

TABLE 6.5 **gsec** security error messages (*continued*)

CHAPTER

7 Database Configuration and Maintenance

This chapter describes configuration and maintenance issues for individual databases. It includes topics of files and filesystems, database shadowing, and database tuning options.

Database files

InterBase database files are in many cases self-contained. All the data and indexes are maintained as data structures within one type of file. The transaction log is also kept within this file.

InterBase permits external files to be used as *external tables*. These tables are optional, and they are limited in their functionality: you may execute only SELECT and INSERT operations on external tables; you cannot define indexes on external data; they are outside of the control of the multi-generational architecture.

You may extend the functions available in InterBase database metadata by creating libraries of functions compiled in your language of choice. You can compile functions into a dynamic library (called a DLL on Windows, and a shared library on UNIX) and use them in queries, stored procedures, triggers, views, etc.

InterBase dynamically creates files in the temporary file space for scratch space during sorting operations involving large amounts of data. See **"Temporary file management" on page 92** for details on temporary files.

File naming conventions

InterBase database files are given a file extension of *.gdb* by convention, though the software does not enforce this and you may choose to use another file extension. For purposes of this documentation, assume that *.gdb* refers to an InterBase database file type.

InterBase is available on a wide variety of platforms. In most cases users in a heterogeneous networking environment can access their InterBase database files regardless of platform differences between client and server machines if they know the target platform's file naming conventions.

Because file naming conventions differ widely from platform to platform, and because the core InterBase documentation set is the same for each of these platforms, all file names in text and in examples are restricted to a base name with a maximum of eight characters, with a maximum extension length of three characters. For example, the sample database on all servers is referred to as *employee.gdb*.

Generally, InterBase fully supports each platform's file naming conventions, including the use of node and path names. InterBase, however, recognizes two categories of file specification in commands and statements that accept more than one file name. The first file specification is called the *primary file specification*. Subsequent file specifications are called *secondary file specifications*. Some commands and statements place restrictions on using node names with secondary file specifications.

In syntax, file specification is denoted as follows:

```
'<filespec>'
```

Primary file specifications

InterBase syntax always supports a full file specification, including optional node name and full path, for primary file specifications. For example, the syntax notation for CREATE DATABASE appears as follows:

```
CREATE {DATABASE | SCHEMA} '<filespec>'
[USER 'username' [PASSWORD 'password']]
[PAGE_SIZE [=] int]
[LENGTH [=] int [PAGE[S]]]
[DEFAULT CHARACTER SET charset]
```

In this syntax, the *filespec* that follows CREATE DATABASE supports a node name and path specification, including a platform-specific drive or volume specification.

Secondary file specifications

For InterBase syntax that supports multiple file specification, such as CREATE DATABASE, all file specifications after the first one are *secondary*. Secondary file specifications generally cannot include a node name, but can specify a full path name. For example, the syntax notation for CREATE DATABASE appears as follows:

```
CREATE {DATABASE | SCHEMA} '<filespec>'
[USER 'username' [PASSWORD 'password']]
[PAGE_SIZE [=] int]
[LENGTH [=] int [PAGE[S]]]
[DEFAULT CHARACTER SET charset]
[<secondary_file>]
<secondary_file> = FILE '<filespec>' [<fileinfo>] [<secondary_file>]
<fileinfo> = LENGTH [=] int [PAGE[S]] | STARTING [AT [PAGE]] int
[<fileinfo>]
```

The database restore task in Server Manager for Windows and in the **gbak** command-line utility permits you to create a multifile database. The only way to alter the file size allocation after a database has been created is to back up and restore the database file.

On-disk structure (ODS)

Each release of InterBase has characteristic features in its internal file format. To distinguish between the file formats, InterBase records an on-disk structure (ODS) number in the *.gdb* file. In general, major ODS versions (those incrementing the number the left of the decimal point) introduce features that are not backward compatible with earlier ODS versions. As a result, earlier InterBase software cannot operate on a database file with a later ODS.

When you create a new database or restore a **gbak** file, the resulting *.gdb* file always has the most recent ODS version that the software supports. The software does not alter the ODS of an older *.gdb* file or automatically promote it to the newest ODS (but see below). Therefore, to upgrade the ODS, you must back up and restore the database.

InterBase 5 uses ODS version 9.1. New features in this ODS that are not recognized by earlier software include indexes with automatic garbage collection, SQL roles, and cascading declarative referential integrity.

The InterBase 5 software reads the ODS of InterBase 4.0 (ODS version 8.0) transparently. The first time the InterBase 5 server connects to a database, the ODS is promoted to 8.2.

Multifile databases

InterBase supports databases that span multiple files and multiple filesystems. You can add additional files to the database without having to take it off line.

For example, in **isql**:

```
CONNECT "first.gdb";
ALTER DATABASE ADD FILE 'second.gdb' STARTING AT 50000;
```

Data distribution

InterBase starts writing data to *second.gdb* once *first.gdb* file fills up. In the example above, the first file will be 50,000 pages (pages are 1KB each by default) long, and the second file contains data in excess of the 50,000th page.

There is no guarantee that a given table resides entirely in one file or another. InterBase stores records based on available space within database files. Over time, records from a given table tend to spread over all the files in a multifile database.

Maximum number of files

InterBase allows up to 65,536 database files, including shadow files. Note that your operating system may have a much lower limit on the number of simultaneous open files that the **ibserver** process can have.

Application considerations

A multifile database is not the same thing as multiple single-file databases. The tables are all part of the same database they used to be in, but they may be stored across the multiple files. From your application's standpoint, they're all part of the same database and are accessed exactly the same way they would be in a single-file database.

Your application does not need to know about any files except the first one. Any time your database operations access/write data in the secondary files, the InterBase software takes care of it without requiring any special programming from your application. The application attaches to the database by specifying the path of the first file of the database; applications don't change.

Reorganizing file allocation

You can rearrange the sizes of the files of a multifile database during a **gbak** restore.

TIP Any database in a production environment should include a definition for at least one secondary file, even if the current size of the database does not warrant a multifile database. Data tend to accumulate without bound, and someday in the future your database might exceed your filesystem size, or the operating system's maximum file size. By defining a secondary file, you specify what action InterBase takes when the database grows beyond these limits. This means that the database administrator is freed from monitoring the database as it approaches the file size limit.

Read-only files and filesystems

InterBase database files must be writable by the **ibserver** process, and the files must reside on writable media. Even applications that use a database exclusively for read-only operations must start a transaction. The transaction states are kept in an internal inventory data structure within the *.gdb* file. Therefore any transaction against the database requires the ability to write to the transaction inventory.

Under both Windows NT and UNIX, database files must have file permissions that permit the user ID that the **ibserver** process is running as to write to the file. However, your operating environment or filesystem may be configured to create files by default to have limited file privileges. If you attempt to attach to a database and get an error of "unavailable database," first check to see if the *.gdb* file's permissions are such that the user ID of the **ibserver** process has write privilege.

TIP If you are designing a product to distribute including an InterBase database, put a **gbak** file of the database on CD-ROM. As part of the installation, restore the database to the workstation's hard disk.

Networked filesystems

An InterBase database must reside on a disk local to the server software that accesses it. The database file (including any secondary files and shadow files) may not reside on networked or remote filesystems (called mapped drives on Windows and NFS filesystems on UNIX). External tables and UDF libraries can reside on networked filesystems, but this practice is not recommended because networked filesystems may suffer from intermittent availability.

On UNIX, the InterBase software detects that a database file is located on an NFS filesystem. In this case, it invokes the remote access method to contact an InterBase server process running on the host that exported the filesystem. If there is no InterBase server software running on that node, any connection to the database fails.

Shadowing

InterBase lets you recover a database in case of disk failure, network failure, or accidental deletion of the database. The recovery method is called *disk shadowing*, or sometimes just *shadowing*. This chapter describes how to set up and use shadowing.This section describes the various tasks involved in shadowing, as well as the advantages and limitations of shadowing.

Tasks for shadowing

The main tasks in setting up and maintaining shadowing are as follows:

- Creating a shadow.

  Shadowing begins with the creation of a shadow. A *shadow* is an identical, physical copy of a database. When a shadow is defined for a database, changes to the database are written simultaneously to its shadow. In this way, the shadow always reflects the current state of the database. For information about the different ways to define a shadow, see **"Creating a shadow" on page 131**.

- Activating a shadow.

  If something happens to make a database unavailable, the shadow can be activated. *Activating* a shadow means it "takes over" for the database; the shadow becomes accessible to users as the main database. Activating a shadow happens either automatically or through the intervention of a DBA, depending on how the shadow was defined. For more information about activating a shadow, see **"Activating a shadow" on page 135**.

- Deleting a shadow.

  If shadowing is no longer desired, it can be stopped by deleting the shadow. For more information about deleting a shadow, see **"Dropping a shadow" on page 136**.

- Adding files to a shadow.

  A shadow can consist of more than one file. As shadows grow in size, files can be added to accommodate the increased space requirements. For more information about adding shadow files, see **"Adding a shadow file" on page 136**.

Advantages of shadowing

Shadowing offers several advantages:

- Recovery is quick. Activating a shadow makes it available immediately.
- Creating a shadow does not require exclusive access to the database.
- Shadow files use the same amount of disk space as the database. Log files, on the other hand, can grow well beyond the size of the database.
- You can control the allocation of disk space. A shadow can span multiple files on multiple disks.
- Shadowing does not use a separate process. The database process handles writing to the shadow.
- Shadowing can run behind the scenes and needs little or no maintenance.

Limitations of shadowing

Shadowing has the following limitations:

- Shadowing is not an implementation of *replication*. Shadowing is one-way writing, duplicating every write operation on the master database. Client applications cannot access the shadow file directly.
- Shadowing is useful only for recovery from hardware failures or accidental deletion of the database. User errors or software failures that corrupt the database are duplicated in the shadow.
- Recovery to a specific point in time is not possible. When a shadow is activated, it takes over as a duplicate of the database. Shadowing is an “all or nothing” recovery method.
- Shadowing can occur only to a local disk. Shadowing to a NFS filesystem or mapped drive is not supported. Shadowing to tape or other media is unsupported.

Creating a shadow

A shadow is created with the CREATE SHADOW statement in SQL. Because this does not require exclusive access, it can be done without affecting users. For detailed information about CREATE SHADOW, see the *Language Reference*.

Before creating a shadow, consider the following topics:

- The location of the shadow

A shadow should be created on a different disk from where the main database resides. Because shadowing is intended as a recovery mechanism in case of disk failure, maintaining a database and its shadow on the same disk defeats the purpose of shadowing.

- Distributing the shadow

  A shadow can be created as a single disk file called a shadow file or as multiple files called a shadow set. To improve space allocation and disk I/O, each file in a shadow set can be placed on a different disk.

- User access to the database

  If a shadow becomes unavailable, InterBase can either deny user access to the database until shadowing is resumed, or allow access even though database changes are not being shadowed. Depending on which database behavior is desired, the DBA creates a shadow either in auto mode or in manual mode. For more information about these modes, see **"Auto mode and manual mode" on page 134**.

- Automatic shadow creation

  To ensure that a new shadow is automatically created, create a conditional shadow. For more information, see "Creating a Conditional Shadow," in this chapter.

  The next sections describe how to create shadows with various options:

- Single-file or multifile shadows
- Auto or manual shadows
- Conditional shadows

These choices are not mutually exclusive. For example, you can create a single-file, conditional shadow in manual mode.

Creating a single-file shadow

To create a single-file shadow for database *employee.gdb*, enter:

```
SQL> CREATE SHADOW 1 '/usr/interbase/examples/employee.shd';
```

The name of the shadow file is *employee.shd*, and it is identified by the number 1. Verify that the shadow has been created by using the isql command SHOW DATABASE:

```
SQL> SHOW DATABASE;
    Database: employee.gdb
     Shadow 1: '/usr/interbase/examples/employee.shd' auto
    PAGE_SIZE 1024
    Number of DB pages allocated = 392
    Sweep interval = 20000
```

The page size of the shadow is the same as that of the database.

Creating a multifile shadow

If the size of a database exceeds the space available on one disk, create a multifile shadow and spread the files over several disks. To create a multifile shadow, specify the name and size of each file in the shadow set. For example,

```
SQL> CREATE SHADOW 1 'employee.shd' LENGTH 1000
CON> FILE 'emp1.shd' LENGTH 2000
CON> FILE 'emp2.shd' LENGTH 2000;
```

The previous example creates a shadow set consisting of three files. The primary file, *employee.shd*, is 1,000 database pages in length. The secondary files, identified by the FILE keyword, are each 2,000 database pages long.

Instead of specifying the page length of secondary files, you can specify their starting page. The previous example could be entered as follows:

```
SQL> CREATE SHADOW 1 'employee.shd' LENGTH 1000
CON> FILE 'emp1.shd' STARTING AT 1000
CON> FILE 'emp2.shd' STARTING AT 3000;
```

In either case, you can use SHOW DATABASE to verify the file names, page lengths, and starting pages for the shadow just created:

```
SQL> SHOW DATABASE;
   Database: employee.gdb
    Shadow 1: '/usr/interbase/examples/employee.shd' auto length 1000
    file /usr/interbase/examples/emp1.shd length 2000 starting 1000
    file /usr/interbase/examples/emp2.shd length 2000 starting 3000
   PAGE_SIZE 1024
   Number of DB pages allocated = 392
   Sweep interval = 20000
```

Note The page length you allocate for secondary shadow files need not correspond to the page length of the database's secondary files. As the database grows and its first shadow file becomes full, updates to the database automatically overflow into the next shadow file.

▸ *Auto mode and manual mode*

A shadow can become unavailable for the same reasons a database becomes unavailable (disk failure, network failure, or accidental deletion). If a shadow becomes unavailable, and it was created in *auto mode*, database operations continue automatically without shadowing. If a shadow becomes unavailable, and it was created in *manual mode*, further access to the database is denied until the DBA intervenes. The benefits of auto mode and manual mode are compared in the following table:

Mode	Advantage	Disadvantage
Auto	Database operation is uninterrupted	Creates a temporary period when the database is not shadowed The DBA might be unaware that the database is operating without a shadow
Manual	Prevents the database from running unintentionally without a shadow	Database operation is halted until the problem is fixed Needs intervention of the DBA

TABLE 7.1 Auto vs. manual shadows

AUTO MODE

The AUTO keyword directs the CREATE SHADOW statement to create a shadow in auto mode:

```
SQL> CREATE SHADOW 1 AUTO 'employee.shd';
```

Auto mode is the default, so omitting the AUTO keyword achieves the same result.

In auto mode, database operation is uninterrupted even though there is no shadow. To resume shadowing, it may be necessary to create a new shadow. If the original shadow was created as a conditional shadow, a new shadow is automatically created. For more information about conditional shadows, see **"Conditional shadows" on page 135**.

MANUAL MODE

The MANUAL keyword directs the CREATE SHADOW statement to create a shadow in manual mode:

```
SQL> CREATE SHADOW 1 MANUAL 'employee.shd';
```

Manual mode is useful when continuous shadowing is more important than continuous operation of the database. When a manual-mode shadow becomes unavailable, further attachments to the database are prevented. To allow database attachments again, enter the following command:

```
gfix -kill database
```

This command deletes metadata references to the unavailable shadow corresponding to *database*. After deleting the references, a new shadow can be created if shadowing needs to resume.

Conditional shadows

You can define a shadow such that if it replaces a database, the server will create a new shadow file, allowing shadowing to continue uninterrupted. A shadow defined with this behavior is called a *conditional shadow*.

To create a conditional shadow, specify the CONDITIONAL keyword with the CREATE SHADOW statement. For example,

```
SQL> CREATE SHADOW 3 CONDITIONAL 'atlas.shd';
```

Creating a conditional file directs InterBase to automatically create a new shadow. This happens in either of two cases:

- The database or one of its shadow files becomes unavailable.
- The shadow takes over for the database due to hardware failure.

Activating a shadow

When a database becomes unavailable, database operations are resumed by activating the shadow. To do so, use gfix with the *–activate* (or *–a*) option.

IMPORTANT Before activating a shadow, check that the main database is unavailable. If a shadow is activated while the main database is available, the shadow can be corrupted by existing attachments to the main database.

To activate a shadow, specify the path name of its primary file. For example, if database *employee.gdb* has a shadow named *employee.shd*, enter:

```
gfix -a employee.shd
```

After a shadow is activated, you should change its name to the name of your original database. Then, create a new shadow if shadowing needs to continue and if another disk drive is available.

Dropping a shadow

To stop shadowing, use the shadow number as an argument to the DROP SHADOW statement. For example,

```
SQL> DROP SHADOW 1
```

If you need to look up the shadow number, use the isql command SHOW DATABASE.

IMPORTANT DROP SHADOW deletes shadow references from a database's metadata, as well as the physical files on disk. Once the files have been removed from disk, there is no opportunity to recover them. However, a shadow is merely a copy of an existing database, so the new shadow is identical to the dropped shadow.

Adding a shadow file

If a database is expected to increase in size, consider adding files to its shadow. To add a shadow file, first use DROP SHADOW to delete the existing shadow, then use CREATE SHADOW to create a multifile shadow.

The page length you allocate for secondary shadow files need not correspond to the page length of the database's secondary files. As the database grows and its first shadow file becomes full, updates to the database automatically overflow into the next shadow file.

Database configuration using Server Manager for Windows

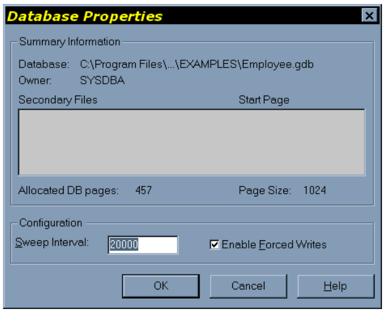
To perform database maintenance and configuration, first choose a database from the server/database tree:

1. Expand the server/database tree to display the database on which to perform maintenance.
2. In an expanded tree, click the database name to highlight it. The highlighted database is the one on which Server Manager operates, referred to as the *current database*. When a database is highlighted, the server on which the database resides becomes the *current server*. Any actions of Server Manager then affect that server.

The name of the current database and its owner are displayed in the summary information area in the left pane, along with the versions of the InterBase server, client, and access method.

To view and configure database properties, choose **Maintenance | Database Properties** from the Main Menu, toolbar, or popup menu. The Database Properties dialog appears:

FIGURE 7.1 Database properties dialog


Database Properties

Summary Information

Database: C:\Program Files\...\EXAMPLES\Employee.gdb

Owner: SYSDBA

Secondary Files	Start Page

Allocated DB pages: 457 Page Size: 1024

Configuration

Sweep Interval: 20000 ☑ Enable Forced Writes

This dialog contains a Summary Information area that displays properties but does not allow modification of them and a Configuration area that does allow modification of the parameters.

The Summary Information area displays:

- Database name.
- User name of the database owner.
- Database Page Size.
- Number of allocated pages.
- Secondary file names and sizes.

The configuration area displays and allows modification of:

- Sweep interval.
- Enabling of forced writes.

Sweep interval and automated housekeeping

Sweeping a database is a systematic way of removing outdated records from the database. Periodic sweeping prevents a database from growing too large. However, sweeping can also slow system performance.

As a DBA, you can tune database sweeping, balancing its advantages and disadvantages to best satisfy users' needs.

Overview of sweeping

InterBase uses a multi-generational architecture. This means that multiple versions of data records are stored directly on the data pages. When a record is updated or deleted, InterBase keeps a copy of the old state of the record and creates a new version. This can increase the size of a database.

GARBAGE COLLECTION

To limit the growth of the database, InterBase performs *garbage collection*, which frees up space allocated to outdated versions of a record. Whenever a transaction accesses a record, outdated versions of that record are garbage collected. Records that were rolled back are not be garbage collected. To guarantee that all records are garbage collected, including those that were rolled back, InterBase periodically "sweeps" the database.

AUTOMATIC HOUSEKEEPING

If a transaction is left in an active (unresolved) state, this is an "interesting" transaction. In a given database's transaction inventory, the first transaction with a state other than committed is known as the Oldest Interesting Transaction (OIT). If a client starts a new transaction and the transaction number is greater than a certain threshold past the number of the OIT, the InterBase server initiates a full sweep of the database. By default, this threshold is 20,000 transactions, and is configurable (see **"Setting the sweep interval" on page 139**).

Note It is a subtle but important distinction that the automatic sweep does *not* necessarily occur every 20,000 transactions. It is only when the *difference* between the OIT and the newest transaction reaches the threshold. If every transaction to the database is committed promptly, then this difference it is not likely to be great enough to trigger the automatic sweep.

The InterBase server process initiates a special thread to perform this sweep asynchronously, so that the client process can continue functioning, unaffected by the amount of work done by the sweep.

TIP Sweeping a database is not the only way to perform systematic garbage collection. Backing up a database achieves the same result, because the InterBase server must read every record, an action that forces garbage collection throughout the database. As a result, regularly backing up a database can reduce the need to sweep. This enables you to maintain better application performance. For more information about the advantages of backing up and restoring, see **"Benefits of backup and restore" on page 159**.

CONFIGURING SWEEPING

The Database Maintenance window enables you to control several aspects of database sweeping. You can:

- Change the automatic sweep interval.
- Disable automatic sweeping.
- Sweep a database immediately.

The first two functions are performed in the Database Properties dialog. The last is performed with the **Maintenance | Database Sweep** menu command.

Setting the sweep interval

To set the automatic sweep threshold, type the number of transactions between each database sweep in the Sweep Interval text field. For example, to set the sweep threshold to 10,000 transactions, type 10000 in the text field.

Sweeping a database can affect transaction start-up if rolled back transactions exist in the database. As the time since the last sweep increases, the time for transaction start-up may also increase. Lowering the sweep interval may help reduce the time for transaction start-up.

On the other hand, frequent database sweeps may reduce application performance. Raising the sweep interval may help improve overall performance. The DBA should weigh the issues for the affected applications and decide whether the sweep interval provides the desired database performance.

TIP Unless the database contains many rolled back transactions, changing the sweep interval has little effect on database size. As a result, it is more common for a DBA to tune the database by disabling sweeping and performing it at specific times. These activities are described in the next two sections.

Disabling automatic sweeping

To disable automatic sweeping, set the sweep interval to zero (0). Disabling automatic sweeping is useful if:

- Maximum throughput is important. Transactions are never delayed by sweeping.
- You want to schedule sweeping at specific times. You can manually sweep the database at any time. It is common to schedule sweeps at a time of least activity on the database server, to avoid competing for resources with clients.

Performing an immediate database sweep

To perform a database sweep, choose **Maintenance | Database Sweep** from the Main Menu, toolbar, or popup menu.

This operation runs an immediate sweep of the database, releasing space held by records which were rolled back and by out-of-date record versions. Sweeps are also done automatically at a specified interval.

Sweeping a database does not strictly require it to be shut down. You can perform sweeping at any time, but it can impact system performance and should be done when it inconveniences users the least.

If a sweep is performed as an exclusive operation on the database, there is additional tuning that the procedure performs. As long as there are no outstanding active transactions, the sweep updates the state of data records and the state of the inventory of past transactions. Non-committed transactions are finally rendered obsolete, and internal data structures need not track them in order to maintain snapshots of database versions. The benefit of this is a reduction of memory use, and a noticeable performance improvement.

Configuring the SuperServer cache

You can set the size of the SuperServer cache for a server. You can then modify that size for a specific database or for a specific ISQL connection.

Default cache size per database

The *buffers* parameter of the **gfix** utility sets the default number of cache pages for a specific database:

```
gfix –buffers n database_name
```

This sets the number of cache pages for the specified database to *n*, overriding the server value, which by default is 256 pages.

Default cache size per server

You can configure the default number of pages used for the SuperServer cache. By default, the database cache size is 256 pages per database. You can modify this default by changing the value of DATABASE_CACHE_PAGES in the $INTERBASE/*isc_config* (UNIX) or *ibconfig* (Windows) file. On Windows 95 or Windows NT platforms, you can also set it in the IB Settings page of the InterBase Server Properties dialog or the Server Configuration dialog. When you change this setting, it applies to every active database on the server.

In InterBase 5, it is possible to set the default cache size for each database using the **gfix** utility. This approach permits much greater flexibility, and reduces the risk that memory is overused, or that database caches are too small. *InterBase Software Corp. strongly recommends that you use* **gfix** *to set cache size instead of* DATABASE_CACHE_PAGES.

Default cache size per ISQL connection

To configure the number of cache pages for the duration of one **isql** connection, invoke **isql** with the following option:

```
isql –c n database_name
```

n is the number of cache pages to be used as the default for the session; *n* overrides any values set by DATABASE_CACHE_PAGES or **gfix** and must be greater than 9.

A CONNECT statement entered in an isql query accepts the argument CACHE *n*. (Refer to the discussion of CONNECT in the *Language Reference* manual for a full description of the CONNECT function). For example:

```
ISQL> connect database_name CACHE n
```

The value *n* can be any positive integer number of database pages. If a database cache already exists in the server because of another attachment to the database, the cache size is increased only if *n* is greater than current cache size.

Setting cache size in applications

You can still use the *isc_dpb_num_buffers* parameter to set cache size in a database parameter buffer (DPB).

Verifying cache size

To verify the size of the database cache currently in use, execute the following commands:

```
ISQL> CONNECT database_name;
ISQL> SET STATS ON;
ISQL> COMMIT;
Current memory = 415768
Delta memory = –2048
Max memory = 419840
Elapsed time = 0.03 sec
Buffers = 256
Reads = 0
Writes 2
Fetches = 2
ISQL> QUIT;
```

The empty COMMIT command prompts **isql** to display information about memory and buffer usage. The “Buffers” line specifies the size of the cache for that database.

Forced writes vs. buffered writes

When an InterBase Server performs *forced writes* (also referred to as synchronous writes), it physically writes data to disk whenever the database performs an (internal) write operation.

If forced writes are not enabled, then even though InterBase performs a write, the data may not be physically written to disk, since operating systems buffer disk writes. If there is a system failure before the data is written to disk, then information can be lost.

Performing forced writes ensures data integrity and safety, but slow performance. In particular, operations which involve data modification are slower.

When forced writes are enabled an ✕ appears in the box labeled “Enable Forced Writes” in the Database Properties dialog. To disable forced writes, click the check box to remove the ✕.

Validation and repair

In day-to-day operation, a database is sometimes subjected to events that pose minor problems to database structures. These events include:

- Abnormal termination of a database application. This does not affect the integrity of the database. When an application is canceled, committed data is preserved, and uncommitted changes are rolled back. If InterBase has already assigned a data page for the uncommitted changes, the page might be considered an *orphan page*. Orphan pages are unassigned disk space that should be returned to free space.
- Write errors in the operating system or hardware. These usually create a problem with database integrity. Write errors can result in "broken" or "lost" data structures, such as a database page or index. These corrupt data structures can make committed data unrecoverable.

You should validate a database:

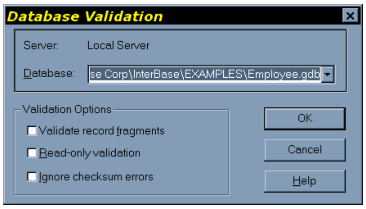
- Whenever a database backup is unsuccessful.
- Whenever an application receives a "corrupt database" error.
- Periodically, to monitor for corrupt data structures or misallocated space.
- Any time you suspect data corruption.

Note Database validation requires exclusive access to the database. Shut down a database to acquire exclusive access. If you do not have exclusive access to the database, you get the error message:

```
OBJECT database_name IS IN USE
```

To validate a database, choose **Maintenance | Database Validation** from the Main Menu, toolbar, or popup menu. The following dialog opens:

FIGURE 7.2 Database validation dialog



Server: Local Server
Database: se Corp\InterBase\EXAMPLES\Employee.gdb
Validation Options
☐ Validate record fragments
☐ Read-only validation
☐ Ignore checksum errors

The name of the current database is displayed in the Database text field. Because there are some conditions such as a checksum error that make it impossible to connect to a database, it is not necessary to connect to the database before performing a validation. If Server Manager is not connected to the database, you can enter the desired database name in the Database text field or select it from the drop-down list by clicking the arrow to the right of the field.

When Server Manager validates a database it verifies the integrity of data structures. Specifically, it does the following:

- Report corrupt data structures
- Report misallocated data pages
- Return orphan pages to free space

Validation options

You can select three options with Database Validation:

- Validate record fragments
- Read-only validation
- Ignore checksum errors

By default, database validation reports and releases only page structures. When you select the Validate record fragments option, validation reports and releases record structures as well as page structures.

By default, validating a database updates it, if necessary. To prevent updating, select the Read-only validation option.

Handling checksum errors

A checksum is a page-by-page analysis of data to verify its integrity. A bad checksum means that a database page has been randomly overwritten (for example, due to a system crash).

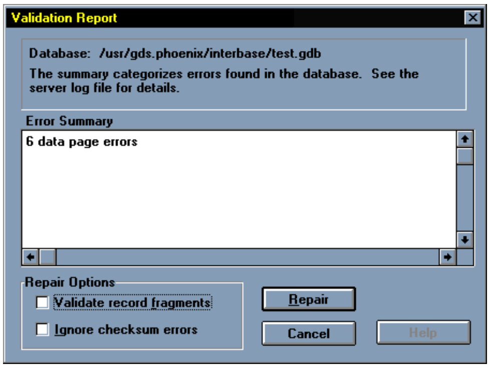
Checksum errors indicate data corruption. To repair a database that reports checksum errors, select the Ignore checksum errors option. This option enables Server Manager to ignore checksums when validating a database. Ignoring checksums allows successful validation of a corrupt database, but the affected data may be lost.

IMPORTANT Even if you can restore a mended database that reported checksum errors, the extent of data loss may be difficult to determine. If this is a concern, you may want to locate an earlier backup copy and restore the database from it.

Repairing a corrupt database

If a database contains errors, the following dialog opens:

FIGURE 7.3 Validation report dialog



Database: /usr/gds.phoenix/interbase/test.gdb
The summary categorizes errors found in the database. See the server log file for details.

6 data page errors

The errors encountered are summarized in the Error Summary area. The repair options you selected in the Database Validation dialog are selected in this dialog also.

To repair the database, choose Repair. This fixes problems that cause records to be corrupt and mark corrupt structures. In subsequent operations (such as backing up), InterBase ignores the marked records.

Note Some corruptions are too serious for Server Manager to correct. These include corruptions to certain strategic structures, such as space allocation pages. In addition, Server Manager cannot fix certain checksum errors that are random by nature and not specifically associated with InterBase.

If you suspect you have a corrupt database, perform the following steps:

1. Make a copy of the database using an operating-system command. Do not use the InterBase Backup utility, because it cannot back up a database containing corrupt data.
2. Repair the copy database to mark corrupt structures. If Server Manager reports any checksum errors, validate and repair the database again, choosing the Ignore checksum errors option. It may be necessary to validate a database multiple times to correct all the errors.

3. Validate the database again, with the Read-only validation option selected. Note that free pages are no longer reported, and broken records are marked as damaged. Any records marked during repair are ignored when the database is backed up.
4. Back up the mended database with Server Manager. At this point, any damaged records are lost, since they were not included during the back up. For more information about database backup, see **Chapter 8: "Database Backup and Restore"**
5. Restore the database to rebuild indexes and other database structures. The restored database should now be free of corruption.

Verify that restoring the database fixed the problem by validating the restored database with the Read-only validation option.

Database shutdown and restart

Maintaining a database often involves shutting it down. Only the SYSDBA or the owner of a database (the user who created it) may shut it down. The user who shut down the database then has *exclusive access* to the database.

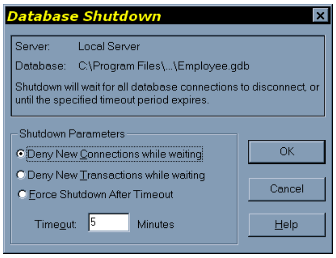
Exclusive access to a database is required to:

- Validate and repair the database.
- Add a foreign key to a table in the database or drop a foreign key from a table in the database.
- Add a secondary database file.

After a database is shut down, the database owner and SYSDBA is still able to connect to it, but any other user attempting to connect gets an error stating that the database is shut down.

To shut down the database, choose **Maintenance | Database Shutdown** from the Main Menu, toolbar, or popup menu. The Database Shutdown dialog appears:

FIGURE 7.4 Database shutdown dialog


Database Shutdown

Server: Local Server

Database: C:\Program Files\...\Employee.gdb

Shutdown will wait for all database connections to disconnect, or until the specified timeout period expires.

Shutdown Parameters

(●) Deny New Connections while waiting
( ) Deny New Transactions while waiting
( ) Force Shutdown After Timeout

Timeout: 5 Minutes

You can specify the following shutdown parameters:

- Timeout period in minutes. Server Manager attempts to shut down the database during this period. Exactly how it does this depends on the other shutdown parameters.
- Deny new connections while waiting. This option allows all existing database connections to complete their operations unaffected. Server Manager shuts down the database after all processes disconnect from the database. At the end of the timeout period, if there are still active connections, then the database is *not* shut down.
- Deny new transactions while waiting. This option allows existing transactions to run to normal completion. Once transaction processing is complete, Server Manager shuts down the database. Denying new transactions also denies new database connections. At the end of the timeout period, if there are still active transactions, then the database is *not* shut down.
- Force shutdown. With this option, there are no restrictions on database transactions or connections. As soon as there are no processes connections to the database, Server Manager shuts down the database. At the end of the timeout period, if there are still active connections, Server Manager rolls back any uncommitted transactions, disconnect any users, and shut down the database.

IMPORTANT Forcing database shutdown interferes with normal database operations, and should only be used after users have been given appropriate broadcast notification well in advance.

Denying new connections

The Deny New Connections shutdown option prevents any new processes from connecting to the database during the timeout period. This enables current users to complete their work, while preventing others from beginning new work.

Suppose the SYSDBA needs to shut down database *orders.gdb* at the end of the day (five hours from now) to perform routine maintenance. The Marketing department is currently using the database to generate important sales reports.

In this case, the SYSDBA would shut down *orders.gdb* with the following parameters:

- Deny New Connections.
- Timeout of 300 minutes (five hours).

These parameters specify to deny any new database connections and to shut down the database any time during the next five hours when there are no more active connections.

Any users who are already connected to the database are able to finish processing their sales reports, but new connections are denied. During the timeout period, the SYSDBA sends out periodic broadcast messages asking users to finish their work by 6 p.m.

When all users have disconnected, the database is shut down. If all users have not disconnected after five hours, then the database is not shut down. Because the shutdown is not critical, it is not forced.

It would be inappropriate to deny new transactions, since generating a report could require several transactions, and a user might be disconnected from the database before completing all necessary transactions. It would also be inappropriate to force shutdown, since it might cause users to lose work.

Denying new transactions

The Deny New Transactions option is the most restrictive shutdown option, since it prevents any new transactions from starting against the database. This option also prevents any new connections to the database.

Suppose the SYSDBA needs to perform critical operations that require shutdown of the database *orders.gdb*. This is a database used by dozens of customer service representatives throughout the day to enter new orders and query existing orders.

At 5 p.m., the SYSDBA initiates a database shutdown of *orders.gdb* with the following parameters:

- Deny New Transactions.
- Timeout of 60 minutes.

These parameters deny new transactions for the next hour. During that time, users can complete their current transactions before losing access to the database. Simply denying new connections would not be sufficient, since the shutdown cannot afford to wait for users to disconnect from the database.

During this hour, the SYSDBA sends out periodic broadcast messages warning users that shutdown will happen at 6 p.m, and instructing them to complete their work. When all transactions have been completed, the database is shut down.

After an hour, if there are still any active transactions, Server Manager cancels the shutdown. Since the SYSDBA needs to perform database maintenance, and has sent out numerous warnings that a shutdown will occur, there is no choice but to force a shutdown.

Forcing shutdown

If critical database maintenance requires a database to be shut down, and there are still active transactions, the SYSDBA can force shut down. This step should be taken only if broadcast messages have been sent out to users that shutdown is about to occur. If users have not heeded repeated warnings and remain active, then their work is rolled back.

The Force Shutdown option does not deny new transactions or connections during the timeout period. If, at any time during the timeout period, there are no connections to the database, Server Manager shuts down the database.

At the end of the timeout period, if there are still active connections, Server Manager rolls back any uncommitted transactions, disconnect any processes, and shut down the database.

Restarting a database

After a database is shut down, it must be restarted (brought back online) before users can access it.

To restart a database which has been shut down, choose **Maintenance | Database Restart** from the Main Menu, toolbar, or popup menu. The currently selected database is brought back online immediately.

Limbo transactions

When committing a transaction that spans multiple databases, InterBase automatically performs a two-phase commit. A *two-phase commit* guarantees that the transaction updates either all of the databases involved or none of them—data is never partially updated.

Note The Borland Database Engine (BDE), as of version 4.5, does not exercise the two-phase commit or distributed transactions capabilities of InterBase, therefore applications using the BDE never create limbo transactions.

DEFINITION OF LIMBO TRANSACTIONS

In the first phase of a two-phase commit, InterBase prepares each database for the commit by writing the changes from each *subtransaction* to the database. A subtransaction is the part of a multi-database transaction that involves only one database. In the second phase, InterBase marks each subtransaction as committed in the order that it was prepared.

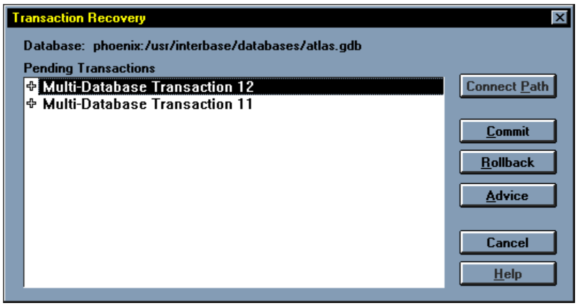
If a two-phase commit fails during the second phase, some subtransactions are committed and others are not. A two-phase commit can fail if a network interruption or disk crash makes one or more databases unavailable. Failure of a two-phase commit causes *limbo transactions*, transactions that the server does not know whether to commit or roll back.

It is possible that some records in a database are inaccessible due to their association with a transaction that is in a limbo state. To correct this, you must recover the transaction using Server Manager. *Recovering* a limbo transaction means committing it or rolling it back.

RECOVERING TRANSACTIONS

To recover limbo transactions, choose **Maintenance | Transaction Recovery** from the Main Menu, toolbar, or popup menu. A dialog displays a list of limbo transactions that can then be operated upon to recover—that is, to commit or roll back:

FIGURE 7.5 Transaction recovery dialog



Database: phoenix:/usr/interbase/databases/atlas.gdb

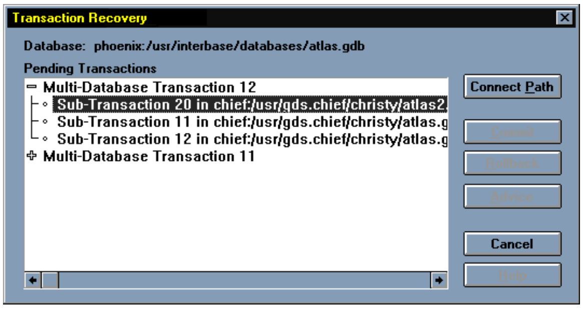
Pending Transactions

+ Multi-Database Transaction 12
+ Multi-Database Transaction 11

All the pending transactions in the database are listed in the scrolling area on the left side of the dialog. Click the "+" to display all the subtransactions of a transaction.

It is also possible to have a single database transaction that has been prepared and not committed. These transactions are displayed with a dingbat to the left of the transaction. You can roll back or commit such transactions.

FIGURE 7.6 Transaction recovery dialog



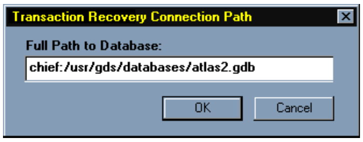
Database: phoenix:/usr/interbase/databases/atlas.gdb

Pending Transactions

− Multi-Database Transaction 12
  ∘ Sub-Transaction 20 in chief:/usr/gds.chief/christy/atlas2.
  ∘ Sub-Transaction 11 in chief:/usr/gds.chief/christy/atlas.g
  ∘ Sub-Transaction 12 in chief:/usr/gds.chief/christy/atlas.g
+ Multi-Database Transaction 11

You can change the path of the database specified by each subtransaction by choosing **Connect Path**. The following dialog appears:

FIGURE 7.7 Transaction recovery dialog


Transaction Recovery Connection Path

Full Path to Database:

chief:/usr/gds/databases/atlas2.gdb

Enter the directory path of the other database involved in the subtransaction, including the server name and separator indicating communication protocol, then choose OK.

The information on the path to the database was stored when the client application attempted the commit. It is possible that the path and network protocol from that machine does not work from the client which is now running Server Manager. Before attempting to roll back or commit any transaction, confirm the path of all involved databases is correct.

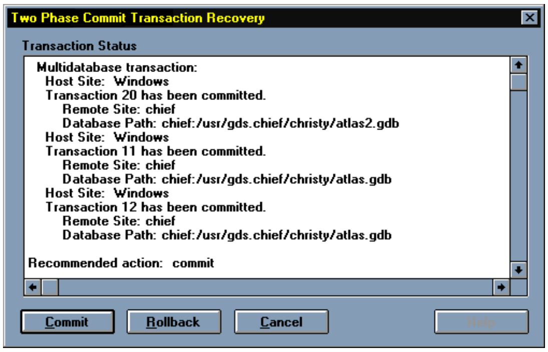
To use TCP/IP, separate the server and directory path with a colon (:). To use Named Pipes/NetBEUI, precede the server name with either a double backslash (\\) or a double slash (//), and then separate the server name and directory path with either a backslash or a slash.

You can choose to either commit or roll back each transaction. To commit or roll back, select the desired transaction ID from the list and choose either **Commit** or **Rollback**.

Note Only entire transactions can be recovered, so the commit and rollback buttons are enabled only when the main transaction is selected. They are disabled when a subtransaction is selected.

You can also seek advice by choosing the **Advice** button. This dialog opens:

FIGURE 7.8 Transaction recovery advice dialog



```
Multidatabase transaction:
  Host Site:  Windows
  Transaction 20 has been committed.
      Remote Site: chief
      Database Path: chief:/usr/gds.chief/christy/atlas2.gdb
  Host Site:  Windows
  Transaction 11 has been committed.
      Remote Site: chief
      Database Path: chief:/usr/gds.chief/christy/atlas.gdb
  Host Site:  Windows
  Transaction 12 has been committed.
      Remote Site: chief
      Database Path: chief:/usr/gds.chief/christy/atlas.gdb

Recommended action:  commit
```

This dialog displays information on each subtransaction: whether it has been committed, the remote server name, and database name. At the bottom, an action is recommended: either commit or roll back.

Server Manager analyzes the state of subtransactions by determining when the two-phase commit failed. If the first transactions are in limbo but later transactions are not, Server Manager assumes that the prepare phase did not complete. In this case, you are prompted to do a rollback.

gfix command-line tool

The **gfix** tool performs various maintenance activities on a database, most of which are duplicated in the Server Manager for Windows.

Syntax `gfix [options] db_name`

Options In the Option column of the following table, only the characters outside the brackets ([ ]) are required. You may specify any additional characters up to and including the full option name. To help identify options that perform similar functions, the Task column indicates the type of activity associated with an option.

Option	Task	Description
–at[tach] *n*	Shutdown	Used with *–shut* to prevent new database connections during timeout period of *n* seconds. Shutdown is canceled if there are still processes connected after *n* seconds.
–b[uffers] *n*	Cache buffers	Set default cache buffers for the database to *n* pages.
–ca[che] *n*		Reserved for future implementation.
–c[ommit] {*ID* | all}	Transaction recovery	Commit limbo transaction specified by *ID* or commit all limbo transactions.
–f[orce] *n*	Shutdown	Used with *–shut* to force shutdown of a database after *n* seconds. This is a drastic solution that should be used with caution.
–f[ull]	Data repair	Used with *–v* to check record and page structures, releasing unassigned record fragments.
–h[ousekeeping] *n*	Sweeping	Change automatic sweep threshold to *n* transactions, or disable sweeping by setting *n* to 0. Default threshold is 20,000 transactions. See **"Overview of sweeping" on page 138**. Exclusive access is not needed.
–i[gnore]	Data repair	Ignore checksum errors when validating or sweeping.
–l[ist]	Transaction recovery	Display IDs of each limbo transaction and indicate what would occur if *–t* were used for automated two-phase recovery.

TABLE 7.2 **gfix** options

Option	Task	Description
–m[end]	Data repair	Mark corrupt records as unavailable, so they are skipped (for example, during a subsequent backup).
–n[o_update]	Data repair	Used with –*v* to validate corrupt or misallocated structures. Structures are reported but not fixed.
–o[nline]	Shutdown	Cancels a –*shut* operation that is scheduled to take effect or rescinds a shutdown that is currently in effect.
–pa[ssword] *text*	Remote access	Check for password *text* before accessing a database.
–p[rompt]	Transaction recovery	Used with –*l* to prompt for action during transaction recovery.
–r[ollback] {*ID* | all}	Transaction recovery	Roll back limbo transaction specified by *ID* or roll back all limbo transactions.
–s[weep]	Sweeping	Force an immediate sweep of the database. Useful if automatic sweeping is disabled. Exclusive access is not necessary.
–sh[ut]	Shutdown	Shut down the database. Must also specify either –*attach*, –*force*, or –*tran*.
–t[wo_phase] {*ID* | all}	Transaction recovery	Perform automated two-phase recovery, either for a limbo transaction specified by *ID* or for all limbo transactions.

TABLE 7.2 **gfix** options (*continued*)

Option	Task	Description
–tr[an] *n*	Shutdown	Used with –*shut* to prevent new transactions from starting during timeout period of *n* seconds. Shutdown is canceled if there are still active transactions after *n* seconds.
–user *name*	Remote access	Check for user *name* before accessing a remote database.
–v[alidate]	Data repair	Locate and release pages that are allocated but unassigned to any data structures. Also reports corrupt structures.
–w[rite] {sync | async}	Database writes	Enable or disable forced (synchronous) writes. *sync* enables forced writes. *async* enables buffered writes.
–z		Show version of **gfix** and of InterBase engine.

TABLE 7.2 **gfix** options (*continued*)

gfix can be used to shut down the database, make minor repairs to data structures,

gfix error messages

Error Message	Causes and Suggested Actions to Take
Database file name <*string*> already given	A command-line option was interpreted as a database file because the option was not preceded by a hyphen (-) or slash (/). Correct the syntax.
Invalid switch	A command-line option was not recognized.
Incompatible switch combinations	You specified at least two options that do not work together, or you specified an option that has no meaning without another option (for example, *-full* by itself).
More limbo transactions than fit. Try again.	The database contains more limbo transactions than **gfix** can print in a single session. Commit or roll back some of the limbo transactions, then try again.

TABLE 7.3 **gfix** database maintenance error messages

Error Message	**Causes and Suggested Actions to Take**
Numeric value required	The *-housekeeping* option requires a single, non-negative argument specifying number of transactions per sweep.
Please retry, specifying <*string*>	Both a file name and at least one option must be specified.
Transaction number or “all” required	You specified *-commit*, *-rollback*, or *-two_phase* without supplying the required argument.

TABLE 7.3 **gfix** database maintenance error messages (*continued*)

CHAPTER

8

Database Backup and Restore

A database *backup* saves a database to a file on a hard disk or other storage medium. To protect a database from power failure, disk crashes, or other potential data loss, you should regularly back up the database. For additional safety, it is recommended to store the backup medium in a different physical location from the database server.

A database *restore* re-creates a database from a backup file.

Benefits of backup and restore

Operating systems usually include facilities to archive database files. Using the InterBase backup and restore feature in **gbak** or Server Manager offers several advantages over such backup methods. The backup and restore process accomplishes the following:

- Improves database performance, by performing garbage collection on outdated records, and balances indexes.
- Reclaims space occupied by deleted records, and pack the remaining data. This often reduces database size.
- Gives you the option of changing the database page size or distributing the database among multiple files or disks.

- Enables backups to run concurrently while other users are using the database. You do not have to shut down the database to run a back up. However, any data changes that clients commit to the database after the backup begins are not recorded in the backup file.
- Provides you with a platform-independent, stable snapshot of the database for archiving purposes.
- Creates a database backup to a disk file that you can include it as part of a regular system backup. Optionally, you can write the backup file directly to a named tape device.

IMPORTANT Do not restore a database that replaces one that is currently in use.

Data can be transferred to another operating system. Different platforms have device-dependent database file formats and therefore databases cannot simply be copied to a platform with a different operating system. You can make a backup in a generic format called a *transportable* backup that allows restoration to a server running a different operating system. Making transportable backups is highly recommended in heterogeneous environments.

- Upgrades ODS

New major releases of the InterBase server often contain changes to the on-disk structure (ODS). If the ODS has changed, and you want to take advantage of any new InterBase features, upgrade your databases to the new ODS.

You need not upgrade databases to use a new version of InterBase. The new versions can still access databases created with a previous version, but cannot take advantage of any new InterBase features.

To upgrade existing databases to a new ODS, perform the following steps:

1. Before installing the new version of InterBase, back up databases using the old version.
2. Install the new version of the InterBase server as described in **Chapter 3, "Installation."**
3. Once the new version is installed, restore the databases with the new version of InterBase.

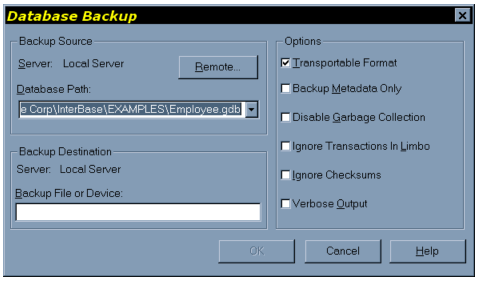
The restored databases are able to use any new InterBase server features.

Backing up a database

The database being backed up is called the *source.* The file or device to which the database is being backed up is called the *destination* or *target.*

1. To back up a database, choose **Tasks | Backup** from the Main Menu, toolbar, or popup menu. Server Manager displays the Database Backup dialog:

FIGURE 8.1 Database backup dialog



Database Backup

Backup Source
Server: Local Server
Database Path: e Corp\InterBase\EXAMPLES\Employee.gdb

Backup Destination
Server: Local Server
Backup File or Device:

Options
- [✓] Transportable Format
- [ ] Backup Metadata Only
- [ ] Disable Garbage Collection
- [ ] Ignore Transactions In Limbo
- [ ] Ignore Checksums
- [ ] Verbose Output

2. Type the name of the source database, including path, in the Database Path field in the Backup Source area. If a database spans multiple files, specify only the first file (the *primary* file) as the source. Server Manager uses the header page of each file to locate additional files, so the entire database can be backed up based on the primary file.

   By default, Server Manager displays the database to which Server Manager is currently connected. Click the drop-down button to the right of the text field to display a drop-down list of other databases previously connected to. To choose another database, select it from the list or type it in the text field.

3. Type the name of the destination file or device in the Backup File or Device field. Database files and backup files can have any name that is legal on the operating system; the *.gdb* and *.gbk* file extensions are InterBase conventions only.

   When creating a backup file, Server Manager writes the data out as one file. Multifile databases are never partially backed up. If a database spans multiple files, Server Manager backs up either all the files or none. You cannot split a large database among multiple backup files with Server Manager. However, you can back up to multiple files using the command-line **gbak** utility. See **"gsplit command-line tool"** on page 177 for information on using this tool.

4. Select the desired backup options, then choose OK to start the backup timer.

   Server Manager opens a standard text display window to display status and any messages during the backup process.

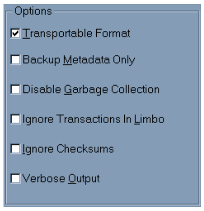
A backup file typically occupies less space than the database because it includes only the current version of data and incurs less overhead for data storage. A backup file also does not contain index data structures, only the index definition.

If you specify a backup file that already exists, Server Manager overwrites it. To avoid overwriting, specify a unique name for the backup file.

Backup options

The backup options are indicated by check boxes on the right side of the Database Backup dialog. If a check box has a ✔ inside, then the option is selected. If the box is empty, the option is not selected.

FIGURE 8.2 Database backup options



Options

- [✓] Transportable Format
- [ ] Backup Metadata Only
- [ ] Disable Garbage Collection
- [ ] Ignore Transactions In Limbo
- [ ] Ignore Checksums
- [ ] Verbose Output

Transportable format

To move a database to a machine with a different operating system from the machine on which the backup was performed, check the Transportable Format option. This option writes data in a generic format, enabling you to restore to any machine that supports InterBase.

To transfer a database to a server running a different operating system:

1. Select the Transportable Format option in the Database Backup dialog.
2. Back up the database.
3. If you backed up to a removable medium, proceed to Step 4. If you created a backup file on disk, use operating-system commands to copy the file to tape, then load the contents of the tape onto another machine. Or copy it across a network to the other machine.

4. On the destination machine, restore the backup file. If restoring from a removable medium, such as tape, specify the device name instead of the backup file.

Back up metadata only

When backing up a database, you can exclude its data, saving only its metadata. You might want to do this to:

- Retain a record of the metadata before it is modified.
- Create an empty copy of the database. The copy has the same metadata but can be populated with different data.

To back up metadata only, select the Back Up Metadata Only option.

TIP You can also extract a database's metadata using InterBase Windows ISQL. ISQL produces an SQL data definition (text) file containing SQL commands. Server Manager creates a backup file (binary) containing metadata only.

This function corresponds to the *-metadata* option of **gbak**.

Disable garbage collection

By default, Server Manager performs garbage collection during backup. To prevent garbage collection during a backup, select the Disable Garbage Collection option.

Garbage collection marks space used by old versions of data records as free for reuse. Generally, you want Server Manager to perform garbage collection during backup.

TIP You do not want to perform garbage collection if there is data corruption in old record versions and you want to prevent InterBase from visiting those records during a backup.

This function corresponds to the *-garbage_collect* option of **gbak**.

Ignore transactions in limbo

To ignore limbo transactions during backup, select the Ignore Transactions in Limbo option.

When Server Manager ignores limbo transactions during backup, it ignores all record versions created by any limbo transaction, finds the most recently committed version of a record, and backs up that version.

Limbo transactions are usually caused by the failure of a two-phase commit. They can also exist due to system failure or when a single-database transaction is prepared.

Before backing up a database that contains limbo transactions, it is a good idea to perform transaction recovery, by choosing **Maintenance | Transaction Recovery** in the Database Maintenance window.

This function corresponds to the *-limbo* option of **gbak**.

Ignore checksums

To ignore checksums during backup, select the Ignore Checksums option.

A checksum is a page-by-page analysis of data to verify its integrity. A bad checksum means that a data page has been randomly overwritten; for example, due to a system crash.

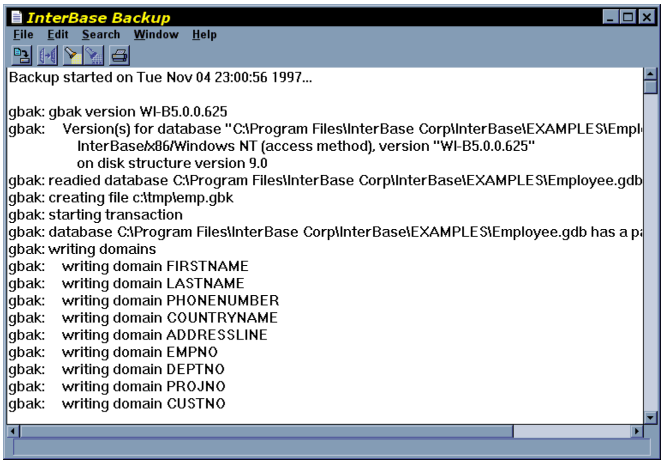
Checksum errors indicate data corruption, and InterBase normally prevents you from backing up a database if bad checksums are detected. Examine the data the next time you restore the database.

This function corresponds to the *-ignore* option of **gbak**.

Verbose output

To monitor the backup process as it runs, select the Verbose Output option. This option opens a standard text display window to display status messages on the screen. For example:

FIGURE 8.3 Database backup verbose output



```
Backup started on Tue Nov 04 23:00:56 1997...

gbak: gbak version WI-B5.0.0.625
gbak:    Version(s) for database "C:\Program Files\InterBase Corp\InterBase\EXAMPLES\Empl
            InterBase/x86/Windows NT (access method), version "WI-B5.0.0.625"
            on disk structure version 9.0
gbak: readied database C:\Program Files\InterBase Corp\InterBase\EXAMPLES\Employee.gdb
gbak: creating file c:\tmp\emp.gbk
gbak: starting transaction
gbak: database C:\Program Files\InterBase Corp\InterBase\EXAMPLES\Employee.gdb has a p
gbak: writing domains
gbak:    writing domain FIRSTNAME
gbak:    writing domain LASTNAME
gbak:    writing domain PHONENUMBER
gbak:    writing domain COUNTRYNAME
gbak:    writing domain ADDRESSLINE
gbak:    writing domain EMPNO
gbak:    writing domain DEPTNO
gbak:    writing domain PROJNO
gbak:    writing domain CUSTNO
```

By default, the backup window displays the time that the backup process starts, the time it ends, and any error messages.

The standard text display window enables you to search for specific text, save the text to a file, and print the text.

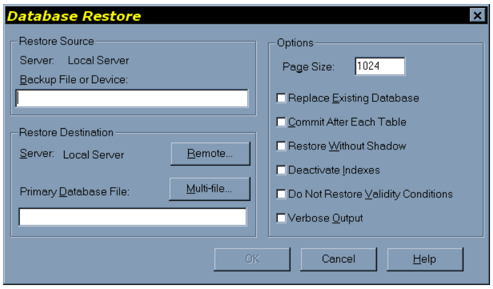
This function corresponds to the *-verbose* option of **gbak**.

Restoring a database

In restore operations, the backup file from which the database is being restored is called the *source*. The database being restored is called the *destination* or *target*.

1. To restore a database, choose **Tasks | Restor**e from the bar menu or toolbar. Server Manager displays the Database Restore dialog:

FIGURE 8.4 Database Restore dialog



2. Type the name of the source file or device on the current server in the Backup File or Device text field.

   To restore a database to more than one database file, click the Multi-file... button. For more information about restoring to multiple database files, see **"Restoring to multiple files" on page 166**.

3. Type the name (including drive letter and directory path) of the database to restore to in the Primary Database File field. **Note:** You cannot restore a database to a networked filesystem (mapped drive).
4. Type the page size, in bytes, in the Page Size field.
5. Select the desired restore options, and choose OK to begin the restore.

Typically, a restored database occupies less disk space than it did before being backed up, but disk space requirements could change if the on-disk structure version changes. For information about the ODS, see **"Benefits of backup and restore" on page 159**.

InterBase 5 restore allows you to restore a database successfully even if for some reason the restore process could not rebuild indexes for the database. For example, this can occur if there is not enough temporary disk space to perform the sorting task necessary to build an index during the restore. If this occurs, the database is restored and available, but indexes are inactive. This is as if you had selected the Deactivate Indexes checkbox, or used the **-i** switch of **gbak**. After the restore completes, use ALTER INDEX to set the indexes active.

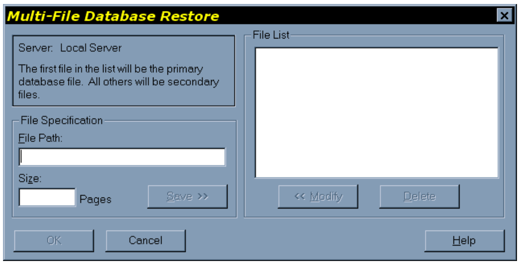
Database ownership In InterBase 5.5 and later, the restore operation preserves the original ownership of the database and its objects. Earlier versions assigned ownership to the login of the user performing the restore.

Restoring to multiple files

You might want to restore a database to multiple files to distribute it among different disks, which provides more flexibility in allocating system resources.

To restore a database to multiple database files, click the Multi-file... button in the Database Restore dialog to display the Multi-File Database Restore dialog:

FIGURE 8.5 Multifile database backup dialog


Multi-File Database Restore
Server: Local Server
The first file in the list will be the primary database file. All others will be secondary files.
File Specification
File Path:
Size:
Pages
Save >>
File List
<< Modify
Delete
OK
Cancel
Help

To specify multiple target database files, type the first file name in the File Path field and then type the number of pages for that file in the Size field. Choose Save, and the file name appears in the File List on the right side of the dialog. Then provide the same information for each successive target file, clicking Save for each one.

Note The minimum number of pages per file is 200.

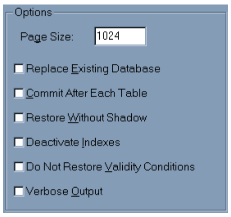
To modify one of the files in the list, select it and choose Modify. The selected file name appears in the File Path text field, where you can edit it, and the associated number of pages appears in the Pages text field. To delete a file, select it in the File List and choose the Delete button.

After entering all the names of the database files to restore to, choose OK to return to the Database Restore dialog.

Restore options

The restore options are shown in check boxes on the right side of the Database Restore dialog. If a check box has an ✔ inside, then the option is selected. If the box is empty, the option is not selected.

FIGURE 8.6 Database restore options



Options
Page Size: 1024
☐ Replace Existing Database
☐ Commit After Each Table
☐ Restore Without Shadow
☐ Deactivate Indexes
☐ Do Not Restore Validity Conditions
☐ Verbose Output

Page size

InterBase supports database page sizes of 1024, 2048, 4096, and 8192 bytes. The default is 1024 bytes. To change page size, back up the database and then restore it, modifying the Page Size field in the Database Restore dialog.

Changing the page size can improve performance for the following reasons:

- Storing and retrieving Blob data is most efficient when the entire Blob fits on a single database page. If an application stores many Blobs exceeding 1K, using a larger page size reduces the time for accessing Blob data.
- InterBase performs better if rows do not span pages. If a database contains long rows of data, consider increasing the page size.
- If a database has a large index, increasing the database page size reduces the number of levels in the index tree. Indexes work faster if their depth is kept to a minimum. Choose **Tasks | Database Statistics** to display index statistics, and consider increasing the page size if index depth is greater than two on any frequently used index.
- If most transactions involve only a few rows of data, a smaller page size may be appropriate, because less data needs to be passed back and forth and less memory is used by the disk cache.

This function corresponds to the *-page_size* option of **gbak**.

▸ *Replace existing database*

Server Manager cannot overwrite an existing database file unless the Replace Existing Database option is selected. If you attempt to restore to an existing database name and this option is not selected, the restore does not proceed.

IMPORTANT Do not replace an existing database while clients are operating on it. When restoring to an existing file name, a safer approach is to rename the existing database file, restore the database, then drop or archive the old database as needed.

This function corresponds to the -*replace* option of **gbak**.

▸ *Commit after each table*

Normally, Server Manager restores all metadata before restoring any data. If you select the Commit After Each Table option, Server Manager restores the metadata and data for each table together, committing one table at a time.

This option is useful when you are having trouble restoring a backup file; for example, if the data is corrupt or invalid according to integrity constraints.

If you have a problem backup file, restoring the database one table at a time lets you recover some of the data intact. You can restore only the tables that precede the bad data; restoration fails the moment it encounters bad data.

This function corresponds to the -*one_at_a_time* option of **gbak**.

▸ *Deactivate indexes*

Normally, InterBase rebuilds indexes when a database is restored. If the database contained duplicate values in a unique index when it was backed up, restoration fails. Duplicate values can be introduced into a database if indexes were temporarily made inactive (for example, to allow insertion of many records or to rebalance an index).

To enable restoration to succeed in this case, select the Deactivate Indexes option. This makes indexes inactive and prevents them from rebuilding. Then eliminate the duplicate index values, and re-activate indexes through ALTER INDEX in Windows ISQL.

A unique index cannot be activated using the ALTER INDEX statement; a unique index must be dropped and then created again. For more information about activating indexes, see the *Language Reference*.

TIP The Deactivate Indexes option is also useful for bringing a database online more quickly. Data access is slower until indexes are rebuilt, but the database is available. After the database is restored, users can access it while indexes are reactivated.

This function corresponds to the -*inactive* option of **gbak**.

Do not restore validity conditions

If you redefine validity constraints in a database where data is already entered, your data might no longer satisfy the validity constraints. You might not discover this until you try to restore the database, at which time an error message about invalid data appears.

IMPORTANT Always make a copy of metadata before redefining it; for example, by extracting it using Windows ISQL.

To restore a database that contains invalid data, select the Do Not Restore Validity Conditions option. This option deletes validity constraints from the metadata. After the database is restored, change the data to make it valid according to the new integrity constraints. Then add back the constraints that were deleted.

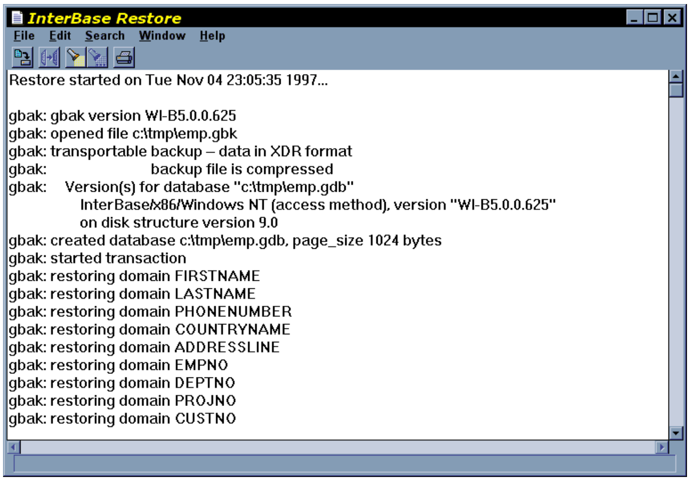
This option is also useful if you plan to redefine the validity conditions after restoring the database. If you do so, thoroughly test the data after redefining any validity constraints.

This function corresponds to the *-no_validity* option of **gbak**.

Verbose output

To monitor the restore process as it runs, select the Verbose Output option. This option opens a standard text display window to display status messages on the screen. For example:

FIGURE 8.7 Database restore verbose output



```
Restore started on Tue Nov 04 23:05:35 1997...

gbak: gbak version WI-B5.0.0.625
gbak: opened file c:\tmp\emp.gbk
gbak: transportable backup -- data in XDR format
gbak:                   backup file is compressed
gbak:    Version(s) for database "c:\tmp\emp.gdb"
             InterBase/x86/Windows NT (access method), version "WI-B5.0.0.625"
             on disk structure version 9.0
gbak: created database c:\tmp\emp.gdb, page_size 1024 bytes
gbak: started transaction
gbak: restoring domain FIRSTNAME
gbak: restoring domain LASTNAME
gbak: restoring domain PHONENUMBER
gbak: restoring domain COUNTRYNAME
gbak: restoring domain ADDRESSLINE
gbak: restoring domain EMPNO
gbak: restoring domain DEPTNO
gbak: restoring domain PROJNO
gbak: restoring domain CUSTNO
```

The standard text display window enables you to search for specific text, save the text to a file, and print the text. For an explanation of how to use the standard text display window, see **Chapter 1, "Introduction."**

This function corresponds to the *-verbose* option of **gbak**.

gbak command-line tool

Backs up or restores a database, optionally changing database characteristics.

Syntax For backing up:

```
gbak [-B] [options] database target
```

For restoring:

```
gbak {-C|-R} [options] source database
```

For restoring to multiple files:

```
gbak {-C|-R} [options] source primary m secondary1 [n1 secondary2 [n2]]
```

Argument	Description
database	Name of a database to back up or restore.
source	Name of a storage device or backup file from which to restore. On UNIX, this can also be *stdin*, in which case **gbak** reads input from the standard input (usually a pipe).
target	Name of a storage device or backup file to which to back up. On UNIX, this can also be *stdout*, in which case **gbak** writes its output to the standard output (usually a pipe).
primary	Primary file when restoring to multiple database files.
m	Length of *primary* in database pages; minimum value is 200.
secondary1	First secondary file when restoring to multiple database files.
n1	Length of *secondary1*; if only one secondary file is supplied, *n1* is optional.
secondary2	Next secondary file; specify as many secondary files as needed.
n2	Length of *secondary2*; it is unnecessary to specify the length of the last secondary file.

TABLE 8.1 **gbak** arguments

Options In the Option column of the following tables, only the characters outside the square brackets ([ ]) are required.

Table 8.2 lists the options to gbak that are available for creating backups.

Option	Description
-b[ackup_database]	Back up database to file or device
-co[nvert]	Convert external files as internal tables
-e[xpand]	Do not create a compressed back up
-fa[ctor] *n*	Use blocking factor *n* for tape device
-g[arbage_collect]	Do not garbage collect during backup
-ig[nore]	Ignore checksums during backup
-l[imbo]	Ignore limbo transactions during backup
-m[etadata]	Back up metadata only, no data
-nt	Create the backup in non-transportable format
-ol[d_descriptions]	Back up metadata in old-style format
-pa[ssword] *text*	Check for password *text* before accessing a database
-role *name*	Connect as role *name*
-t[ransportable]	Create a transportable backup (default); see **"Transportable format" on page 162**
-u[ser] *name*	Check for user *name* before accessing remote database
-v[erbose]	Show what **gbak** is doing
-y [*file* | suppress_output]	Direct status messages to *file; file* must not already exist; *suppress_output* suppress output messages
-z	Show version of **gbak** and of InterBase engine

TABLE 8.2 **gbak** backup options

Table 8.3 lists **gbak** options that are available when restoring databases.

Option	Description
-c[reate_database]	Restore database to a new file
-bu[ffers]	Cache size for restored database
-i[nactive]	Make indexes inactive upon restore
-k[ill]	Do not create any shadows that were previously defined
-n[o_validity]	Delete validity constraints from restored metadata; allows restoration of data that would otherwise not meet validity constraints
-o[ne_at_a_time]	Restore one table at a time; useful for partial recovery if database contains corrupt data
-p[age_size] *n*	Reset page size to *n* bytes (1024, 2048, 4196, or 8192); default is 1024.
-pa[ssword] *text*	Check for password *text* before accessing a database
-r[eplace_database]	Restore database to new file or replace existing file.
-user *name*	Check for user *name* before accessing remote database; required when using **gbak** from a PC client
-use_[all_space]	Restore database with 100% fill ratio on every data page, instead of the default 80% fill ratio
-v[erbose]	Show what **gbak** is doing
-y [*file* | suppress_output]	If used with -*v*, direct status messages to *file;* if used without -*v* and *file* is omitted, suppress output messages
-z	Show version of **gbak** and of InterBase engine

TABLE 8.3 **gbak** restore options

See Also **gsplit command-line tool**

gbak error messages

Error Message	Causes and Suggested Actions to Take
Array dimension for column <*string*> is invalid	Fix the array definition before backing up.
Bad attribute for RDB$CHARACTER_SETS	An incompatible character set is in use.
Bad attribute for RDB$COLLATIONS	Fix the attribute in the named system table.
Bad attribute for table constraint	Check integrity constraints; if restoring, consider using the -*no_validity* option to delete validity constraints.
Blocking factor parameter missing	Supply a numeric argument for "factor" option.
Cannot commit files	Database may contain corruption, or metadata may violate integrity constraints. Try restoring tables using -*one_at_a_time* option, or delete validity constraints using -*no_validity* option.
Cannot commit index <*string*>	Data may conflict with defined indexes. Try restoring using "inactive" option to prevent rebuilding indexes.
Cannot find column for Blob	
Cannot find table <*string*>	
Cannot open backup file <*string*>	Correct the file name you supplied and try again.
Cannot open status and error output file <*string*>	Messages are being redirected to invalid file name. Check format of file or access permissions on the directory of output file.
Commit failed on table <*string*>	Data corruption or violation of integrity constraint in the specified table. Check metadata or restore "one table at a time."
Conflicting switches for backup/restore	A backup-only option and restore-only option were used in the same operation. Fix the command and execute again.
Could not open file name <*string*>	Fix the file name and re-execute command.
Could not read from file <*string*>	Fix the file name and re-execute command.
Could not write to file <*string*>	Fix the file name and re-execute command.

TABLE 8.4 **gbak** backup and restore error messages

Error Message	**Causes and Suggested Actions to Take**
Datatype *n* not understood	An illegal datatype is being specified.
Database format *n* is too old to restore to	The **gbak** version used is incompatible with the InterBase version of the database. It may be necessary to back up the database using the *-expand* or *-old* options before it can be restored.
Database <*string*> already exists. To replace it, use the –R switch	You used *-create* in restoring a back up file, but the target database already exists. Either rename the target database or use *-replace*.
Could not drop database <*string*> (database might be in use).	You used *-replace* in restoring a file to an existing database, but the database is in use. Either rename the target database or wait until it is not in use.
Device type not specified	The *-device* option (Apollo only) must be followed by *ct* or *mt*. Obsolete as of InterBase V3.3.
Device type <*string*> not known	The *-device* option (Apollo only) was used incorrectly. Obsolete as of InterBase V3.3.
Do not recognize record type *n*	
Do not recognize <*string*> attribute *n* -- continuing	
Do not understand BLOB INFO item *n*	
Error accessing BLOB column <*string*> -- continuing	
ERROR: Backup incomplete	The backup cannot be written to the target device or file system. Either there is insufficient space, a hardware write problem, or data corruption.
Error committing metadata for table <*string*>	A table within the database may be corrupt. If restoring a database, try using *-one_at_a_time* to isolate the table.
Exiting before completion due to errors	This message accompanies other error messages and indicates that back up or restore could not execute. Check other error messages for the cause.
Expected array dimension *n* but instead found *m*	The problem array may need to be redefined.
Expected array version number *n* but instead found *m*	The problem array may need to be redefined.

TABLE 8.4 **gbak** backup and restore error messages (*continued*)

Error Message	Causes and Suggested Actions to Take
Expected backup database <*string*>, found <*string*>	Check the name of the backup file being restored.
Expected backup description record	
Expected backup start time <*string*>, found <*string*>	
Expected backup version 1, 2, or 3. Found *n*	
Expected blocking factor, encountered <*string*>	The *-factor* option requires a numeric argument.
Expected data attribute	
Expected database description record	
Expected number of bytes to be skipped, encountered <*string*>	
Expected page size, encountered <*string*>	The *-page_size* option requires a numeric argument.
Expected record length	
Expected volume number *n*, found volume *n*	When backing up or restoring with multiple tapes, be sure to specify the correct volume number.
Expected XDR record length	
Failed in put_blr_gen_id	
Failed in store_blr_gen_id	
Failed to create database <*string*>	The target database specified is invalid. It may already exist.
column <*string*> used in index <*string*> seems to have vanished	An index references a non-existent column. Check either the index definition or column definition.
Found unknown switch	An unrecognized **gbak** option was specified.
Index <*string*> omitted because *n* of the expected *m* keys were found	
Input and output have the same name. Disallowed.	A backup file and database must have unique names. Correct the names and try again.

TABLE 8.4 **gbak** backup and restore error messages (*continued*)

Error Message	**Causes and Suggested Actions to Take**
Length given for initial file (*n*) is less than minimum (*m*)	In restoring a database into multiple files, the primary file was not allocated sufficient space. InterBase automatically increases the page length to the minimum value. No action necessary.
Missing parameter for the number of bytes to be skipped	
Multiple sources or destinations specified	Only one device name can be specified as a source or target.
No table name for data	The database contains data that is unassigned to a table. Use **gfix** to validate or mend the database.
Page size is allowed only on restore or create	The *-page_size* option was used during a back up instead of a restore.
Page size parameter missing	The *-page_size* option requires a numeric argument.
Page size specified (*n* bytes) rounded up to *m* bytes	Invalid page sizes are rounded up to 1024, 2048, 4096, or 8192, whichever is closest.
Page size specified (*n*) greater than limit (8192 bytes)	Specify a page size of 1024, 2048, 4096, or 8192.
Password parameter missing	The back up or restore is accessing a remote machine. Use *-password* and specify a password.
Protection is not there yet	Unimplemented option *-unprotected* used.
Redirect location for output is not specified	You specified an option reserved for future use by InterBase.
REPLACE specified, but the first file <*string*> is a database	Check that the file name following the *-replace* option is a backup file rather than a database.
Requires both input and output file names	Specify both a source and target when backing up or restoring.
RESTORE: decompression length error	Possible incompatibility in the **gbak** version used for backing up and the **gbak** version used for restoring. Check whether *-expand* should be specified during back up.
Restore failed for record in table <*string*>	Possible data corruption in the named table.
Skipped *n* bytes after reading a bad attribute *n*	

TABLE 8.4 **gbak** backup and restore error messages (*continued*)

Error Message	Causes and Suggested Actions to Take
Skipped *n* bytes looking for next valid attribute, encountered attribute *m*	
Trigger <*string*> is invalid	
Unexpected end of file on backup file	Restoration of the backup file failed. The backup procedure that created the backup file may have terminated abnormally. If possible, create a new backup file and use it to restore the database.
Unexpected I/O error while <*string*> backup file	A disk error or other hardware error may have occurred during a backup or restore.
Unknown switch <*string*>	An unrecognized **gbak** option was specified.
User name parameter missing	The backup or restore is accessing a remote machine. Supply a user name with the *–user* option.
Validation error on column in table <*string*>	The database cannot be restored because it contains data that violates integrity constraints. It may be necessary to delete constraints from the metadata by specifying *–no_validity* during restore.
Warning -- record could not be restored	Possible corruption of the named data.
Wrong length record, expected *n* encountered *n*	

TABLE 8.4 **gbak** backup and restore error messages (*continued*)

gsplit command-line tool

Syntax

```
gsplit -split filename [size {k|m|g}][filename [size {k|m|g}]...]

gsplit -join filename [filename ...]
```

Options **gsplit** is a utility that works as a filter to write the output of **gbak** to multiple files. You can use this to backup files larger than the operating system's limit for file size. You can use **gsplit** to split the backup file created by **gbak** into multiple files. You can also use **gsplit** to restore a database from several files.

Specify the size of each file that gsplit writes to with a filename, followed by one of *k*, *m*, or *g*. These letters correspond to kilobytes, megabytes, and gigabytes respectively.

To back up a database using **gbak** and **gsplit**, direct output from **gbak** to *stdout* and pipe it to **gsplit**. **gsplit** takes input only from *stdin*. The value of *size* must be at least 1 MB.

To restore a multifile backup, direct output from **gsplit** to a pipe to **gbak** reading input from *stdin*.

Example To limit the first backup file, *f1.gbk* to 20 Megabytes:

```
gbak -backup some.gdb stdout | gsplit -split f1.gbk 20m f2.gbk 20m
```

To rejoin the backup files for a restore:

```
gsplit -join f1.gbk f2.gbk | gbak -restore stdin database.gdb
```

See Also **gbak**

CHAPTER

9

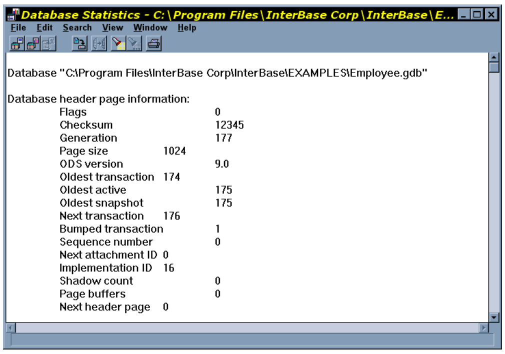
Database and Server Statistics

This chapter provides an overview of the facilities in InterBase for collecting statistics on database files, lock requests, and client database operation requests.

Viewing statistics using Server Manager for Windows

To view database statistics, select the desired database in the server/database tree and then choose **Tasks | Database Statistics** from the Server Manager menu. This opens the Database Statistics window:

FIGURE 9.1 Database statistics output


Database Statistics - C:\Program Files\InterBase Corp\InterBase\E...
File Edit Search View Window Help

```
Database "C:\Program Files\InterBase Corp\InterBase\EXAMPLES\Employee.gdb"

Database header page information:
        Flags                   0
        Checksum                12345
        Generation              177
        Page size       1024
        ODS version             9.0
        Oldest transaction  174
        Oldest active           175
        Oldest snapshot         175
        Next transaction    176
        Bumped transaction      1
        Sequence number         0
        Next attachment ID  0
        Implementation ID   16
        Shadow count            0
        Page buffers            0
        Next header page    0
```

In addition to the features of the standard text display window, the Database Statistics window contains a View menu, enabling you to view:

- Database Summary
- Database Analysis

These operations display database statistics from the header page, such as the oldest transaction and number of allocated pages. The database pages can also be analyzed to determine the number of pages and the fill distribution of pages within each table.

Database summary

To view information from the database header page, choose **View | Database Summary**. This section shows example database summary header page information, followed by an explanation of each item.

This function corresponds to the *-log* option of **gstat**.

Header page information

The first line displays the name and location of the primary database file.

```
Database 'C:\Program Files\InterBase Corp\
    InterBase\examples\employee.gdb'
```

The next lines contain information on the database header page.

```
Database header page information:
 Flags 0
 Checksum 15351
```

```
Generation 174
Page size 1024
ODS version 9.1
Oldest transaction 22
Oldest active 166
Next transaction 170
Bumped transaction 1
Sequence number 0
Next attachment ID 0
Implementation ID 2
Shadow count 0
Number of cache buffers 0
Next header page 0
Creation date May 11, 1994 10:21:46
```

These items are:

- Checksum: the header page checksum. This is a unique value computed from all the data in the header page. When the header page is stored to disk and later retrieved, the checksum of the retrieved page is recomputed and compared to the stored value to ensure that the information is correct.
- Generation: counter incremented each time header page is written.
- Page size: the current database page size, in bytes.
- ODS version: the version of the database's on-disk structure.
- Oldest transaction: the transaction ID number of the oldest "interesting" transaction (those that are not committed, but active, in limbo, or rolled back).
- Oldest active: the transaction ID number of the oldest active transaction.
- Next transaction: the transaction ID number that InterBase assigns to the next transaction.

  Note The difference between the oldest transaction and the next transaction determines when database sweeping occurs. For example, if the difference is greater than this difference (set to 20,000 by default), then InterBase initiates a database sweep. See **"Overview of sweeping" on page 138**.
- Sequence number: the sequence number of the header page (zero is used for the first page, one for second page, and so on).
- Next connection ID: ID number of the next database connection.

- Implementation ID: architecture of the system on which the database was created:

1	Apollo	9	Reserved
2	Sun, HP 9000, IMP Delta, NeXT	10	RS 6000
3	Reserved	11	Data General AViiON
4	VMS	12	HP MPE/XL
5	VAX Ultrix	13	Silicon Graphics Iris
6	Reserved	14	Reserved
7	HP 900	15	DEC OSF/1
8	OS/2, Windows NT, Novell NetWare		

- Shadow count: the number of shadow files defined for the database.
- Number of cache buffers: the number of page buffers in the database cache.
- Next header page: the ID of the next header page.
- Creation date: the date when the database was created.
- Attributes:
  - force write—indicates that forced database writes are enabled.
  - no_reserve—indicates that space is not reserved on each page for old generations of data. This enables data to be packed more closely on each page and therefore makes the database occupy less disk space.
  - shutdown—indicates database is shut down.

Database file sequence

This section lists all the files associated with the database, including any secondary files. For example:

```
Database file sequence:
File /usr/interbase/examples/employee.gdb is the only file
```

Database analysis

Choosing **View | Database Analysis** displays general information on the current database in the text display area, including all the database header page information displayed by **View | Database Summary** plus table and index information. Since this command analyzes all the tables and indexes in a database, it can take a long time for large databases.

This section shows example table and index information, followed by an explanation of each item.

This function corresponds to the *-all* option of **gstat**.

Table and index information

The information on each table and index in the database is displayed. For example:

```
COUNTRY (31)
 Primary pointer page: 246, Index root page: 247
 Data pages: 1, data page slots: 1, average fill: 59%
 Fill distribution:
 0 - 19% = 0
 20 - 39% = 0
 40 - 59% = 1
 60 - 79% = 0
 80 - 99% = 0
```

- Primary pointer page: the page that is the first pointer page for the table.
- Index root page: the page number that is the first pointer page for indexes.
- Data pages: the total number of data pages.
- Data page slots: the number of pointers to database pages, whether the pages are still in the database or not.
- Average fill: the average percentage to which the data pages are filled.
- Fill distribution: a histogram that shows the number of data pages that are filled to the percentages.

For each index in the database, the following information is displayed.

```
Index RDB$PRIMARY1 (0)
 Depth: 1, leaf buckets: 1, nodes: 14
 Average data length: 6, total dup: 0, max dup: 0
 Fill distribution:
 0 - 19% = 1
```

```
20 - 39% = 0
40 - 59% = 0
60 - 79% = 0
80 - 99% = 0
```

- Index: the name of the index.
- Depth: the number of levels in the index page tree.
- Leaf buckets: the number of leaf (bottom level) pages in the index page tree.
- Nodes: the total number of index pages in the tree.
- Average data length: the average length of each key, in bytes.
- Total dup: the total number of rows that have duplicate indexes.
- Max dup: the number of duplicates of the index with the most duplicates.
- Fill distribution: a histogram that shows the number of index pages filled to the specified percentages.

TIP If the depth of the index page tree is greater than three, then sorting may not be as efficient as possible. To reduce the depth of the index page tree, increase the page size. If increasing the page size does not reduce the depth, then return it to its previous size.

gstat command-line tool

Syntax `gstat [options] database`

Description The **gstat** program is a command-line tool for retrieving and reporting database statistics. Its function is the same as that described for Server Manager in **Chapter 9: "Database and Server Statistics"** of this book.

Note You can run **gstat** only against local databases: run **gstat** on the server host.

Options The valid options for **gstat** are described in the following table.

Option	Description
`-header`	Stop reporting statistics after reporting the information on the header page
`-log`	Stop reporting statistics after reporting the information on the log pages
`-index`	Retrieve and display statistics on indexes in the database
`-data`	Retrieve and display statistics on data tables in the database
`-all`	Equivalent to supplying *-index* and *-data*; this is the default if you supply none of *-index*, *-data*, or *-all*
`-system`	Retrieve statistics on system tables and indexes in addition to user tables and indexes
`-z`	Print product version of **gstat**

TABLE 9.1 **gstat** options

Viewing lock statistics in Server Manager for Windows

Locking is a mechanism that InterBase uses to maintain the consistency of the database when it is accessed by multiple users. The lock manager is a thread in the **ibserver** process that coordinates locking.

The lock manager uses a lock table to coordinate resource sharing among client threads in the **ibserver** process connected to the database. The lock table contains information on all the locks in the system and their states. The global header information contains useful aggregate information such as the size of the lock table, the number of free locks, the number of deadlocks, and so on. There is also process information which includes whether the lock has been granted or is waiting, and so on. This information is useful when trying to correct deadlock situations.

To view lock manager statistics, choose **Tasks | Lock Manager Statistics**. This opens a standard text display window and automatically display lock manager statistics for the current server.

iblockpr command-line tool

Syntax

```
iblockpr [a,o,w]
iblockpr [-i{a,o,w}] [t n]
```

Description You can use **iblockpr** to monitor performance by checking lock requests. This is a command-line tools that reports the same output as **Tasks | Lock Manager Statistics** in Server Manager for Windows.

Note The program is called gds_lock_print on UNIX.

The first form of syntax given above is to retrieve a report of lock statistics at one instant in time. The second form is to monitor performance by collecting samples at fixed intervals.

These options display interactive information on current activity in the lock table. *t* specifies the time in seconds between samplings. *n* (count) specifies the number of samples to be taken. The utility prints out the events per second for each sampling and gives the average of the values in each column at the end.

Options

Option	Description
`[none]`	Same as *-o*
`-a`	Prints a static view of the contents of the lock table
`-o`	Prints a static lock table summary and a list of all entities that own blocks
`-w`	Prints out all the information provided by the *-o* flag plus wait statistics for each owner; this option helps to discover which owner's request is blocking others in the lock table

TABLE 9.2 **iblockpr/gds_lock_print** options

Option	Description
	The following options supply interactive statistics (events/second) for the requested items, which are sampled *n* times every *t* seconds, with one line printed for each sample. The average of the sample values is printed at the end of each column. If you do not supply values for *n* and *t*, the default is *n*=1.
`-i`	Prints all statistics; the output is easier to read if you issue *-ia*, *-io*, and *-iw* separatly.
`-ia`	Prints how many threads are trying to acquire access to the lock table per second
`-io`	Prints operation statistics such lock requests, conversions, downgrades, and releases per second
`-iw`	Prints number of lock acquisitions and requests waiting per second, wait percent, and retries

TABLE 9.2 **iblockpr/gds_lock_print** options

Example The following statement prints "acquire" statistics (access to lock table: *acquire/s, acqwait/s, %acqwait, acqrtry/s,* and *rtrysuc/s*) every 3 seconds until 10 samples have been taken:

```
gds_lock_print –ia 3 10
```

Retrieving statistics programmatically

InterBase 5 includes programming facilities to gather performance timings and database operation statistics.

You can use the API function **isc_database_info( )** to retrieve statistics, by specifying one or more of the following request buffer items:

Request Buffer Item	Result Buffer Contents
isc_info_fetches	Number of reads from the memory buffer cache. Calculated since the InterBase server started.
isc_info_marks	Number of writes to the memory buffer cache. Calculated since the InterBase server started.
isc_info_reads	Number of page reads. Calculated since the InterBase server started.
isc_info_writes	Number of page writes. Calculated since the InterBase server started.
isc_info_backout_count	Number of removals of record versions per table since the current database attachment started.
isc_info_delete_count	Number of row deletions. Reported per table. Calculated since the current database attachment started.
isc_info_expunge_count	Number of removals of a record and all of its ancestors, for records whose deletions have been committed. Reported per table. Calculated since the current database attachment started.
isc_info_insert_count	Number of inserts into the database. Reported per table. Calculated since the current database attachment started.
isc_info_purge_count	Number of removals of old versions of fully mature records (records committed, resulting in older-ancestor-versions no longer being needed). Reported per table. Calculated since the current database attachment started.
isc_info_read_idx_count	Number of reads done via an index. Reported per table. Calculated since the current database attachment started.

TABLE 9.3 Database I/O statistics information items

Request Buffer Item	Result Buffer Contents
`isc_info_read_seq_count`	Number of sequential database reads, that it, the number of sequential table scans (row reads). Reported per table. Calculated since the current database attachment started.
`isc_info_read_update_count`	Number of row updates. Reported per table. Calculated since the current database attachment started.

TABLE 9.3 Database I/O statistics information items

See the *API Guide* for information on request buffers, and details of using this API call.

CHAPTER

10 Interactive Query

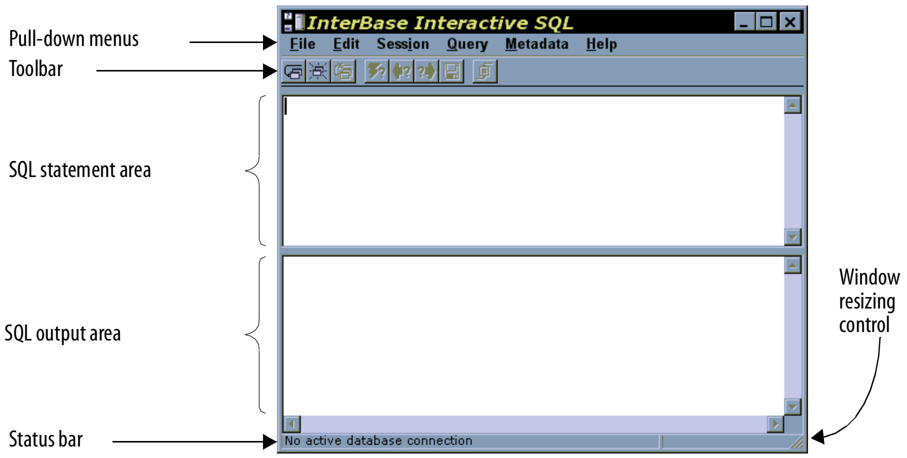
This chapter documents the Windows ISQL and command-line **isql** utilities for InterBase. These tools provide an interface to the Dynamic SQL interpreter of InterBase. You can use the query tools to perform data definition, prototype queries before implementing them in your application, or perform ad hoc examination of data in your database.

Windows ISQL

Starting Windows ISQL

To start Windows ISQL, double-click the Windows ISQL icon. The ISQL window opens:

InterBase Windows ISQL main window



The ISQL window can also be opened from the Server Manager by choosing **Tasks | Interactive SQL** or clicking the corresponding ToolBar button. Windows ISQL is connected to Server Manager's current database (if any).

The ISQL window

The Interactive SQL window consists of a menu bar with pull-down menus, a toolbar, the SQL Statement area, the ISQL Output area, and a status bar at the bottom of the window.

ISQL Menus

FILE MENU

Contains commands to connect to, create, drop, and disconnect from a database, execute an SQL script file, save results and the session to a file, commit and roll back work, and exit Windows ISQL.

EDIT MENU

Standard Windows options: Undo, Copy, Paste, Select All. Also, a Clear Output item to clear the SQL Output area.

Note Undo in the Edit menu does *not* undo database changes. Use **File | Rollback** to abort database changes.

SESSION MENU

Contains statements to set basic and advanced ISQL settings and display ISQL settings and version.

QUERY MENU

Contains statements to execute the current statement in the SQL statement area, and to browse forward and backward through the history of statements.

METADATA MENU

Contains commands to view and extract definitions for the database, tables, views, and other metadata objects.

HELP MENU

Provides online help.

ISQL work areas

THE SQL STATEMENT AREA

The SQL statement area is where you type an SQL statement to be executed. It scrolls vertically.

THE ISQL OUTPUT AREA

The ISQL output area is where the results of the SQL statements are displayed. It scrolls both vertically and horizontally.

ISQL toolbar

Button	Description
	Connect See "Connecting to an existing database" on page 195.
	Create database See "Creating a new database" on page 197.
	Disconnect See "Disconnecting from a database" on page 198.
	Execute query See "Executing SQL statements" on page 198.
	Previous query See "Executing SQL statements" on page 198.
	Next query See "Executing SQL statements" on page 198.
	Save results See "Saving results to a file" on page 201.
	Connect version See "Executing SQL statements" on page 198.

TABLE 10.1 Buttons on the InterBase Windows ISQL toolbar

Status bar

The status bar at the bottom of the ISQL window shows the name of the database to which Windows ISQL is connected or “No active database connection” if it is not connected to a database. It also shows the name of the remote server on which the database resides, or “Local Server” if the database is local.

To issue statements in Windows ISQL, you must either create a new database or connect to an existing database.

Exiting Windows ISQL

To exit Windows ISQL, choose **File | Exit**. This closes the connection to the current database (if any) and exit Windows ISQL. Any uncommitted changes to the database are rolled back.

Temporary files

Windows ISQL creates temporary files used during a session to store information such as the command history, output file names, and so on. These files are named in the form *isql_aa.xx*. The files are stored in the directory specified by the TMP environment variable, or if that is not defined, the working directory, or if that is not defined, then the *windows* directory.

To avoid cluttering the *windows* directory with temporary files, specify a directory in which to store them by defining TMP.

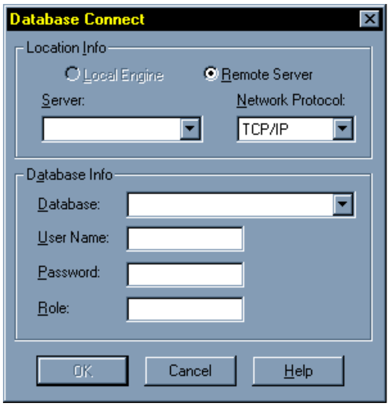
When you exit, Windows ISQL deletes these temporary files. If Windows ISQL abnormally terminates (for example, due to a power failure), then these files remain and may be freely deleted without any adverse effects. You should not delete any of these temporary files while Windows ISQL is running, because they may be used in the current session.

Connecting to an existing database

Choose **File | Connect to Database** to connect to an existing database. If Windows ISQL is currently connected to a database, it closes the connection, prompting you to commit changes to it if there are any. If you choose No, then all database changes since the last commit are rolled back and the connection is closed. If you choose Yes, then database changes are committed and the connection is closed.

The Database Connect dialog opens:

FIGURE 10.2 Windows ISQL Database Connect dialog



The Database Connection dialog enables you to connect to databases on a local or remote server. If a Local InterBase Server is running on your PC, the Local Engine button is enabled.

To connect to a local database server, choose Local Server and provide the name of the database (including full volume and directory path), a valid username, and password. You can select a database from the drop-down list of previously used databases or use the Browse button to select other databases.

To connect to a remote database server, choose Remote Server and select a network protocol from the drop-down list. Information about network protocols is provided in Help.You must provide the name of the database, including full volume and directory path, and a valid username and password. You can select a database from the drop-down list of previously used databases.

To connect as a role, specify the name of the role in the Role field. Connecting as a role gives you all privileges that have been assigned to that role, assuming that you have previously been granted that role with the GRANT statement.

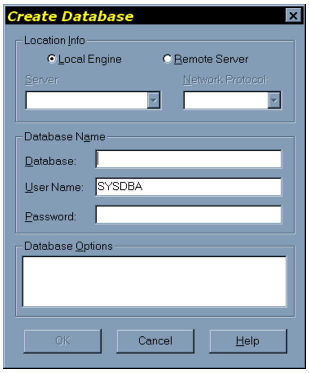
Note For access to InterBase databases on a server, a valid user name and password is required. See **Chapter 6, "Database Security."**

Creating a new database

To create a new database and connect to it, choose **File | Create Database**. If there is an existing connection to a database, a dialog prompts you to commit any changes to it. If you choose No, then all database changes since the last commit are rolled back. If you choose Yes, then database changes are committed.

The Create Database dialog opens:

FIGURE 10.3 Windows ISQL create database dialog



User Name: SYSDBA

The Create Database dialog enables you to create and connect to databases on a local or remote server.

To create a local database, choose Local Server and provide the name of the database, including full volume and directory path, a valid username, and a password.

To create a remote database, choose Remote Server and select a network protocol from the drop-down list. Information about network protocols is provided in the online Help. You must provide the name of the database (including full volume and directory path) and a valid username and password.

In the Database Options area, enter any additional options of the CREATE DATABASE statement, such as PAGE_SIZE, DEFAULT CHARACTER SET, or secondary files. For a complete list of CREATE DATABASE options, see the *Language Reference*. To create a basic database without any options, leave the Database Options area blank.

Note Database files must reside on a local drive.

Choose OK to create the database. InterBase creates the database on the specified server and connect to the database.

For more information about creating databases, see the *Language Reference*.

Dropping a database

Dropping a database deletes the database to which you are currently connected, removing both data and metadata. To drop the current database, choose **File | Drop Database**. A dialog asks you to confirm that you want to delete the database. A database can be dropped only by its creator or the SYSDBA user.

A dropped database is removed from the list of databases maintained in *interbas.ini*.

IMPORTANT Dropping a database deletes all data and metadata in the database.

Disconnecting from a database

To disconnect from the database to which Windows ISQL is connected, choose **File | Disconnect from Database**. A dialog opens to confirm that you want to disconnect. If there are any uncommitted database changes, Windows ISQL prompts you to commit them before disconnecting.

Executing SQL statements

In Windows ISQL, you can execute SQL statements:

- Interactively, one statement at a time.
- From a file containing an SQL script.

Executing SQL interactively

To execute an SQL statement interactively, type it in the SQL Statement area and choose Run or press Ctrl-Enter. The Output Area echoes the statement and displays up to 32Kb of the results. You can scroll to view output beyond 32Kb.

TIP You can copy text from other Windows applications such as the Notepad and Wordpad text editors and paste it into the SQL Statement area. You can also copy statements from the ISQL Output area and paste them into the SQL Statement area. This cut-and-paste method is also a convenient way to use the online SQL tutorial provided in the online Help.

When an SQL statement is executed (whether successfully or not), it becomes part of the ISQL *command history,* a sequential list of SQL statements entered in the current session. The *current statement* is the statement displayed in the SQL Statement area.

The three buttons in the Toolbar pertaining to execution of SQL statements are:

Button	Description
⚡?	**Execute query** Executes the current statement. The resultant output is displayed in the ISQL Output area. This button is dimmed if there is no active database connection.
⇦?	**Previous query** Recalls the previous SQL statement in the command history, making it the current statement. When the current statement is the first statement in the command history, this button is grayed out
?⇨	**Next query** Recalls the next SQL statement in the command history, making it the current statement. When the current statement is the last statement in the command history, this button is grayed out.

TABLE 10.2 Toolbar buttons for executing SQL statements

As an alternative to these buttons, use the hot keys Ctrl-Enter, Ctrl-p, and Ctrl-n, respectively.

LEGAL STATEMENTS

- You can execute interactively any SQL statement identified as "available in DSQL" in the *Language Reference*. You cannot use any statements that are specifically identified as **isql** statements. All statements that are specific to **isql** have functionally equivalent menu items in Windows ISQL.

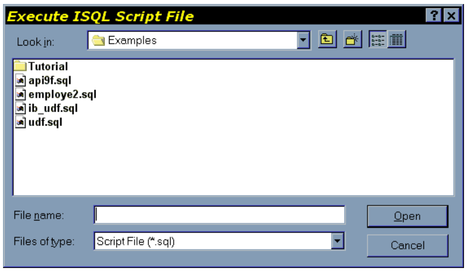
  For example, the SET NAMES statement cannot be entered in the SQL Statement area. To change the active character set, choose **Session | Advanced Settings** and select the desired character set in the Advanced Set Options dialog.

- SQL script files can include statements that are not legal to enter interactively. For example, you can use the SET statements such as SET LIST or SET TERM in scripts.
- Transaction names may not be used with SET TRANSACTION statement.
- Each statement entered in the SQL statement area can optionally be terminated by a semicolon (**;**), but this is not required. The SQL statement area accepts only one statement at a time, even if they are separated by the semicolon. It is an error to enter two statements in the SQL statement area. If you do this and try to execute the two statements together, *neither* statement executes.

Executing an ISQL script file

To execute a file containing SQL statements, choose **File | Run an ISQL Script**. The following dialog appears in Windows 95 (Windows NT uses a different file dialog):

FIGURE 10.4 Execute ISQL script dialog



Choose the file containing the script to run. If you have made uncommitted changes to the database, Windows ISQL prompts you to commit or roll back the work. Then, a dialog appears asking, "Save Output to a File?" If you choose Yes, then another dialog appears, enabling you to specify an output file. If you choose No, then the results are displayed in the ISQL Output area. If you choose Cancel, then the operation is canceled.

After Windows ISQL finishes executing a script file, a summary dialog appears, indicating if there were any errors. If there were errors, then an error message appears in the ISQL Output Area (or output file) after each statement that caused the error.

Every **isql** script file must begin with either a CREATE DATABASE statement or a CONNECT statement (including username and password) to specify the database on which the script file operates. For more information about these SQL statements, see the *Language Reference*.

Statements executed in a script file do not become part of the command history.

Committing and rolling back work

Changes to the database from data definition (DDL) statements—for example, CREATE and ALTER statements—are automatically committed by default. To turn off automatic commit of DDL, choose **Session | Basic Settings** and click off the Auto Commit DDL check box.

Changes made to the database by data manipulation (DML) statements—for example INSERT and UPDATE—are not permanent until they are committed. Commit changes by choosing **File | Commit Work**.

To undo all database changes from DML statements since the last commit, choose **File | Rollback Work**.

Saving results to a file

Windows ISQL enables you to save to a file:

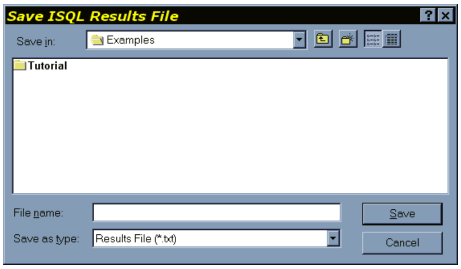
- The output of the last SQL statement executed.
- SQL statements entered in the current session.

Saving isql output

To save to a file the results of the last SQL statement executed, choose **File | Save Result to File** or click the Save Result button in the ISQL window.

The following dialog appears in Windows 95 (Windows NT uses a different dialog):

FIGURE 10.5 Save ISQL results dialog



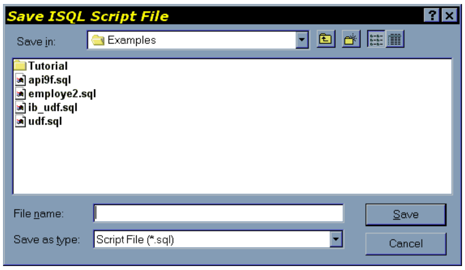
Choose the desired directory and file name or type the file name in the text field, and choose OK. The output from the last successful statement and the statement itself is saved to the named text file.

If you run an SQL script, and then choose **File | Save Result to File**, then all the commands in the script file and their results are saved to the output file. If command display has been turned off in a script with SET ECHO OFF, then SQL statements in the script are not saved to the file.

Saving the session

To save the SQL statements entered in the current session to a text file, choose **File | Save Session to a File**. The following dialog appears:

FIGURE 10.6 Save ISQL script dialog



Choose the desired directory and file name or type the file name, and click OK to save the SQL statements to the file.

Only the SQL statements entered in the current session, not the output, are saved to the specified file.

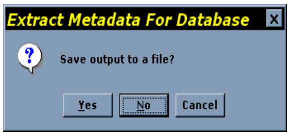
Extracting metadata

Windows ISQL enables you to extract metadata for the entire database and for a specific table or view.

Extracting database metadata

To extract data definition statements (metadata) from a database to a file, choose **Metadata | Extract Database**. The following dialog opens:

FIGURE 10.7 Extract database dialog



If you choose Yes, then another dialog opens, enabling you to enter the name of the file to which to extract the metadata. If you choose No, then the metadata is displayed to the ISQL Output area only. If you choose Cancel, then the operation is canceled.

Extract Database extracts metadata in a specific order, to allow the resulting script to be used as input to recreate the database.

8.

Metadata	Comments
1. Database	Extracts database with default character set and PAGE_SIZE
2. Domains	Must be before tables that reference domains
3. Tables	
4. Indexes	Must be after tables
5. FOREIGN KEY constraints	Must be added after tables to avoid tables referenced before being created
6. Views	Must be after tables
7. CHECK constraints	Must be after tables
8. Exceptions	Must be extracted before stored procedures and triggers that contain code to raise exceptions
9. Stored procedures	Stored procedures are shown with no body in CREATE PROCEDURE and then ALTER PROCEDURE to add the text of the procedure body; this is to allow circular or recursive procedure references
10. Triggers	Must be after tables Must be after stored procedures, to allow trigger code to reference procedures Does not extract triggers from CHECK constraints.
11. Roles	Must be before GRANT privileges
12. GRANTs	Must be after tables, views, stored procedures, triggers, and roles

TABLE 10.3 Order of metadata extraction

The Extract Database command does not extract:

- Code of external functions or filters, because that code is not part of the database. The declarations to the database (with DECLARE EXTERNAL FUNCTION and DECLARE FILTER) are extracted.
- System tables, system views, and system triggers.

- Because DDL statements do not contain references to object ownership, the extracted file does not show ownership. The output file includes the name of the object and the owner if one is defined. There is no way to assign an object to its original owner.

Extracting table metadata

To extract metadata for a single table, choose **Metadata | Extract Table**. The following dialog opens:

FIGURE 10.8 Extract table dialog

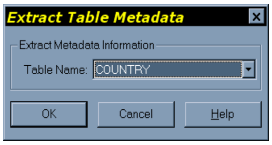


Table Name: COUNTRY

Click the arrow to the right of the Table Name field to see a drop-down list of tables in the database. Select a table, then choose OK to extract metadata from that table.

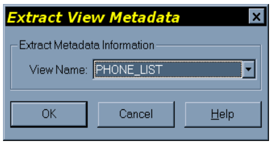
Another dialog opens, asking whether to save output to a file. Choose Yes to save the metadata to a text file, No to display the metadata to the Output area only, or Cancel to cancel the operation.

If there are no tables in the database, then the menu item is grayed out, and you cannot select it.

Extracting view metadata

To extract metadata for a single view, choose **Metadata | Extract View**. The following dialog opens:

FIGURE 10.9 Extract view dialog



View Name: PHONE_LIST

Click the arrow to the right of the View Name field to see a drop-down list of views in the database. Choose a view, then choose OK to extract metadata from that view.

Another dialog opens, asking whether to save output to a file. Choose Yes to save the metadata to a text file, No to display the metadata to the Output area only, or Cancel to cancel the operation.

If there are no views defined for the database, then the menu item is grayed out, and you cannot select it.

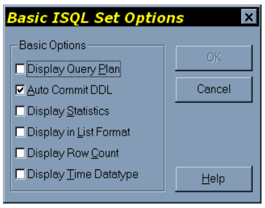
Changing Windows ISQL settings

The Session menu enables you to change **isql** settings for the current session and display information about the database and its metadata.

Basic isql settings

Choose **Session | Basic Settings** to open a dialog displaying all the basic settings that can be toggled on or off:

FIGURE 10.10 Basic settings dialog


Basic ISQL Set Options

Basic Options

- [ ] Display Query Plan
- [✓] Auto Commit DDL
- [ ] Display Statistics
- [ ] Display in List Format
- [ ] Display Row Count
- [ ] Display Time Datatype

Each setting has a corresponding check box. If there is a ✔ in the box, then the setting is on. Otherwise, it is off. Click the check box or the setting name to toggle the setting.

The following table summarizes basic **isql** settings:

Setting	Behavior when setting is ON
Display Query Plan	Display the query plan chosen by the optimizer when a SELECT is entered. To modify the optimizer plan, use the PLAN option of the SQL SELECT statement. See "SET PLAN" on page 232.
Auto Commit DDL	Automatically commits DDL (data definition) statements as each statement is entered. This setting is ON by default. If this setting is off, you must explicitly commit DDL statements (with **File | Commit Work**) to make them permanent. See "SET AUTODDL" on page 225.

TABLE 10.4 Basic **isql** settings

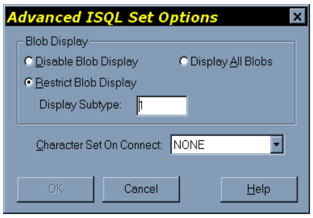
Setting	Behavior when setting is ON
Display Statistics	Displays performance statistics for each statement entered. The following performance statistics appear after the result of each statement: • Number of read or write requests • Number of requests for data or updates that can be serviced in cache • Elapsed time • CPU time • Memory usage • Database page size • Database buffers used See “SET STATS” on page 233.
Display in List Format	Displays data in list format, with headings on the left and column values on the right, one row at a time. If this setting is off (default), data is displayed in tabular format with data in rows and columns, which may wrap longer rows. See “SET LIST” on page 230.
Display Row Count	Displays the number of rows returned by each SELECT query entered. See “SET COUNT” on page 228.
Display Time Datatype	Displays the time portion of DATE values. If this setting is OFF, then only the date portion of DATE values is displayed. See “SET TIME” on page 236.

TABLE 10.4 Basic **isql** settings (*continued*)

Advanced isql settings

There are two advanced **isql** settings: Blob display and character set choice. Choose **Session | Advanced Settings** to open the following dialog:

FIGURE 10.11 Advanced settings dialog


Advanced ISQL Set Options

Blob Display
- ○ Disable Blob Display
- ○ Display All Blobs
- ● Restrict Blob Display

Display Subtype: 1

Character Set On Connect: NONE

Choose OK to accept all the setting changes, or choose Cancel to cancel setting changes.

BLOB DISPLAY

The upper area in the dialog enables you to determine the display of Blob datatypes.

This setting determines the display of Blob data. SELECT always displays the Blob ID for columns of Blob datatype. By default, a SELECT also displays actual Blob data of text subtypes beneath the associated row.

The choices are:

- Disable Blob Display: Do not display contents of Blob columns.
- Display ALL Blobs: Display Blob data of all subtypes.
- Restrict Blob Display: Display contents of Blob columns only for the specified subtype. Use 0 for an unknown subtype; 1 for a text subtype (the default), and other integers for other Blob subtypes.

See "SET BLOBDISPLAY" on page 226.

CHARACTER SET

This setting determines the active character set for strings for subsequent connections to the database. It enables you to override the default character set for a database.

Specify the character set before connecting to the database whose character set you want to specify. For a complete list of character sets recognized by InterBase, see the *Language Reference*.

Choice of character set limits possible collation orders to a subset of all available collation orders. Given a character set, a collation order can be specified when data is selected, inserted, or updated in a column.

You can perform the same function in an SQL script with the SET NAMES command. Use SET NAMES before connecting to the database whose character set you want to specify. **See "SET NAMES" on page 231.**

▸ *Displaying settings*

To display all the current **isql** settings, choose **Session | Display Settings**. The basic settings and selected advanced settings is displayed in the ISQL Output area.

See "SET" on page 223.

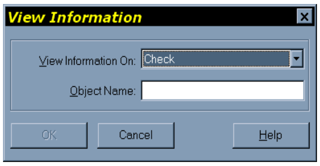
Displaying version information

To display version information, choose **Session | Display Connect Version** in the ISQL Output Area. This displays the version of Windows ISQL being used. If connected to a database, this command also displays the versions of the InterBase access method, server, and remote interface.

Displaying metadata information

Choose **Metadata | Show** to display database information and metadata. The following dialog opens:

FIGURE 10.12 Show metadata dialog



Select the object type for which to display information, supply any required information in the Object Name text field, and choose OK. Generally, if you do not supply an Object Name, then Windows ISQL displays the names of all objects of the selected type in the database. If you do supply an Object Name, then Windows ISQL displays information about that object.

The following table lists the items that you can access through the View Information dialog.

Item	DIsplays
Check	Check constraints for the table specified in the Object Name field
Database	File name, page size and allocation, and sweep interval for the current database; do not specify an Object Name
Domain	With no Object Name: Names of all domains in the current database With Object Name: Name and datatype of the specified domain

TABLE 10.5 Metadata information items

Item	DIsplays
Exception	With no Object Name: Names of all exceptions in the current database, their associated messages, and the names of triggers and stored procedures that use them With Object Name: Name and message of specified exception and names of triggers and stored procedures that use it
Filter	Names of all Blob filters defined for the current database
Function	Names of all UDFs defined for the current database
Generator	With no Object Name: Names and current values of all generators in the current database With Object Name: Name and current value of the generator given as Object Name
Grant	Permissions for the table or view given as Object Name
Index	With no Object Name: Names of all indexes in the current database, their constituent columns, and uniqueness With table Object Name: Names of all indexes for the table, their constituent columns, and uniqueness With index Object Name: Constituent columns for the index given as Object Name, and the index's uniqueness
Procedure	With no Object Name: Names and dependencies of all stored procedures in the current database With Object Name: Procedure body, input parameters, and output parameters for the specified procedure
Role	Displays roles defined in the current database
System	Displays the names of system tables and system views for the current database
Table	With no Object Name: Names of all tables in the current database With Object Name: Columns, datatypes, PRIMARY KEY, FOREIGN KEY, and CHECK constraints for the specified table
Trigger	With no Object Name: Names of all triggers in the current database and the tables for which they are defined With table Object Name: Bodies of all triggers defined on that table With trigger Object Name: Body of the specified trigger
View	With no Object Name: Names of all views in the current database With Object Name: Columns, datatypes, and view source for the specified view

TABLE 10.5 Metadata information items (*continued*)

Command-line isql tool

Command-line **isql** is a utility for processing SQL data definition (DDL) and data manipulation (DML) statements from interactive input or from a source file. It enables you to create and view metadata, add and modify data, grant user permissions, test queries, and perform database administration tasks.

This section provides an introduction to using **isql**. For a description of the standard SQL commands available in **isql**, see the *Language Reference*. For a description of special **isql** commands, see **"isql command reference" on page 217**.

Invoking isql

You can use **isql** in the following ways:

- Interactively to process SQL statements, by entering statements at the **isql** prompt
- Non-interactively to process SQL statements in a file

To start the isql utility, type the following at a UNIX shell prompt or Windows console prompt:

```
isql [options] [database_name]
```

where *options* are command-line options and *database_name* is the name of the database to connect to, including disk and directory path.

If no options are specified, **isql** starts an interactive session. If no database is specified, you must connect to an existing database or create a new one. If a database was specified, **isql** starts the interactive session by connecting to the named database.

If options are specified, **isql** starts interactively or non-interactively, depending on the options. For example, reading an input file and writing to an output file are non-interactive tasks, so the *-input* or *-output* options do not start an interactive session. Additional noninteractive options include *-a*, *-database*, *-extract*, and *-x*, which are used when extracting DDL statements.

When you start an interactive **isql** session, the following prompt appears:

```
SQL>
```

You must then end each command with a terminator character. The default terminator is a semicolon (`;`). You can change the terminator to any character or group of characters with the SET TERMINATOR command or with the *-terminator* command-line option. If you omit the terminator, a continuation prompt appears (`CON>`).

Note For clarity, all of the commands and examples in this chapter end with the default semicolon terminator.

Command-line options

Only the initial characters in an option are required. You can also type any portion of the text enclosed in brackets, including the full option name. For example, specifying *-n*, *-no*, or *-noauto* has the same effect.

Option	**Description**
-a	Extracts all DDL for the named database.
-d[atabase] *name*	Used with *-x*. Changes the CREATE DATABASE statement that is extracted to a file. Without *-d*, CREATE DATABASE appears as a C-style comment and uses the database name specified on the **isql** command line. With *-d*, **isql** extracts an uncommented CREATE DATABASE and substitutes *name* as its database argument.
-c[ache]	Set number of cache buffers for this connection to the database. See **"Default cache size per ISQL connection" on page 141**.
-e[cho]	Displays (echoes) each statement before executing it.
-ex[tract]	Same as *-x*.
-i[nput] *file*	Reads commands from an input file instead of from standard input. Input files can contain *-input* commands that call other files, enabling execution to branch and then return. **isql** exits (with a commit) when it reaches the end of the first file. In interactive sessions, use *-input* to read commands from a file.
-n[oauto]	Turns off automatic commitment of DDL statements. By default, DDL statements are committed automatically in a separate transaction.
-o[utput] *file*	Writes results to an output file instead of to standard output. In interactive sessions, use *-output* to write results to a file.
-pas[sword] *password*	Used with *-user*. Specifies a password when connecting to a remote server. For access, both *password* and *user* must represent a valid entry in the security database.
-page[length] *n*	Prints column headers every *n* lines instead of the default 20.

TABLE 10.6 **isql** command-line options

Option	Description
-q[uiet]	
-r[ole] *rolename*	Adopt role *rolename* on connection to the database.
-t[erminator] *x*	Change the end-of-statement symbol from the default semicolon (**;**) to *x*, where *x* is a single character or any sequence of characters.
-u[ser] *user*	Used with *-password*. Specifies a user name when connecting to a remote server. For access, both *password* and *user* must represent a valid entry in the security database.
-x	Extracts DDL for the named database. Displays DDL to the screen unless redirected to a file.
-z	Displays the software version of **isql**.

TABLE 10.6 **isql** command-line options (*continued*)

Examples

- Suppose *createdb.sql* contains DDL statements to create a database. To execute the statements, enter:

```
isql -input createdb.sql
```

- The following example starts an interactive connection to a remote database. The remote server, jupiter, accepts the specified user and password combination with the privileges assigned to the "staff" role:

```
isql -user sales -password mycode -role staff
jupiter:/usr/customer.gdb
```

- The next example starts an interactive session but does not attach to a database. **isql** commands are displayed, and query results print column headers every 30 lines:

```
isql -echo -page 30
```

Exiting isql

To exit **isql** and roll back all uncommitted work, enter:

```
QUIT;
```

To exit **isql** and commit all work, enter:

```
EXIT;
```

Connecting to a database

If you do not specify a database on the command-line when invoking **isql**, you must either connect to an existing database or create a new one. Use the CONNECT command to connect to a database and CREATE DATABASE to create a database. For the full syntax of CONNECT and CREATE DATABASE, see the *Language Reference*.

You can connect to a database in two ways. You can connect to:

- A local database on Windows NT or Windows 95. Use the CONNECT command with the full path of the database as the argument. For example:

  ```
  connect C:/InterBase Corp/InterBase/examples/employee.gdb role
  “staff”;
  ```

- A remote database on an Windows or UNIX server using TCP/IP. Use the CONNECT command with the full node name and path of the database as the argument. Separate the node name from the database path with a colon.

  To connect to a database on a Unix platform named jupiter:

  ```
  connect jupiter:/usr/interbase/examples/employee.gdb;
  ```

  To connect to a database on a Windows NT platform named venus:

  ```
  connect venus:c:/Program Files/InterBase Corp/
     InterBase/examples/employee.gdb;
  ```

  Note Be careful not to confuse node names and shared disks, since both are specified with a colon separator. If you specify a single letter that maps to a disk drive, it is assumed to be a drive, not a node name.

TIP You may use either forward slashes (“/”) or backslashes (“\”) as directory separators. InterBase automatically converts either type of slash to the appropriate type for the server operating system.

Transaction behavior in isql

When you start **isql**, InterBase begins a transaction. That transaction remains in effect until you issue a COMMIT or ROLLBACK statement. You must issue a COMMIT or ROLLBACK statement to end a transaction. Issuing one of these statements automatically starts a new transaction. You can also start a transaction with the SET TRANSACTION statement.

isql uses a separate transaction for DDL statements. When these statements are issued at the `SQL>` prompt, they are committed automatically as soon as they are completed. DDL scripts should issue a COMMIT after every CREATE statement to ensure that new database objects are available to all subsequent statements that depend on them. For more information on DDL statements, see the *Data Definition Guide*.

Extracting metadata

You can extract the DDL statements that define the metadata for a database to an output file with the *-extract* option. Adding the optional *-output* flag reroutes output to a named file. Use this syntax:

```
isql [[-extract | -x][-a] [[-output | -o] outputfile]] database;
```

The *-x* option is an abbreviation for *-extract*. The *-a* flag directs **isql** to extract all database objects. Note that the output file specification, *outputfile,* must follow the *-output* flag, while you can place the name of the database being extracted at the end of the command.

Option	Description
database	File specification of the database from which metadata is being extracted.
outputfile	File specification of the text file to receive the extracted statements. If omitted, **isql** writes the information to the screen.

TABLE 10.7 **isql** extracting metadata arguments

You can use the resulting text file to:

- Examine the current state of a database's system tables before you plan alterations to it, or when a database has changed significantly since its creation.
- Use your text editor to make changes to the database definition or create a new database source file.

The *-extract* option does not extract UDF code and Blob filters, because they are not part of the database. It *does* extract the declarations to the database (with DECLARE EXTERNAL FUNCTION and DECLARE FILTER).

The *-extract* option also does not extract system tables, system views, or system triggers.

Because DDL statements do not contain references to object ownership, the extracted file does not show ownership. The output file includes the name of the object and the owner if one is defined. There is no way to assign an object to its original owner.

For a list of the order of extraction of metadata objects, see **"Extracting database metadata" on page 202**.

For example, the following statement extracts the system catalogs from the database *employee.gdb* to a file called *employee.sql*:

```
isql -extract -output employee.sql employee.gdb;
```

The resulting output script is created with *-commit* following each set of commands, so that tables can be referenced in subsequent definitions. This command extracts all keywords and object names in uppercase when possible (some international metadata has no uppercase).

To extract DDL statements from database *employee.gdb* and store in the file *employee.sql*, enter:

```
isql -a employee.gdb -output employee.sql
```

The following example extracts the DDL statements from the database *dev.gdb*:

```
isql -x dev.gdb
```

This example combines the *-extract* and *-output* options to extract the DDL statements from the database *dev.gdb* into a file called *dev.out*. The output database name must follow the *-output* flag.

```
isql -extract -output dev.out dev.gdb
```

isql Commands

At the `SQL>` prompt, you can enter any of three kinds of commands:

- SQL data definition (DDL) statements, such as CREATE, ALTER, DROP, GRANT, and REVOKE. These statements create, modify, or remove metadata and objects, and control user access (via privileges) to the database. For more information about DDL, see the *Data Definition Guide*.
- SQL data manipulation (DML) statements such as SELECT, INSERT, UPDATE, and DELETE. These four data manipulation operations affect the data in a database. They retrieve, modify, add, or delete data. For more information about DML statements, see the *Language Reference*.
- **isql** commands that fall into three main categories:
  - SHOW commands (to display metadata or other database information)
  - SET commands (to modify the **isql** environment)

- Other commands (for example, commands to read an input file, write to an output file, or end an **isql** session)

Some **isql** commands have many options. **See "isql command reference" on page 217.**

SHOW *commands*

SHOW commands are used to display metadata, including tables, indexes, procedures, and triggers.

SHOW commands list all of the specified objects or give information about a particular object when used with *name*.

SHOW commands operate on a separate transaction from user statements. They run as READ COMMITTED background statements and acknowledge all metadata changes immediately.

SET commands

SET commands enable you to view and change the **isql** environment.

Other isql commands

The remaining **isql** commands perform a variety of useful tasks, including reading an SQL file, executing shell commands, and exiting **isql**. The other isql commands are: BLOBDUMP, EDIT, EXIT, HELP, INPUT, OUTPUT, QUIT, SHELL.

Exiting isql

To exit the **isql** utility and roll back all uncommitted work, enter:

```
QUIT;
```

To exit the **isql** utility and commit all work, enter:

```
EXIT;
```

Error handling

InterBase handles errors in **isql** and DSQL in the same way. To indicate the causes of an error, **isql** uses the SQLCODE variable and the InterBase status array.

The following table lists values that are returned to SQLCODE:

SQLCODE	Message	Meaning
< 0	SQLERROR	Error occurred. Statement did not execute.
0	SUCCESS	Successful execution.
+1–99	SQLWARNING	System warning or informational message.
+100	NOT FOUND	No qualifying rows found, or end of current active set of rows reached.

TABLE 10.8 SQLCODE and message summary

For a detailed discussion of error handling, see the *Programmer's Guide*. For a complete listing of SQLCODE and InterBase status array codes, see the *Language Reference*.

isql command reference

This chapter describes the syntax and usage for commands available only in InterBase **isql** (interactive SQL). These commands are also available in SQL scripts. For a description of the standard DSQL commands available in **isql**, see the *Language Reference*.

Command-line **isql** supports the following special commands:

BLOBDUMP	SET BLOBDISPLAY	SHELL	SHOW INDEX
EDIT	SET COUNT	SHOW CHECK	SHOW INDICES
EXIT	SET ECHO	SHOW DATABASE	SHOW PROCEDURES
HELP	SET LIST	SHOW DOMAINS	SHOW ROLES
INPUT	SET NAMES	SHOW EXCEPTIONS	SHOW SYSTEM
OUTPUT	SET PLAN	SHOW FILTERS	SHOW TABLES
QUIT	SET STATS	SHOW FUNCTIONS	SHOW TRIGGERS
SET	SET TERM	SHOW GENERATORS	SHOW VERSION
SET AUTODDL	SET TIME	SHOW GRANT	SHOW VIEWS

TABLE 10.9 **isql** commands

BLOBDUMP

Places the contents of a Blob column in a named file for reading or editing.

Syntax

```
BLOBDUMP blob_id filename;
```

Argument	Description
blob_id	System-assigned hexadecimal identifier, made up of two hexadecimal numbers separated by a colon (**:**). The first number is the ID of the table containing the Blob column. The second number is a sequential number identifying a particular instance of Blob data.
filename	Name of the file into which to place Blob contents.

Description BLOBDUMP stores Blob data identified by *blob_id* in the file specified by *filename*. Because binary files cannot be displayed, BLOBDUMP is useful for viewing or editing binary data. BLOBDUMP is also useful for saving blocks of text (Blob data) to a file.

To determine the *blob_id* to supply in the BLOBDUMP statement, issue any SELECT statement that selects a column of Blob data. When the table's columns appear, any Blob columns contain hexadecimal Blob IDs. The display of Blob output can be controlled using SET BLOBDISPLAY.

Example Suppose that Blob ID 58:c59 refers to graphical data in JPEG format. To place this Blob data into a graphics file named PICTURE.JPG, enter:

```
BLOBDUMP 58:c59 PICTURE.JPG;
```

See Also SET BLOBDISPLAY

EDIT

Allows editing and re-execution of **isql** commands.

Syntax

```
EDIT [filename];
```

Argument	Description
filename	Name of the file to edit.

Description The EDIT command enables you to edit commands in:

- A source file and then execute the commands upon exiting the editor.
- The current **isql** session, then re-execute them.

On Windows 95 and Windows NT, EDIT calls the text editor specified by the EDITOR environment variable. If this environment variable is not defined, then EDIT uses the Microsoft mep editor.

On UNIX, EDIT calls the text editor specified by either the VISUAL environment variable or EDITOR, in that order. If neither variable is defined, then EDIT uses the **vi** editor.

If given *filename* as an argument, EDIT places the contents of *filename* in an edit buffer. If no file name is given, EDIT places the commands in the current isql session in the edit buffer.

After exiting the editor, isql automatically executes the commands in the edit buffer.

Filenames with spaces You can optionally delimit the filename with double or single quotes. This allows you to use filenames with spaces in EDIT statements.

Examples To edit the commands in a file called *start.sql* and execute the commands when done, enter:

```
EDIT START.SQL;
```

In the next example, a user wants to enter `SELECT DISTINCT JOB_CODE, JOB_TITLE FROM JOB;` interactively: Instead, the user mistakenly omits the DISTINCT keyword. Issuing the `EDIT` command opens the statement in an editor and then executes the edited statement when the editor exits.

```
SELECT JOB_CODE, JOB_TITLE FROM JOB;
EDIT;
```

See Also INPUT, OUTPUT, SHELL

EXIT

Commits the current transaction, closes the database, and ends the **isql** session.

Syntax `EXIT;`

Description Both EXIT and QUIT close the database and end an **isql** session. EXIT commits any changes made since the last COMMIT or ROLLBACK, whereas QUIT rolls them back.

EXIT is equivalent to the end-of-file character, which differs across systems.

IMPORTANT EXIT commits changes without prompting for confirmation. Before using EXIT, be sure that no transactions need to be rolled back.

See Also QUIT, SET AUTODDL

HELP

Displays a list of ISQL commands and short descriptions.

Syntax

```
HELP;
```

Description HELP lists the built-in **isql** commands, with a brief description of each.

Example To save the HELP screen to a file named ISQLHELP.LST, enter:

```
OUTPUT ISQLHELP.LST;
HELP;
OUTPUT;
```

After issuing the HELP command, use OUTPUT to redirect output back to the screen.

INPUT

Read and execute commands from the named file.

Syntax

```
INPUT filename;
```

Argument	Description
filename	Name of the file containing SQL statements and SQL commands.

Description INPUT reads commands from *filename* and executes them as a block. In this way, INPUT enables execution of commands without prompting. *filename* must contain SQL statements or **isql** commands.

Input files can contain their own INPUT commands. Nesting INPUT commands enables **isql** to process multiple files. When **isql** reaches the end of one file, processing returns to the previous file until all commands are executed.

The INPUT command is intended for non-interactive use. Therefore, the EDIT command does not work in input files.

Using INPUT *filename* from within an **isql** session has the same effect as using *-input filename* from the command line.

Unless output is redirected using OUTPUT, any results returned by executing *filename* appear on the screen.

You can optionally delimit the filename with double or single quotes. This allows you to use filenames with spaces in INPUT statements.

Examples For this example, suppose that file ADD.LST contains the following INSERT statement:

```
INSERT INTO COUNTRY (COUNTRY, CURRENCY)
VALUES ('Mexico', 'Peso');
```

To execute the command stored in ADD.LST, enter:

```
INPUT ADD.LST;
```

For the next example, suppose that the file, TABLE.LST, contains the following SHOW commands:

```
SHOW TABLE COUNTRY;
SHOW TABLE CUSTOMER;
SHOW TABLE DEPARTMENT;
SHOW TABLE EMPLOYEE;
SHOW TABLE EMPLOYEE_PROJECT;
SHOW TABLE JOB;
```

To execute these commands, enter:

```
INPUT TABLE.LST;
```

To record each command and store its results in a file named TABLE.OUT, enter

```
SET ECHO ON;
OUTPUT TABLE.OUT;
INPUT TABLE.LST;
OUTPUT;
```

See Also OUTPUT

OUTPUT

Redirects output to the named file or to standard output.

Syntax `OUTPUT [filename];`

Argument	Description
filename	Name of the file in which to save output. If no file name is given, results appear on the standard output.

Description OUTPUT determines where the results of **isql** commands are displayed. By default, results are displayed on standard output (usually a screen). To store results in a file, supply a *filename* argument. To return to the default mode, again displaying results on the standard output, use OUTPUT without specifying a file name.

By default, only data is redirected. Interactive commands are not redirected unless SET ECHO is in effect. If SET ECHO is in effect, **isql** displays each command before it is executed. In this way, **isql** captures both the results and the command that produced them. SET ECHO is useful for displaying the text of a query immediately before the results.

Note Error messages cannot be redirected to an output file.

Using OUTPUT *filename* from within an **isql** session has the same effect as using the option -*output filename* from the command line.

You can optionally delimit the filename with double or single quotes. This allows you to use filenames with spaces in OUTPUT statements.

Example The following example stores the results of one SELECT statement in the file, SALES.OUT. Normal output processing resumes after the SELECT statement.

```
OUTPUT SALES.OUT;
SELECT * FROM SALES;
OUTPUT;
```

See Also INPUT, SET ECHO

QUIT

Rolls back the current transaction, closes the database, and ends the **isql** session.

Syntax `QUIT;`

Description Both EXIT and QUIT close the database and end an **isql** session. QUIT rolls back any changes made since the last COMMIT or ROLLBACK, whereas EXIT commits the changes.

IMPORTANT QUIT rolls back uncommitted changes without prompting for confirmation. Before using QUIT, be sure that any changes that need to be committed are committed. For example, if SET AUTODDL is off, DDL statements must be committed explicitly.

See Also EXIT, SET AUTODDL, ROLLBACK

SET

Lists the status of the features that control an **isql** session.

Syntax

```
SET;
```

Description **isql** provides several SET commands for specifying how data is displayed or how other commands are processed.

The SET command, by itself, verifies which features are currently set. Some SET commands turn a feature on or off. Other SET commands assign values.

Many **isql** SET commands have corresponding SQL statements that provide similar or identical functionality. In addition, some of the **isql** features controlled by SET commands can also be controlled using **isql** command-line options. SET Statements are used to configure the **isql** environment from a script file. Changes to the session setting from SET statements in a script affect the session only while the script is running. After a script completes, the session settings prior to running the script are restored.

The **isql** SET statements are:

Statement	Description	Default
SET AUTODDL	Toggles the commit feature for DDL statements	ON
SET BLOBDISPLAY *n*	Turns on the display of Blob type *n*; the parameter *n* is required to display Blob types.	OFF
SET COUNT	Toggles the count of selected rows on or off	OFF
SET ECHO	Toggles the display of each command on or off	OFF
SET LIST *string*	Displays columns vertically or horizontally	OFF

TABLE 10.10 SET statements

Statement	Description	Default
SET NAMES	Specifies the active character set	OFF
SET PLAN	Specifies whether or not to display the optimizer's query plan	OFF
SET STATS	Toggles the display of performance statistics on or off	OFF
SET TERM *string*	Allows you to change to an alternate terminator character(s)	;
SET TIME	Toggles display of time in DATE values	ON

TABLE 10.10 SET statements (*continued*)

By default, all settings are initially OFF except AUTODDL and TIME, and the terminator is a semicolon (**;**). Each time you start an **isql** session or execute an **isql** script file, settings begin with their default values.

SET statements are used to configure the **isql** environment from a script file. Changes to the session setting from SET statements in a script affect the session only while the script is running. After a script completes, the session settings prior to running the script are restored to their values before the script was run. So you can modify the settings for interactive use, then change them as needed in an **isql** script, and after running the script they automatically return to their previous configuration.

Notes

- You cannot enter **isql** SET statements interactively in the SQL Statement area of Windows ISQL. You can perform the same functions with menu items.
- SET GENERATOR and SET TRANSACTION (without a transaction name) are DSQL statements and so you can enter them interactively in Windows ISQL or **isql**. These statements are not exclusively **isql** statements, so they are not documented in this chapter. See the *Language Reference* for details.
- SET DATABASE is exclusively an embedded SQL statement. See the *Language Reference* and the *Programmer's Guide* for details.

Example To display the **isql** features currently in effect, enter:

```
SET;
   Print statistics:OFF
   Echo commands:   OFF
   List format:     OFF
```

```
Row count:       OFF
Autocommit DDL:  OFF
Access plan:     OFF
Display BLOB type:1
Terminator:      ;
Time:            OFF
```

The output shows that **isql** is set to not echo commands, to display Blob data if they are of subtype 1 (text), to automatically commit DDL statements, and to recognize a semicolon (**;**) as the statement termination character.

See Also SET AUTODDL, SET BLOBDISPLAY, SET COUNT, SET ECHO, SET LIST, SET NAMES, SET PLAN, SET STATS, SET TERMINATOR, SET TIME

SET AUTODDL

Specifies whether DDL statements are committed automatically after being executed or committed only after an explicit COMMIT.

Syntax

```
SET AUTODDL [ON | OFF];
```

Argument	Description
ON	Turns on automatic commitment of DDL (default)
OFF	Turns off automatic commitment of DDL

Description SET AUTODDL is used to turn on or off the automatic commitment of data definition language (DDL) statements. By default, DDL statements are automatically committed immediately after they are executed, in a separate transaction. This is the recommended behavior.

If the OFF keyword is specified, auto-commit of DDL is then turned off. In OFF mode, DDL statements can only be committed explicitly through a user's transaction. This mode is useful for database prototyping, because uncommitted changes are easily undone by rolling them back.

SET AUTODDL has a shorthand equivalent, SET AUTO.

TIP The ON and OFF keywords are optional. If they are omitted, SET AUTO switches from one mode to the other. Although you can save typing by omitting the optional keyword, including the keyword is recommended because it avoids potential confusion.

Examples The following example shows part of an **isql** script that turns off AUTODDL, creates a table named TEMP, then rolls back the work.

```
. . .
SET AUTO OFF;
CREATE TABLE TEMP (a INT, b INT);
ROLLBACK;
. . .
```

This script creates TEMP and then rolls back the statement. If you choose **View | Metadata** and select "Tables," the TEMP table does not appear, because its creation was rolled back.

The next script uses the default AUTODDL ON. It creates the table TEMP and then performs a rollback:

```
. . .
CREATE TABLE TEMP (a INT, b INT);
ROLLBACK;
. . .
```

Because DDL is automatically committed, the rollback does not affect the creation of TEMP. If you choose **View | Metadata** and select "Tables," you see the TEMP table.

See Also EXIT, QUIT

SET BLOBDISPLAY

Specifies subtype of Blob data to display.

Syntax `SET BLOBDISPLAY [n | ALL | OFF];`

Argument	Description
n	Integer specifying the Blob subtype to display. Use 0 for Blob data of an unknown subtype; use 1 (default) for Blob data of a text subtype, and other integer values for other subtypes.
ALL	Display Blob data of all subtypes.
OFF	Turn off display of Blob data of all subtypes.

Description SET BLOBDISPLAY has the following uses:

To display Blob data of a particular subtype, use SET BLOBDISPLAY *n*. By default, **isql** displays Blob data of text subtype ($n = 1$).

To display Blob data of all subtypes, use SET BLOBDISPLAY ALL.

To avoid displaying Blob data, use SET BLOBDISPLAY OFF. Omitting the OFF keyword has the same effect. Turn Blob display off to make output easier to read.

In any column containing Blob data, the actual data does not appear in the column. Instead, the column displays a Blob ID that represents the data. If SET BLOBDISPLAY is on, data associated with a Blob ID appears under the row containing the Blob ID. If SET BLOBDISPLAY is off, the Blob ID still appears even though its associated data does not.

SET BLOBDISPLAY has a shorthand equivalent, SET BLOB.

To determine the subtype of a Blob column, use SHOW TABLE.

Examples The following examples show output from the same SELECT statement. Each example uses a different SET BLOB command to affect how output appears. The first example turns off Blob display.

```
SET BLOB OFF;
SELECT PROJ_NAME, PROJ_DESC FROM PROJECT;
```

With BLOBDISPLAY OFF, the output shows only the Blob ID:

```
PROJ_NAME               PROJ_DESC
====================    =================
Video Database          24:6
DigiPizza               24:8
AutoMap                 24:a
MapBrowser port         24:c
Translator upgrade      24:3b
Marketing project 3     24:3d
```

The next example restores the default by setting BLOBDISPLAY to subtype 1 (text).

```
SET BLOB 1;
SELECT PROJ_NAME, PROJ_DESC FROM PROJECT;
```

Now the contents of the Blob appear below each Blob ID:

```
PROJ_NAME        PROJ_DESC
=====================================
Video Database   24:6
==============================================================
PROJ_DESC:
Design a video data base management system for
controlling on-demand video distribution.
```

```
PROJ_NAME        PROJ_DESC
=====================================
DigiPizza        24:8
==============================================================
PROJ_DESC:
Develop second generation digital pizza maker
with flash-bake heating element and
digital ingredient measuring system.
. . .
```

See Also BLOBDUMP

SET COUNT

Specifies whether to display number of rows retrieved by queries.

Syntax `SET COUNT [ON | OFF];`

Argument	Description
ON	Turns on display of the "rows returned" message
OFF	Turns off display of the "rows returned" message (default).

Description By default, when a SELECT statement retrieves rows from a query, no message appears to say how many rows were retrieved.

Use SET COUNT ON to change the default behavior and display the message. To restore the default behavior, use SET COUNT OFF.

TIP The ON and OFF keywords are optional. If they are omitted, SET COUNT switches from one mode to the other. Although you can save typing by omitting the optional keyword, including the keyword is recommended because it avoids potential confusion.

Example The following example sets COUNT ON to display the number of rows returned by all following queries:

```
SET COUNT ON;
SELECT * FROM COUNTRY
   WHERE CURRENCY LIKE '%FRANC%';
```

The output displayed would then be:

```
   COUNTRY          CURRENCY
```

```
    ===============  ==========
    SWITZERLAND      SFRANC
    FRANCE           FFRANC
    BELGIUM          BFRANC
3 rows returned
```

SET ECHO

Specifies whether commands are displayed to the **isql** Output area before being executed.

Syntax

```
SET ECHO [ON | OFF];
```

Argument	Description
ON	Turns on command echoing (default).
OFF	Turns off command echoing.

Description By default, commands in script files are displayed (echoed) in the **isql** Output area, before being executed. Use SET ECHO OFF to change the default behavior and suppress echoing of commands. This may be useful when sending the output of a script to a file, if you want only the results of the script and not the statements themselves in the output file.

Command echoing is useful if you want to see the commands as well as the results in the **isql** Output area.

TIP The ON and OFF keywords are optional. If they are omitted, SET ECHO switches from one mode to the other. Although you can save typing by omitting the optional keyword, including the keyword is recommended because it avoids potential confusion.

Example Suppose you execute the following script from Windows ISQL:

```
. . .
SET ECHO OFF;
SELECT * FROM COUNTRY;
SET ECHO ON;
SELECT * FROM COUNTRY;
EXIT;
```

The output (in a file or the **isql** Output area) looks like this:

```
. . .
SET ECHO OFF;
```

```
COUNTRY      CURRENCY
===========  ========
USA          Dollar
England      Pound
. . .
SELECT * FROM COUNTRY;
COUNTRY      CURRENCY
===========  ========
USA          Dollar
England      Pound
. . .
```

The first SELECT statement is not displayed, because ECHO is OFF. Notice also that the SET ECHO ON statement itself is not displayed, because when it is executed, ECHO is still OFF. After it is executed, however, the second SELECT statement is displayed.

See Also INPUT, OUTPUT

SET LIST

Specifies whether output appears in tabular format or in list format.

Syntax
```
SET LIST [ON | OFF];
```

Argument	Description
ON	Turns on list format for display of output.
OFF	Turns off list format for display of output (default).

Description By default, when a SELECT statement retrieves rows from a query, the output appears in a tabular format, with data organized in rows and columns.

Use SET LIST ON to change the default behavior and display output in a list format. In list format, data appears one value per line, with column headings appearing as labels. List format is useful when columnar output is too wide to fit nicely on the screen.

TIP The ON and OFF keywords are optional. If they are omitted, SET LIST switches from one mode to the other. Although you can save typing by omitting the optional keyword, including the keyword is recommended because it avoids potential confusion.

Example Suppose you execute the following statement in a script file:

```
SELECT JOB_CODE, JOB_GRADE, JOB_COUNTRY, JOB_TITLE FROM JOB
   WHERE JOB_COUNTRY = 'Italy';
```

The output is:

```
JOB_CODE   JOB_GRADE   JOB_COUNTRY   JOB_TITLE
========   =========   ===========   ====================
SRep       4           Italy         Sales Representative
```

Now suppose you precede the SELECT with SET LIST ON:

```
SET LIST ON;
SELECT JOB_CODE, JOB_GRADE, JOB_COUNTRY, JOB_TITLE FROM JOB
   WHERE JOB_COUNTRY = 'Italy';
```

The output is:

```
   JOB_CODE         SRep
   JOB_GRADE        4
   JOB_COUNTRY      Italy
   JOB_TITLE        Sales Representative
```

SET NAMES

Specifies the active character set to use in database transactions.

Syntax

```
SET NAMES [charset];
```

Argument	Description
charset	Name of the active character set. Default: NONE.

Description SET NAMES specifies the character set to use for subsequent database connections in **isql**. It enables you to override the default character set for a database. To return to using the default character set, use SET NAMES with no argument.

Use SET NAMES before connecting to the database whose character set you want to specify. For a complete list of character sets recognized by InterBase, see the *Language Reference*.

Choice of character set limits possible collation orders to a subset of all available collation orders. Given a specific character set, a specific collation order can be specified when data is selected, inserted, or updated in a column.

Example The following statement at the beginning of a script file indicates to set the active character set to ISO8859_1 for the subsequent database connection:

```
SET NAMES ISO8859_1;
CONNECT 'jupiter:/usr/interbase/examples/employee.gdb';
. . .
```

SET PLAN

Specifies whether to display the optimizer's query plan.

Syntax

```
SET PLAN [ON | OFF];
```

Argument	Description
ON	Turns on display of the optimizer's query plan.
OFF	Turns off display of the optimizer's query plan (default).

Description By default, when a SELECT statement retrieves rows from a query, **isql** does not display the query plan used to retrieve the data.

Use SET PLAN ON to change the default behavior and display the query optimizer plan. To restore the default behavior, use SET PLAN OFF.

To change the query optimizer plan, use the PLAN clause in the SELECT statement.

TIP The ON and OFF keywords are optional. If they are omitted, SET PLAN switches from one mode to the other. Although you can save typing by omitting the optional keyword, including the keyword is recommended because it avoids potential confusion.

Example The following example shows part of a script which sets PLAN ON:

```
SET PLAN ON;
SELECT JOB_COUNTRY, MIN_SALARY FROM JOB
   WHERE MIN_SALARY > 50000
      AND JOB_COUNTRY = 'France';
```

The output then includes the query optimizer plan used to retrieve the data as well as the results of the query:

```
PLAN (JOB INDEX (RDB$FOREIGN3,MINSALX,MAXSALX))
JOB_COUNTRY      MIN_SALARY
===============  ======================
France           118200.00
```

SET STATS

Specifies whether to display performance statistics after the results of a query.

Syntax

```
SET STATS [ON | OFF];
```

Argument	Description
ON	Turns on display of performance statistics.
OFF	Turns off display of performance statistics (default).

Description By default, when a SELECT statement retrieves rows from a query, **isql** does not display performance statistics after the results. Use SET STATS ON to change the default behavior and display performance statistics. To restore the default behavior, use SET STATS OFF. Performance statistics include:

- Current memory available, in bytes
- Change in available memory, in bytes
- Maximum memory available, in bytes
- Elapsed time for the operation
- CPU time for the operation
- Number of cache buffers used
- Number of reads requested
- Number of writes requested
- Number of fetches made

Performance statistics can help determine if changes are needed in system resources, database resources, or query optimization.

TIP The ON and OFF keywords are optional. If they are omitted, SET STATS switches from one mode to the other. Although you can save typing by omitting the optional keyword, including the keyword is recommended because it avoids potential confusion.

Do not confuse SET STATS with the SQL statement SET STATISTICS, which recalculates the selectivity of an index.

Example The following part of a script file turns on display of statistics and then performs a query:

```
SET STATS ON;
```

```
SELECT JOB_COUNTRY, MIN_SALARY FROM JOB
   WHERE MIN_SALARY > 50000
      AND JOB_COUNTRY = 'France';
```

The output displays the results of the SELECT statement and the performance statistics for the operation:

```
   JOB_COUNTRY      MIN_SALARY
   ===============  ======================
   France           118200.00

Current memory = 407552
Delta memory = 0
Max memory = 412672
Elapsed time= 0.49 sec
Cpu = 0.06 sec
Buffers = 75
Reads = 3
Writes = 2
Fetches = 441
```

See Also SHOW DATABASE

SET TERM

Specifies which character or characters signal the end of a command.

Syntax `SET TERM *string*;`

Argument	Description
string	Specifies a character or characters to use in terminating a statement. Default: semicolon (**;**).

Description By default, when a line ends with a semicolon, **isql** interprets it as the end of a command. Use SET TERM to change the default behavior and define a new termination character.

SET TERM is typically used with CREATE PROCEDURE or CREATE TRIGGER. Procedures and triggers are defined using a special "procedure and trigger" language in which statements end with a semicolon. If **isql** were to interpret semicolons as statement terminators, then procedures and triggers would execute during their creation, rather than when they are called.

A script file containing CREATE PROCEDURE or CREATE TRIGGER definitions should include one SET TERM command before the definitions and a corresponding SET TERM after the definitions. The beginning SET TERM defines a new termination character; the ending SET TERM restores the semicolon (**;**) as the default.

Note You do not need to change the terminator before entering an interactive CREATE PROCEDURE or CREATE TRIGGER statement in the Windows ISQL SQL statement area. The contents of the SQL statement area is always treated as one DSQL statement, even if it contains semicolons. Use of SET TERM is necessary only in **isql** and when running SQL script files from **isql** or Windows ISQL.

Example The following example shows a text file that uses SET TERM in creating a procedure. The first SET TERM defines "##" as the termination characters. The matching SET TERM restores ";" as the termination character.

```
SET TERM ## ;
CREATE PROCEDURE ADD_EMP_PROJ (EMP_NO SMALLINT, PROJ_ID CHAR(5))
AS
BEGIN
   BEGIN
      INSERT INTO EMPLOYEE_PROJECT (EMP_NO, PROJ_ID)
         VALUES (:emp_no, :proj_id);
   WHEN SQLCODE -530 DO
   EXCEPTION UNKNOWN_EMP_ID;
   END
   RETURN;
END ##
SET TERM ; ##
```

SET TIME

Specifies whether to display the time portion of a DATE value.

Syntax

```
SET TIME [ON | OFF];
```

Argument	Description
ON	Turns on display of time in DATE value.
OFF	Turns off display of time in DATE value (default).

Description The InterBase Date datatype includes a date portion (including day, month, and year) and a time portion (including hours, minutes, and seconds).

By default, **isql** displays only the date portion of Date values. SET TIME ON turns on the display of time values. SET TIME OFF turns off the display of time values.

TIP The ON and OFF keywords are optional. If they are omitted, the command toggles time display from ON to OFF or OFF to ON.

Example The following example shows the default display of a Date datatype, which is to display day, month, and year:

```
SELECT HIRE_DATE FROM EMPLOYEE WHERE EMP_NO = 145;
HIRE_DATE
-------------------
2-MAY-1994
```

This example shows the effects of SET TIME ON, which causes the hours, minutes and seconds to be displayed as well:

```
SET TIME ON;
SELECT HIRE_DATE FROM EMPLOYEE WHERE EMP_NO = 145;
HIRE_DATE
-------------------
2-MAY-1994 12:25:00
```

SHELL

Allows execution of an operating system command or temporary access to an operating system shell.

Syntax

```
SHELL [<os_command>];
```

Argument	Description
<os_command>	An operating system command. If no command is specified, **isql** provides interactive access to the operating system.

Description The SHELL command provides temporary access to operating system commands in an **isql** session. Use SHELL to execute an operating-system command without ending the current **isql** session.

If *<os_command>* is specified, the operating system executes the command and then returns to **isql** when complete.

If no command is specified, an operating system shell prompt appears, enabling you to execute a sequence of commands. To return to **isql**, type **exit**. For example, SHELL can be used to edit an input file and run it at a later time. By contrast, if an input file is edited using the EDIT command, the input file is executed as soon as the editing session ends.

Using SHELL does not commit transactions before it calls the shell.

There is no equivalent function in Windows ISQL to this **isql** statement.

Example The following example uses SHELL to display the contents of the current directory:

```
SHELL DIR;
```

See Also EDIT

SHOW CHECK

Displays all CHECK constraints defined for a specified table.

Syntax

```
SHOW CHECK table;
```

Argument	Description
table	Name of an existing table in the current database.

Description SHOW CHECK displays CHECK constraints for a named table in the current database. Only user-defined metadata is displayed. To see a list of existing tables, use SHOW TABLE.

Example The following example shows CHECK constraints defined for the table, JOB. The SHOW TABLES command is used first to display a list of available tables.

```
SHOW TABLES;
   COUNTRY          CUSTOMER
   DEPARTMENT       EMPLOYEE
   EMPLOYEE_PROJECT JOB
   PHONE_LIST       PROJECT
   PROJ_DEPT_BUDGET SALARY_HISTORY
   SALES

SHOW CHECK JOB;
   CHECK (min_salary < max_salary)
```

See Also SHOW TABLE

SHOW DATABASE

Displays information about the current database.

Syntax `SHOW [DATABASE | DB];`

Description SHOW DATABASE displays the current database's file name, page size and allocation, and sweep interval.

The output of SHOW DATABASE is used to verify data definition or to administer the database. For example, use the backup and restore utilities to change page size or reallocate pages among multiple files, and use the database maintenance utility to change the sweep interval.

SHOW DATABASE has a shorthand equivalent, SHOW DB.

Example The following example connects to a database and displays information about it:

```
CONNECT 'employee.gdb';
   Database: employee.gdb

SHOW DB;
   Database: employee.gdb
         Owner: SYSDBA
   PAGE_SIZE 1024
   Number of DB pages allocated = 422
   Sweep interval = 20000
```

SHOW DOMAINS

Lists all domains or displays information about a specified domain.

Syntax `SHOW {DOMAINS | DOMAIN name};`

Argument	Description
name	Name of an existing domain in the current database.

Options To see a list of existing domains, use SHOW DOMAINS without specifying a domain name. SHOW DOMAIN *name* displays information about the named domain in the current database. Output includes a domain's datatype, default value, and any CHECK constraints defined. Only user-defined metadata is displayed.

Example The following example lists all domains and then shows the definition of the domain, SALARY:

```
SHOW DOMAINS;
   FIRSTNAME        LASTNAME
   PHONENUMBER      COUNTRYNAME
   ADDRESSLINE      EMPNO
   DEPTNO           PROJNO
   CUSTNO           JOBCODE
   JOBGRADE         SALARY
   BUDGET           PRODTYPE
   PONUMBER

SHOW DOMAIN SALARY;
   SALARY           NUMERIC(15, 2) Nullable
                    DEFAULT 0
                    CHECK (VALUE > 0)
```

SHOW EXCEPTIONS

Lists all exceptions or displays the text of a specified exception.

Syntax `SHOW {EXCEPTIONS | EXCEPTION name};`

Argument	Description
name	Name of an existing exception in the current database.

Description SHOW EXCEPTIONS displays an alphabetical list of exceptions. SHOW EXCEPTION *name* displays the text of the named exception.

Examples To list all exceptions defined for the current database, enter:

```
SHOW EXCEPTIONS;
Exception Name   Used by, Type
================== ========================================
UNKNOWN_EMP_ID   ADD_EMP_PROJ, Stored procedure
   Invalid employee number or project ID.
. . .
```

To list the message for a specific exception and the procedures or triggers which use it, enter the exception name:

```
SHOW EXCEPTION CUSTOMER_CHECK;
Exception Name                   Used by, Type
===========================  =======================================
CUSTOMER_CHECK               SHIP_ORDER, Stored procedure
 Overdue balance -- can’t ship.
```

SHOW FILTERS

Lists all Blob filters or displays information about a specified filter.

Syntax `SHOW {FILTERS | FILTER name};`

Argument	Description
name	Name of an existing Blob filter in the current database.

Options To see a list of existing filters, use SHOW FILTERS. SHOW FILTER *name* displays information about the named filter in the current database. Output includes information previously defined by the DECLARE FILTER statement, the input subtype, output subtype, module (or library) name, and entry point name.

Example The following example lists all filters and then shows the definition of the filter, DESC_FILTER:

```
SHOW FILTERS;
   DESC_FILTER

SHOW FILTER DESC_FILTER;
   BLOB Filter: DESC_FILTER
   Input subtype: 1 Output subtype -4
   Filter library is: desc_filter
   Entry point is: FILTERLIB
```

SHOW FUNCTIONS

Lists all user-defined functions (UDFs) defined in the database or displays information about a specified UDF.

Syntax `SHOW {FUNCTIONS | FUNCTION name};`

Argument	Description
name	Name of an existing UDF in the current database.

Options To see a list of existing functions defined in the database, use SHOW FUNCTIONS. SHOW FUNCTION *name* displays information about the named function in the current database. Output includes information previously defined by the DECLARE EXTERNAL FUNCTION statement: the name of the function and function library, the name of the entry point, and the datatypes of return values and input arguments.

Example The following example lists all UDFs and then shows the definition of the **maxnum()** function:

```
SHOW FUNCTIONS;
   ABS      MAXNUM
   TIME     UPPER_NON_C
   UPPER
```

```
SHOW FUNCTION maxnum;
   Function MAXNUM:
   Function library is /usr/interbase/lib/gdsfunc.so
   Entry point is FN_MAX
   Returns BY VALUE DOUBLE PRECISION
   Argument 1: DOUBLE PRECISION
   Argument 2: DOUBLE PRECISION
```

SHOW GENERATORS

Lists all generators or displays information about a specified generator.

Syntax `SHOW {GENERATORS | GENERATOR name};`

Argument	Description
name	Name of an existing generator in the current database.

Description To see a list of existing generators, use SHOW GENERATORS. SHOW GENERATOR *name* displays information about the named generator in the current database. Output includes the name of the generator and its next value.

SHOW GENERATOR has a shorthand equivalent, SHOW GEN.

Example The following example lists all generators and then shows information about EMP_NO_GEN:

```
SHOW GENERATORS;
   Generator EMP_NO_GEN, Next value: 146
   Generator CUST_NO_GEN, Next value: 1016

SHOW GENERATOR EMP_NO_GEN;
   Generator EMP_NO_GEN, Next value: 146
```

SHOW GRANT

Displays privileges for a database object.

Syntax

```
SHOW GRANT object;
```

Argument	Description
object	Name of an existing table, view, or procedure in the current database.

Description SHOW GRANT displays the privileges defined for a specified table, view, or procedure. Allowed privileges are DELETE, EXECUTE, INSERT, SELECT, UPDATE, or ALL. To change privileges, use the SQL statements GRANT or REVOKE.

Before using SHOW GRANT, you might want to list the available database objects. Use SHOW PROCEDURES to list existing procedures; use SHOW TABLES to list existing tables; use SHOW VIEWS to list existing views.

Example To display GRANT privileges on the table, JOB, enter:

```
SHOW GRANT JOB;
   GRANT SELECT ON JOB TO ALL
   GRANT DELETE, INSERT, SELECT, UPDATE ON JOB TO MANAGER
```

SHOW GRANT can also show role membership:

```
SHOW GRANT DOITALL;
   GRANT DOITALL TO SOCKS
```

See Also SHOW PROCEDURES, SHOW TABLES, SHOW VIEWS

SHOW INDEX

Displays index information for a specified index, for a specified table, or for all tables in the current database.

Syntax

```
SHOW {INDICES | INDEX {index | table} };
```

Argument	Description
index	Name of an existing index in the current database.
table	Name of an existing table in the current database.

Description SHOW INDEX displays the index name, the index type (for example, UNIQUE or DESC), and the columns on which an index is defined.

If the *index* argument is specified, SHOW INDEX displays information only for that index. If *table* is specified, SHOW INDEX displays information for all indexes in the named table; to display existing tables, use SHOW TABLES. If no argument is specified, SHOW INDEX displays information for all indexes in the current database.

SHOW INDEX has a shorthand equivalent, SHOW IND. SHOW INDICES is also a synonym for SHOW INDEX. SHOW INDEXES is not supported.

Examples To display indexes for database *employee.gdb*, enter:

```
SHOW INDEX;
   RDB$PRIMARY1 UNIQUE INDEX ON COUNTRY(COUNTRY)
   CUSTNAMEX INDEX ON CUSTOMER(CUSTOMER)
   CUSTREGION INDEX ON CUSTOMER(COUNTRY, CITY)
   RDB$FOREIGN23 INDEX ON CUSTOMER(COUNTRY)
. . .
```

To display index information for the SALES table, enter:

```
SHOW IND SALES;
   NEEDX INDEX ON SALES(DATE_NEEDED)
   QTYX DESCENDING INDEX ON SALES(ITEM_TYPE, QTY_ORDERED)
   RDB$FOREIGN25 INDEX ON SALES(CUST_NO)
   RDB$FOREIGN26 INDEX ON SALES(SALES_REP)
   RDB$PRIMARY24 UNIQUE INDEX ON SALES(PO_NUMBER)
   SALESTATX INDEX ON SALES(ORDER_STATUS, PAID)
```

See Also SHOW TABLES

SHOW PROCEDURES

Lists all procedures or displays the text of a specified procedure.

Syntax `SHOW {PROCEDURES | PROCEDURE name};`

Argument	Description
name	Name of an existing procedure in the current database.

Description SHOW PROCEDURES displays an alphabetical list of procedures, along with the database objects they depend on. Deleting a database object that has a dependent procedure is not allowed. To avoid an **isql** error, delete the procedure (using DROP PROCEDURE) before deleting the database object.

SHOW PROCEDURE *name* displays the text and parameters of the named procedure.

SHOW PROCEDURE has a shorthand equivalent, SHOW PROC.

Examples To list all procedures defined for the current database, enter:

```
SHOW PROCEDURES;
   Procedure Name   Dependency Type
   ================ ==================== =======
   ADD_EMP_PROJ     EMPLOYEE_PROJECTTable
                    UNKNOWN_EMP_IDException
   DELETE_EMPLOYEE  DEPARTMENTTable
                    EMPLOYEETable
                    EMPLOYEE_PROJECTTable
                    PROJECTTable
                    REASSIGN_SALESException
                    SALARY_HISTORYTable
                    SALES  Table
   DEPT_BUDGET      DEPARTMENTTable
                    DEPT_BUDGETProcedure
. . .
```

To display the text of the procedure, ADD_EMP_PROJ, enter:

```
SHOW PROC ADD_EMP_PROJ;
   Procedure text:
   =============================================================

   BEGIN
      BEGIN
      INSERT INTO EMPLOYEE_PROJECT (EMP_NO, PROJ_ID) VALUES (:emp_no,
         :proj_id);
      WHEN SQLCODE -530 DO
      EXCEPTION UNKNOWN_EMP_ID;
      END
      RETURN;
   END
   =============================================================
   Parameters:
```

```
   EMP_NO INPUT SMALLINT
   PROJ_ID INPUT CHAR(5)
```

SHOW ROLES

Displays the names of SQL roles for the current database.

Syntax

```
SHOW {ROLES | ROLE}
```

Description SHOW ROLES displays the names of all roles defined for the current database. To show user membership in roles, use SHOW GRANT *rolename*.

Example

```
SHOW ROLES;
   DOITALL      DONOTHING
   DOONETHING   DOSOMETHING
```

See Also SHOW GRANT

SHOW SYSTEM

Displays the names of system tables and system views for the current database.

Syntax

```
SHOW SYSTEM [TABLES];
```

Description SHOW SYSTEM lists system tables and system views in the current database. SHOW SYSTEM accepts an optional keyword, TABLES, which does not affect the behavior of the command.

SHOW SYSTEM has a shorthand equivalent, SHOW SYS.

Example To list system tables and system views for the current database, enter:

```
SHOW SYS;
   RDB$CHARACTER_SETS      RDB$CHECK_CONSTRAINTS
   RDB$COLLATIONS          RDB$DATABASE
   RDB$DEPENDENCIES        RDB$EXCEPTIONS
   RDB$FIELDS              RDB$FIELD_DIMENSIONS
   RDB$FILES               RDB$FILTERS
   RDB$FORMATS             RDB$FUNCTIONS
   RDB$FUNCTION_ARGUMENTS  RDB$GENERATORS
   RDB$INDEX_SEGMENTS      RDB$INDICES
   RDB$LOG_FILES           RDB$PAGES
```

```
    RDB$PROCEDURES          RDB$PROCEDURE_PARAMETERS
    RDB$REF_CONSTRAINTS     RDB$RELATIONS
    RDB$RELATION_CONSTRAINTS RDB$RELATION_FIELDS
    RDB$ROLES               RDB$SECURITY_CLASSES
    RDB$TRANSACTIONS        RDB$TRIGGERS
    RDB$TRIGGER_MESSAGES    RDB$TYPES
    RDB$USER_PRIVILEGES     RDB$VIEW_RELATIONS
```

See Also For more information about system tables, see the *Language Reference.*

SHOW TABLES

Lists all tables or views, or displays information about a specified table or view.

Syntax

```
SHOW {TABLES | TABLE name};
```

Argument	Description
name	Name of an existing table or view in the current database.

Description SHOW TABLES displays an alphabetical list of tables and views in the current database. To determine which listed objects are views rather than tables, use SHOW VIEWS.

SHOW TABLE *name* displays information about the named object. If the object is a table, command output lists column names and definitions, PRIMARY KEY, FOREIGN KEY, and CHECK constraints, and triggers. If the object is a view, command output lists column names and definitions, as well as the SELECT statement that the view is based on.

Examples To list all tables or views defined for the current database, enter:

```
SHOW TABLES;
    COUNTRY          CUSTOMER
    DEPARTMENT       EMPLOYEE
    EMPLOYEE_PROJECT JOB
    PHONE_LIST       PROJECT
    PROJ_DEPT_BUDGET SALARY_HISTORY
    SALES
```

To show the definition for the COUNTRY table, enter:

```
SHOW TABLE COUNTRY;
    COUNTRY (COUNTRYNAME) VARCHAR(15) NOT NULL
```

```
   CURRENCY VARCHAR(10) NOT NULL
   PRIMARY KEY (COUNTRY)
```

See Also SHOW VIEWS

SHOW TRIGGERS

Lists all triggers or displays information about a specified trigger.

Syntax `SHOW {TRIGGERS | TRIGGER name};`

Argument	Description
name	Name of an existing trigger in the current database.

Description SHOW TRIGGERS displays all triggers defined in the database, along with the table they depend on. SHOW TRIGGER *name* displays the name, sequence, type, activation status, and definition of the named trigger.

SHOW TRIGGER has a shorthand equivalent, SHOW TRIG.

Deleting a table that has a dependent trigger is not allowed. To avoid an **isql** error, delete the trigger (using DROP TRIGGER) before deleting the table.

Examples To list all triggers defined for the current database, enter:

```
SHOW TRIGGERS;
   Table name      Trigger name
   ===========     ============
   EMPLOYEE        SET_EMP_NO
   EMPLOYEE        SAVE_SALARY_CHANGE
   CUSTOMER        SET_CUST_NO
   SALES           POST_NEW_ORDER
```

To display information about the SET_CUST_NO trigger, enter:

```
SHOW TRIG SET_CUST_NO;

   Triggers:
   SET_CUST_NO, Sequence: 0, Type: BEFORE INSERT, Active
   AS
   BEGIN
      new.cust_no = gen_id(cust_no_gen, 1);
   END
```

SHOW VERSION

Displays information about software versions.

Syntax

```
SHOW VERSION;
```

Description SHOW VERSION displays the software version of **isql**, the InterBase engine, and the on-disk structure (ODS) of the database to which the session is attached.

Certain tasks might not work as expected if performed on databases that were created using older versions of InterBase. To check the versions of software that are running, use SHOW VERSION.

SHOW VERSION has a shorthand equivalent, SHOW VER.

Example To display software versions, enter:

```
SHOW VER;
   ISQL Version: WI-V5.5.5
   InterBase/Windows NT (access method), version 'WI-V5.5.5'
   on disk structure version 9.1
```

See Also SHOW DATABASE

SHOW VIEWS

Lists all views or displays information about a specified view.

Syntax

```
SHOW {VIEWS | VIEW name};
```

Argument	Description
name	Name of an existing view in the current database.

Description SHOW VIEWS displays an alphabetical list of all views in the current database. SHOW VIEW *name* displays information about the named view.

Example To list all views defined for the current database, enter:

```
SHOW VIEWS;
   PHONE_LIST
```

See Also SHOW TABLES

Using SQL scripts

The basic steps for using script files with Windows ISQL are:

1. Create the file using a text editor
2. Execute the file with **isql** or Windows ISQL
3. View output and confirm database changes

Creating an isql script

You can use any text editor to create an SQL script file, as long as the final file format is "plain text" (ASCII).

Every SQL script file must begin with either a CREATE DATABASE statement or a CONNECT statement (including username and password) to specify the database on which the script file operates.

The CONNECT or CREATE statement must contain a complete database file name and directory path.

An SQL script can contain any of the following elements:

- SQL statements, as described in the *Language Reference*
- **isql** SET commands as described in this chapter
- Comments.

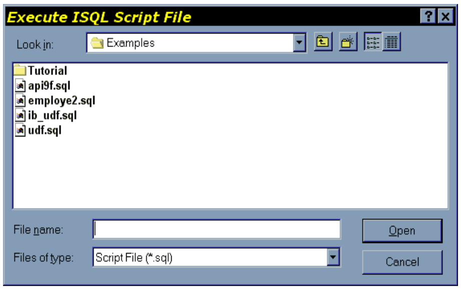
Each SQL statement in a script *must* be terminated by a semicolon (**;**) or the current terminator if it has been changed with SET TERM. Furthermore, the SQL statement silently fails if significant text follows the terminator character on the same line. Whitespace and comments can safely follow the terminator, but other statements cannot.

Each SQL script file should end with either EXIT to commit database changes made since the last COMMIT, or QUIT to roll back changes made by the script. If neither is specified, then database changes are committed by default.

For the full syntax of CONNECT and CREATE DATABASE, see the *Language Reference*.

Executing an SQL script

To execute a file containing SQL statements, choose **File | Execute ISQL Script**. The following dialog appears:



Enter the path and name of the file and click OK. A dialog appears asking, "Save output to a file?" If you choose Yes, then another dialog opens, enabling you to enter a file name to which to save output. If you choose No, then output and any error messages are displayed in the SQL Output area.

After Windows ISQL finishes executing a script file, a summary dialog appears indicating if there were any errors. If there were errors, then an error message appears in the **isql** Output Area (or output file) after each statement that caused the error.

After a script is executed, all Windows ISQL settings prior to executing it are restored as well as the previous database connection, if any. Any **isql** SET commands in the script affect only the **isql** session while the script is running.

Committing work in an SQL script

Changes to the database from data definition (DDL) statements—for example, CREATE and ALTER statements—are automatically committed by default. This means that other users of the database see changes as soon as each DDL statement is executed. To turn off automatic commit of DDL in a script, use SET AUTODDL OFF.

Note When creating tables and other database objects with AUTODDL OFF, it is good practice to put a COMMIT statement in the SQL script after each CREATE statement or group of related statements. This ensures that other users of the database see the objects immediately.

Changes made to the database by data manipulation (DML) statements—for example INSERT and UPDATE—are not permanent until they are committed. Commit changes in a script with COMMIT. To undo all database changes since the last COMMIT, use ROLLBACK. For the full syntax of COMMIT and ROLLBACK, see the *Language Reference* book.

Adding comments in an isql script

isql scripts are commented exactly like C programs:

```
/* comment */
```

A comment can occur on the same line as an SQL statement or **isql** command and can be of any length, as long as it is preceded by "/*" and followed by "*/".

APPENDIX

A

InterBase Document Conventions

This appendix describes the InterBase 5 documentation set, the printing conventions used to display information in text and in code examples, and conventions for naming database objects and files in applications.

The InterBase documentation set

The InterBase documentation set is an integrated package designed for all levels of users. It consists of five printed books. Each of these books is also provided in Adobe Acrobat PDF format and is accessible on line through the Help menu. If Adobe Acrobat is not already installed on your system, you can find it on the InterBase distribution CD-ROM or at *http//www.adobe.com/prodindex/acrobat/readstep.html*. Acrobat is available for Windows NT, Windows 95, and most flavors of UNIX. Windows users also have help available through the WinHelp system.

Book	Description
Operations Guide	Provides an introduction to InterBase and an explanation of tools and procedures for performing administrative tasks on databases and database servers. Also includes full reference on InterBase utilities, including isql, gbak, Server Manager for Windows, and others.
Data Definition Guide	Explains how to create, alter, and delete database objects through ISQL.
Language Reference	Describes SQL and DSQL syntax and usage.
Programmer's Guide	Describes how to write embedded SQL and DSQL database applications in a host language, precompiled through gpre.
API Guide	Explains how to write database applications using the InterBase API.

TABLE A.1 Books in the InterBase 5 documentation set

Printing conventions

The InterBase documentation set uses various typographic conventions to identify objects and syntactic elements.

The following table lists typographic conventions used in text, and provides examples of their use:

Convention	Purpose	Example
UPPERCASE	SQL keywords, SQL functions, and names of all database objects such as tables, columns, indexes, and stored procedures.	The following SELECT statement retrieves data from the CITY column in the CITIES table.
italic	New terms, emphasized words, file names, and host- language variables.	The *isc4.gdb s*ecurity database is not accessible without a valid user name and password.
bold	Utility names, user-defined functions, and host-language function names. Function names are always followed by parentheses to distinguish them from utility names.	Use **gbak** to back up and restore a database. Use the **datediff()** function to calculate the number of days between two dates.

TABLE A.2 Text conventions

Syntax conventions

The following table lists the conventions used in syntax statements and sample code, and provides examples of their use:

Convention	Purpose	Example
UPPERCASE	Keywords that must be typed exactly as they appear when used.	`SET TERM !!;`
italic	Parameters that cannot be broken into smaller units. For example, a table name cannot be subdivided.	`CREATE GENERATOR` *`name`*`;`
<italic>	Parameters in angle brackets that *can* be broken into smaller syntactic units.	`WHILE (<`*`condition`*`>) DO <`*`compound_statement`*`>`
[ ]	Optional syntax: you do not need to include anything that is enclosed in square brackets.	`CREATE [UNIQUE][ASCENDING|DESCENDING]`
{ }	One of the enclosed options *must* be included in actual statement use. If the contents are separated by a pipe symbol (|), you must choose only one.	`{SMALLINT | INTEGER | FLOAT | DOUBLE PRECISION}`

|	You can choose only one of a group whose elements are separated by this pipe symbol.
	When objects separated by this symbol occur within curly brackets, you *must* choose one; when they are within square brackets you can choose one or none.
		`SET {DATABASE | SCHEMA}`
		`SELECT [DISTINCT |ALL]`

...	The clause enclosed in brackets with the … symbol can be repeated as many times as necessary.	`(<`*`col`*`> [,<`*`col`*`>…])`

TABLE A.3 Syntax conventions

Index