Embarcadero® InterBase XE7™

Operations Guide

Published November, 2014

© 11/14/14 Embarcadero Technologies, Inc. Embarcadero, the Embarcadero Technologies logos, and all other Embarcadero Technologies product or service names are trademarks or registered trademarks of Embarcadero Technologies, Inc. All other trademarks are property of their respective owners.

This software/documentation contains proprietary information of Embarcadero Technologies, Inc.; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited.

Embarcadero Technologies, Inc. is a leading provider of award-winning tools for application developers and database professionals so they can design systems right, build them faster and run them better, regardless of their platform or programming language. Ninety of the Fortune 100 and an active community of more than three million users worldwide rely on Embarcadero products to increase productivity, reduce costs, simplify change management and compliance, and accelerate innovation. Founded in 1993, Embarcadero is headquartered in San Francisco, with offices located around the world. To learn more, please visit http://www.embarcadero.com.

November 14, 2014

Contents

Contents xiii

Tables xiii

Figures xv

Chapter 1
Introduction

Who Should Use this Guide 1-1
Topics Covered in this Guide 1-1
InterBase PDF Documentation 1-2
About Enhanced Acrobat Reader 1-2
Installing Acrobat 1-3
Using Full-Text Search 1-3
System Requirements and Server Sizing 1-4
Primary InterBase Features 1-5
SQL Support 1-7
Multiuser Database Access 1-7
Transaction Management 1-7
Multigenerational Architecture 1-7
Optimistic Row-level Locking. 1-8
Database Administration 1-8
Managing Server Security 1-8
Backing Up and Restoring Databases . . 1-9
Maintaining a Database 1-9
Viewing Statistics 1-9
About InterBase SuperServer Architecture . . 1-10
Overview of Command-line Tools 1-10
isql 1-10
gbak 1-10
gfix 1-11
gsec 1-11
gstat 1-11
iblockpr (gds_lock_print) 1-11
ibmgr 1-12

Chapter 2
Licensing

InterBase License Options 2-1
Server Edition 2-2
Developer Edition 2-3
Desktop Edition 2-3
Add-ons available for the Desktop Edition 2-3
ToGo Edition 2-4
Using the License Manager 2-5
Accessing the License Manager 2-6

Chapter 3
IBConsole: The InterBase Interface

Starting IBConsole 3-2
IBConsole Menus 3-2
Context Menus 3-3
IBConsole Toolbar 3-4
Tree Pane 3-5
Work Pane 3-7
Standard Text Display Window 3-7
Switching Between IBConsole Windows . . 3-8
Managing Custom Tools in IBConsole. . . . 3-8
Monitoring Database Performance 3-9
Displaying the Performance Monitor . . . 3-9

Chapter 4
Server Configuration

Configuring Server Properties 4-1
The General Tab 4-2
The Alias Tab 4-3
Multi-Instance 4-3
Windows Server Setup 4-4
Accessing Remote Databases. 4-4
Client Side Settings. 4-4
Remote Servers 4-4
Accessing Local Databases 4-5
Automatic Rerouting of Databases 4-6
Server Side Setup 4-6
Client Side Settings. 4-7
Startup Parameters 4-7
SMP Support 4-8
Expanded Processor Control: CPU_AFFINITY . 4-8
ibconfig Parameter: MAX_THREADS 4-9
ibconfig Parameter: THREAD_STACK_SIZE_MB 2. 4-9
Hyper-threading Support on Intel Processors 4-10
Using InterBase Manager to Start and Stop InterBase 4-10
Starting and Stopping the InterBase Server on UNIX. 4-11
Using ibmgr to Start and Stop the Server . 4-11
Starting the Server 4-12
Stopping the Server 4-13
Starting the Server Automatically 4-13
The Attachment Governor 4-16

Using Environment Variables 4-17
ISC_USER and ISC_PASSWORD. 4-17
The INTERBASE Environment Variables . 4-17
The TMP Environment Variable 4-18
UNIX and Linux Host Equivalence 4-18
Managing Temporary Files 4-18
Configuring History Files 4-19
Configuring Sort Files 4-19
Configuring Parameters in *ibconfig* *4-20*
Viewing the Server Log File 4-28

Chapter 5
Network Configuration

Network Protocols 5-1
Connecting to Servers and Databases 5-2
Adding a Server 5-2
Logging in to a Server 5-9
Logging Out from a Server 5-10
Un-registering a Server 5-10
Adding a Database 5-11
Connecting to a Database 5-12
Connect 5-12
Connect As 5-13
Disconnecting a Database 5-14
Un-registering a Database 5-14
Connection-specific Examples 5-15
Encrypting Network Communication 5-15
Requirements and Constraints 5-15
Setting up OTW Encryption 5-16
Generating Security Certificates 5-16
Setting up the Client Side 5-17
Setting up the Server Side 5-21
Sample OTW Configurations. 5-24
Sample 1: Setting up the Client and Server Without Client Verification by the Server . 5-24
Sample 2: Setting up the Client and Server for Verifying the Client 5-25
Sample 3: Setting up a JDBC Client and InterBase Server for Verifying the Client . 5-26
Connection Troubleshooting 5-28
Connection Refused Errors 5-28
Is there low-level network access between the client and server? 5-28
Can the client resolve the server's hostname? 5-28
Is the server behind a firewall? 5-28
Are the client and server on different subnets? 5-29
Can you connect to a database locally? 5-29
Can you connect to a database loopback? . . 5-29
Is the server listening on the InterBase port? 5-29
Is the services file configured on client and server? 5-30
Connection Rejected Errors 5-30
Did you get the correct path to the database? 5-30
Is UNIX host equivalence established?. 5-30
Is the database on a networked file system? . 5-30
Are the user and password valid? . . . 5-30
Does the server have permissions on the database file? 5-31
Does the server have permissions to create files in the InterBase install directory? 5-31
Disabling Automatic Internet Dialup 5-31
Reorder network adapter bindings . . . 5-31
Disabling autodial in the registry 5-31
Preventing RAS from dialing out for local network activity 5-32
Other Errors 5-32
Unknown Win32 error 10061 5-32
Unable to complete network request to host . 5-32
Communication Diagnostics 5-33
DB Connection Tab 5-34
To Run a DB Connection Test 5-34
Sample output (local connection). . . . 5-35
TCP/IP Tab 5-35
NetBEUI Tab 5-36

Chapter 6
Database User Management

Security Model 6-1
The SYSDBA User 6-2
Other Users 6-2
Users on UNIX. 6-2
The InterBase Security Database. 6-3
Implementing Stronger Password Protection . . 6-4
Enabling Embedded User Authentication. . . . 6-6
Check if EUA is Active with isc_database Info API 6-6
Enabling EUA Using iSQL 6-6
Enabling EUA Using IBConsole 6-7

Adding and Modifying Users in a EUA-enabled Database . . . 6-8
System Table Security . . . 6-8
Older Databases . . . 6-8
Scripts for Changing Database Security . . . 6-9
Migration Issues . . . 6-9
SQL Privileges . . . 6-10
Groups of Users . . . 6-10
SQL Roles . . . 6-10
UNIX Groups . . . 6-11
Other Security Measures . . . 6-11
Restriction on Using InterBase Tools . . . 6-11
Protecting your Databases . . . 6-12
User Administration with IBConsole . . . 6-12
Displaying the User Information Dialog . . . 6-12
Adding a User . . . 6-13
Modifying User Configurations . . . 6-14
Deleting a User . . . 6-15
User Administration With the InterBase API . . . 6-15
Using gsec to Manage Security . . . 6-16
Running gsec Remotely . . . 6-16
Running gsec with Embedded Database User Authentication . . . 6-16
Using gsec Commands . . . 6-16
Displaying the Security Database . . . 6-17
Adding Entries to the Security Database 6-17
Modifying the Security Database . . . 6-19
Deleting Entries from the Security Database 6-19
Using gsec from a Windows Command Prompt 6-19
Using gsec to Manage Database Alias . . . 6-19
gsec Error Messages . . . 6-20

Chapter 7
Database Configuration and Maintenance

Database Files . . . 7-2
Database File Size . . . 7-2
Dynamic File Sizing . . . 7-2
Database File Preallocations . . . 7-2
isql -extract PREALLOCATE . . . 7-3
GSTAT . . . 7-3
API DPB Parameter . . . 7-3
External Files . . . 7-4
Temporary Files . . . 7-4
File Naming Conventions . . . 7-4
Primary File Specifications . . . 7-5
Secondary File Specifications . . . 7-5
Multifile Databases . . . 7-5
Adding Database Files . . . 7-5
Altering Database File Sizes . . . 7-6
Maximum Number of Files . . . 7-6
Application Considerations . . . 7-6
Reorganizing File Allocation . . . 7-7
Networked File Systems . . . 7-7
On-disk Structure (ODS) . . . 7-7
Read-write and Read-only Databases . . . 7-8
Read-write Databases . . . 7-8
Read-only Databases . . . 7-8
Properties of Read-only Databases . . . 7-8
Making a Database Read-only . . . 7-9
Read-only with Older InterBase Versions 7-9
Creating Databases . . . 7-9
Database Options . . . 7-11
Page Size . . . 7-11
Default Character Set . . . 7-11
SQL Dialect . . . 7-11
Dropping Databases . . . 7-12
Backup File Properties . . . 7-12
Removing Database Backup Files . . . 7-13
Shadowing . . . 7-13
Tasks for Shadowing . . . 7-13
Advantages of Shadowing . . . 7-14
Limitations of Shadowing . . . 7-14
Creating a Shadow . . . 7-15
Creating a Single-file Shadow . . . 7-16
Creating a Multifile Shadow . . . 7-16
Auto Mode and Manual Mode . . . 7-17
Conditional Shadows . . . 7-18
Activating a Shadow . . . 7-18
Dropping a Shadow . . . 7-19
Adding a Shadow File . . . 7-19
Setting Database Properties . . . 7-19
Alias Tab . . . 7-19
General Tab . . . 7-20
Sweep Interval and Automated Housekeeping 7-22
Overview of Sweeping . . . 7-23
Garbage Collection . . . 7-23
Automatic Housekeeping . . . 7-23
Configuring Sweeping . . . 7-24
Setting the Sweep Interval . . . 7-24
Disabling Automatic Sweeping . . . 7-24
Performing an Immediate Database Sweep 7-25
Configuring the Database Cache . . . 7-25
Default Cache Size Per Database . . . 7-26
Default Cache Size Per isql Connection . . . 7-26
Setting Cache Size in Applications . . . 7-26

Default Cache Size Per Server. 7-27
Verifying Cache Size 7-27
Forced Writes vs. Buffered Writes. 7-28
Validation and Repair 7-28
Validating a Database 7-28
Validating a Database Using gfix 7-29
Validating a Database using IBConsole. 7-29
Repairing a Corrupt Database 7-31
Shutting Down and Restarting Databases . . 7-32
Shutting Down a Database. 7-32
Shutdown Timeout Options 7-33
Shutdown Options 7-33
Restarting a Database 7-35
Limbo Transactions 7-35
Recovering Transactions. 7-35
Transaction Tab 7-36
Details Tab 7-37
Viewing the Administration Log 7-37
gfix Command-line Tool 7-38
gfix Error Messages 7-41

Chapter 8
Database Backup and Restore

About InterBase backup and restore options . . 8-1
InterBase backup and restore tools. . . . 8-2
The difference between logical and physical backups. 8-2
Database ownership 8-3
Restoring the ODS 8-3
Performing backups and restores using the gbak command 8-4
General guidelines for using gbak 8-4
Initiating multi- and single-file backups. . . . 8-5
Creating incremental backups 8-7
Incremental backup guidelines 8-8
Executing an incremental backup. 8-8
Over-writing Incremental backups . . . 8-10
Timestamp Changes 8-11
Database parameter blocks used by an incremental backup 8-12
Page Appendix File 8-12
Preallocating database space with gbak 8-13
Using the switch -PR(EALLOCATE) argument 8-13
Restoring a database with gbak 8-14
Using gbak with InterBase Service Manager. . 8-18
The user name and password 8-19
Some backup and restore examples. . . . 8-19
Database backup examples 8-19
Database restore examples 8-20
gbak error messages 8-21
Performing backups and restores using IBConsole. 8-25
Performing a full backup using IBConsole. 8-25
About IBConsole backup options. . . . 8-27
Format 8-27
Metadata Only 8-28
Garbage collection 8-28
Transactions in limbo 8-28
Checksums 8-29
Convert to Tables. 8-29
Verbose Output. 8-29
Transferring databases to servers running different operating systems 8-30
Performing an incremental backup using IBConsole 8-30
Restoring a database using IBConsole . . 8-31
Restore options. 8-34
Page Size. 8-34
Overwrite 8-34
Commit After Each Table 8-35
Create Shadow Files 8-35
Deactivate Indexes 8-35
Validity Conditions 8-36
Use All Space. 8-36
Verbose Output. 8-36

Chapter 9
Journaling and Disaster Recovery

About Journals, Journal Files, and Journal Archives 9-1
How Journaling Works. 9-2
How Journal Archiving Works 9-5
Configuring your System to Use Journaling and Journal Archiving. 9-5
Additional Considerations 9-5
Enabling Journaling and Creating Journal Files 9-6
About Preallocating Journal Space. . . . 9-8
Tips for Determining Journal Rollover Frequency. 9-8
Tips for Determining Checkpoint Intervals 9-9
Displaying Journal Information 9-10
Using IBConsole to Initiate Journaling. . . 9-10
Disabling Journal Files. 9-10
Using Journal Archiving. 9-11

The command that Activates Journal Archiving 9-11
The Command that Archives Journal Files . 9-12
The Command that Performs Subsequent Archive Dumps 9-12
How Often Should you Archive Journal Files? . 9-12
Disabling a Journal Archive 9-12
Using a Journal Archive to Recover a Database . 9-13
Managing Archive Size 9-13
About Archive Sequence Numbers and Archive Sweeping 9-13
Tracking the Archive State 9-14
Journaling Tips and Best Practices 9-15
Designing a Minimal Configuration . . . 9-15
Creating a Sample Journal Archive . . . 9-16

Chapter 10
Database Statistics and Connection Monitoring

Monitoring with System Temporary Tables . . 10-1
Querying System Temporary Tables 10-2
Refreshing the Temporary Tables 10-2
Listing the Temporary Tables 10-3
Security 10-3
Examples 10-3
Updating System Temporary Tables 10-4
Making Global Changes 10-5
Viewing Statistics using IBConsole 10-5
Database Statistics Options 10-6
All Options 10-6
Data Pages 10-7
Database Log 10-7
Header Pages 10-8
Index Pages 10-11
System Relations 10-11
Monitoring Client Connections with IBConsole . . 10-12
The gstat Command-line Tool 10-13
Viewing Lock Statistics 10-16
Retrieving Statistics with isc_database_info() 10-18

Chapter 11
Interactive Query

The IBConsole isql Window 11-1
SQL Input Area 11-2
SQL Output Area 11-2
Status Bar 11-3
isql Menus 11-3
File Menu 11-3
Edit Menu 11-3
Query Menu 11-4
Database Menu 11-5
Transactions Menu 11-5
Windows Menu 11-6
isql Toolbar 11-6
Managing isql Temporary Files 11-8
Executing SQL Statements 11-8
Executing SQL Interactively 11-8
Preparing SQL Statements 11-9
Valid SQL Statements 11-9
Executing a SQL Script File 11-10
Using Batch Updates to Submit Multiple Statements 11-10
Using the Batch Functions in isql 11-12
Committing and Rolling Back Transactions 11-12
Saving isql Input and Output 11-13
Saving SQL Input 11-13
Saving SQL Output 11-13
Changing isql Settings 11-14
Options Tab 11-14
Advanced Tab 11-16
Inspecting Database Objects 11-17
Viewing Object Properties 11-18
Viewing Metadata 11-18
Extracting Metadata 11-20
Extracting Metadata 11-21
Command-line isql Tool 11-22
Invoking isql 11-22
Command-line Options 11-23
Using Warnings 11-24
Examples 11-25
Exiting isql 11-25
Connecting to a Database 11-25
Setting isql Client Dialect 11-26
Transaction Behavior in isql 11-27
Extracting Metadata 11-27
isql Commands 11-28
SHOW Commands 11-29
SET Commands 11-29
Other isql Commands 11-29
Exiting isql 11-29
Error Handling 11-29
isql Command Reference 11-30
BLOBDUMP 11-30
EDIT 11-31

EXIT 11-32
HELP 11-32
INPUT 11-33
OUTPUT 11-34
QUIT 11-35
SET 11-35
SET AUTODDL 11-37
SET BLOBDISPLAY 11-38
SET COUNT 11-39
SET ECHO 11-40
SET LIST 11-41
SET NAMES 11-42
SET PLAN 11-43
SET STATS 11-44
SET TIME 11-45
SHELL 11-46
SHOW CHECK 11-46
SHOW DATABASE 11-47
SHOW DOMAINS 11-48
SHOW EXCEPTIONS 11-48
SHOW FILTERS 11-49
SHOW FUNCTIONS 11-49
SHOW GENERATORS 11-50
SHOW GRANT 11-51
SHOW INDEX 11-51
SHOW PROCEDURES 11-52
SHOW ROLES 11-53
SHOW SYSTEM 11-54
SHOW TABLES 11-54
SHOW TRIGGERS 11-55
SHOW VERSION 11-56
SHOW VIEWS 11-56
Using SQL Scripts 11-57
Creating an isql Script 11-57
Running a SQL Script 11-58
To Run a SQL Script Using IBConsole 11-58
To Run a SQL Script Using the Command-line isql Tool 11-58
Committing Work in a SQL Script 11-59
Adding Comments in an isql Script 11-59

Chapter 12
Database and Server Performance

Introduction 12-1
Hardware Configuration 12-2
Choosing a Processor Speed 12-2
Sizing Memory 12-2
Using High-performance I/O Subsystems . 12-3
Distributing I/O 12-4
Using RAID 12-4
Using Multiple Disks for Database Files 12-4
Using Multiple Disk Controllers 12-5
Making Drives Specialized 12-5
Using High-bandwidth Network Systems . 12-5
Using high-performance Bus 12-6
Useful Links 12-7
Operating System Configuration 12-7
Disabling Screen Savers 12-8
Console Logins 12-8
Sizing a Temporary Directory 12-9
Use a Dedicated Server 12-9
Optimizing Windows for Network Applications . 12-9
Understanding Windows Server Pitfalls . .12-10
Network Configuration 12-10
Choosing a Network Protocol 12-11
NetBEUI 12-11
TCP/IP 12-11
Configuring Hostname Lookups 12-11
Database Properties 12-12
Choosing a Database Page Size 12-12
Setting the Database Page Fill Ratio . . .12-13
Sizing Database Cache Buffers 12-14
Buffering Database Writes 12-15
Database Design Principles 12-15
Defining Indexes 12-16
What is an Index? 12-16
What Queries Use an Index? 12-16
What Queries Don't Use Indexes? . . .12-17
Directional Indexes 12-17
Normalizing Databases 12-17
Choosing Blob Segment Size 12-17
Database Tuning Tasks 12-18
Tuning Indexes 12-18
Rebuilding Indexes 12-18
Recalculating Index Selectivity 12-18
Performing Regular Backups 12-18
Increasing Backup Performance 12-18
Increasing Restore Performance 12-19
Facilitating Garbage Collection 12-19
Application Design Techniques 12-19
Using Transaction Isolation Modes 12-19
Using Correlated Subqueries 12-20
Preparing Parameterized Queries 12-21
Designing Query Optimization Plans . . .12-21
Deferring Index Updates 12-22
Application Development Tools 12-22

InterBase Express™ (IBX) 12-22
IB Objects 12-23
Visual Components 12-23
Understanding Fetch-all Operations . 12-23
TQuery 12-23
TTable 12-24

Appendix A
Migrating to InterBase

Migration Process 13-1
Server and Database Migration 13-1
Client Migration. 13-2
Migration Issues 13-2
InterBase SQL Dialects 13-2
Clients and Databases 13-3
Keywords Used as Identifiers 13-3
Understanding SQL Dialects 13-3
Dialect 1 Clients and Databases 13-3
Dialect 2 Clients. 13-4
Dialect 3 Clients and Databases 13-4
Setting SQL Dialects 13-5
Setting the isql Client Dialect. 13-5
Setting the gpre Dialect 13-6
Setting the Database Dialect 13-6
Features and Dialects 13-7
Features Available in All Dialects 13-7
IBConsole, InterBase's Graphical Interface. 13-7
Read-only Databases 13-7
Altering Column Definitions 13-7
Altering Domain Definitions 13-7
The EXTRACT() Function. 13-7
SQL Warnings 13-7
The Services API, Install API, and Licensing API 13-8
New gbak Functionality 13-8
InterBase Express™ (IBX) 13-8
Features Available Only in Dialect 3 Databases 13-8
Delimited Identifiers 13-8
INT64 Data Storage 13-8
DATE and TIME Datatypes 13-8
New InterBase Keywords 13-9
Delimited Identifiers 13-10
How Double Quotes have Changed . .13-10
DATE, TIME, and TIMESTAMP Datatypes 13-10
Converting TIMESTAMP Columns to DATE or TIME 13-12
Casting Date/time Datatypes 13-13
Adding and Subtracting Datetime Datatypes. 13-15
Using Date/time Datatypes with Aggregate Functions 13-16
Default Clauses. 13-16
Extracting Date and Time Information .13-16
DECIMAL and NUMERIC Datatypes . . .13-18
Compiled Objects 13-19
Generators. 13-20
Miscellaneous Issues 13-20
Migrating Servers and Databases 13-21
"In-place" Server Migration 13-21
Migrating to a New Server 13-22
About InterBase 6 and Later, Dialect 1 Databases 13-23
Migrating Databases to Dialect 3 13-24
Overview. 13-24
Method One: In-place Migration 13-24
Column Defaults and Column Constraints . . 13-27
Unnamed Table Constraints 13-28
About NUMERIC and DECIMAL Datatypes . 13-28
Method Two: Migrating to a New Database . . . 13-30
Migrating Older Databases 13-31
Migrating Clients 13-31
Migrating Data from Other DBMS Products. .13-33

Appendix B
InterBase Limits

Various InterBase Limits 14-2

Tables

1.1 InterBase Features 1-5
3.1 IBConsole Context Menu for a Server Icon . 3-3
3.2 IBConsole Context Menu for a Connected Database Icon. 3-4
3.3 IBConsole Standard Toolbar 3-5
3.4 Server/database Tree Commands 3-6
4.1 DB_ROUTE Table - Structure. 4-6
4.2 DB_ROUTE Table Information 4-7
4.3 Start and Stop InterBase Commands . . 4-11
4.4 Contents of ibconfig. 4-21
5.1 Matrix of Connection Supported Protocols 5-2
5.2 New Parameters 5-17
5.3 Client-side OTW Options 5-18
5.4 Client Server Verification Parameters. . 5-20
5.5 Server-side Configuration Parameters . 5-22
5.6 Using Communication Diagnostics to Diagnose Connection Problems 5-36
6.1 Format of the InterBase Security Database. 6-4
6.2 Summary of gsec Commands 6-17
6.3 Options for gsec 6-18
6.4 gsec Security Error Messages 6-20
7.1 Auto vs. Manual Shadows 7-17
7.2 General Options 7-21
7.3 Validation Options. 7-30
7.4 gfix Options 7-39
7.5 gfix Database Maintenance Error Messages 7-41
8.2 Arguments for gbak 8-5
8.3 Backup Options for gbak 8-6
8.4 Restoring a Database with gbak 8-15
8.5 gbal Restore Options 8-15
8.6 Host_service Syntax for Calling the Service Manager with gbak 8-18
8.7 gbak Backup and Restore Error Messages. 8-21
9.1 CREATE JOURNAL Options 9-6
9.2 RDB$JOURNAL_ARCHIVES Table . . 9-14
10.1 InterBase Temporary System Tables . . 10-2
10.2 Data page Information 10-7
10.3 Header Page Information. 10-9
10.4 Index Pages Information 10-11
10.5 gstat Options 10-13
10.6 iblockpr/gds_lock_print Options 10-17
10.7 Database I/O Statistics Information Items . . 10-18
11.1 Toolbar Buttons for Executing SQL Statements 11-7
11.2 Options Tab of the SQL Options Dialog 11-15
11.3 Advanced Tab of the SQL Options Dialog . . 11-17
11.4 Object Inspector Toolbar Buttons. . . . 11-18
11.5 Metadata Information Items 11-19
11.6 Metadata Extraction Constraints 11-20
11.7 Order of Metadata Extraction. 11-21
11.8 isql Command-line Options. 11-23
11.9 isql Extracting Metadata Arguments . . 11-27
11.10 SQLCODE and Message Summary . . 11-30
11.11 isql Commands 11-30
11.12 SET Statements 11-35
13.1 isql Dialect Precedence 13-5
13.2 Results of Casting to Date/time Datatypes . . 13-13
13.3 Results of Casting to Date/time Datatypes . . 13-14
13.4 Adding and Subtracting Date/time Datatypes 13-15
13.5 Extracting Date and Time Information . 13-17
13.6 Handling quotation marks inside of strings . . 13-25
13.7 Migrating Clients: Summary 13-32
14.1 InterBase Specifications 14-2

Figures

2.1 The License Manager window 2-6
3.1 IBConsole window 3-2
3.2 IBConsole Toolbar 3-5
3.3 IBConsole Tree pane 3-6
3.4 Active Windows dialog 3-8
3.5 Tools dialog 3-8
3.6 Tool Properties dialog 3-9
3.7 Performance monitor, Database activities tab 3-10
4.1 Server Log dialog 4-29
5.1 Local or Remote Server dialog 5-3
5.2 Welcome to IBConsole 5-4
5.3 Local Server Setup dialog 5-4
5.4 Specify Credentials dialog 5-5
5.5 Finish Wizard dialog 5-6
5.6 Remove Server Setup dialog 5-7
5.7 Specify Credentials 5-8
5.8 Finish Wizard dialog 5-9
5.9 Server Login dialog 5-9
5.10 Add Database and Connect dialog . . . 5-11
5.11 Communications Dialog: DB Connection 5-34
5.12 Communications dialog: TCP/IP 5-35
5.13 Communications Dialog: NetBEUI 5-37
6.1 Create Database Dialog 6-7
6.2 User iInformation Dialog 6-13
7.1 Create Database Dialog 7-10
7.2 Backup Alias Properties 7-12
7.3 Database Properties: Alias tab 7-20
7.4 Database Properties: General Tab. . . . 7-21
7.5 Database Validation Dialog 7-29
7.6 Validation Report Dialog. 7-31
7.7 Database Shutdown Dialog 7-32
7.8 Transaction Recovery: Limbo Transactions . . 7-36
7.9 Transaction Recovery: Details 7-37
7.10 Administration Log Dialog 7-38
8.1 Database Backup Dialog 8-26
8.2 Database Backup Options 8-27
8.3 Database Backup Verbose Output . . . 8-30
8.4 Specifying an Incremental Backup . . . 8-31
8.5 Database Restore Dialog 8-32
8.6 Database Restore Options 8-34
8.7 Database Restore Verbose Output . . . 8-37
9.1 The Create Journal Dialog 9-10
10.1 Database Statistics Options 10-5
10.2 Database Statistics Dialog. 10-6
10.3 Database Connections Dialog. 10-12
11.1 The Interactive SQL Editor in IBConsole 11-2
11.2 INSERT Without Batch Updates 11-11
11.3 INSERT With Batch Updates 11-11
11.4 Options Tab of the SQL Options Dialog . 11-14
11.5 Advanced Tab of the SQL Options Dialog. . . 11-16
11.6 Object Inspector 11-17
12.1 Comparing external transfer rate of disk I/O interfaces 12-3
12.2 Comparing bandwidth of network interfaces . 12-6
12.3 Comparing Throughput of Bus Technologies . 12-7

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 for database administrators or system administrators who are responsible for operating and maintaining InterBase database servers. The material is also useful for application developers who wish to understand more about InterBase technology. The guide assumes knowledge of:

- Server operating systems for Windows, Linux, and UNIX
- Networks and network protocols
- Application programming

Topics Covered in this Guide

- Introduction to InterBase features
- Using IBConsole
- Server configuration, startup and shutdown
- Network configuration and troubleshooting guidelines

- Security configuration for InterBase servers, databases, and data; reference for the security configuration tools
- Implementing stronger password protection
- Database configuration and maintenance options; reference for the maintenance tools
- Licensing: license registration tools, available certificates, the contents of the InterBase license file
- Backing up and restoring databases; reference for the backup tools
- Tracking database statistics and connection monitoring
- Interactive query profiling; reference for the interactive query tools
- Performance troubleshooting and tuning guidelines.
- Two appendices covering migration and the limits of a number of InterBase characteristics

InterBase PDF Documentation

InterBase includes the complete documentation set in PDF format. They are accessible from the Start menu on Windows machines and are found in the *Doc* directory on all platforms. If they were not included with the original InterBase installation, you can install them later by running the InterBase install and choosing Custom install, where you can select the document set. You can also access them from the CD-ROM or copy them from the CD-ROM.

About Enhanced Acrobat Reader

The InterBase PDF document set has been indexed for use with Acrobat's Full Text Search, which allows you to search across the entire document set. To take advantage of this feature, you need the enhanced version of Acrobat Reader, rather than the "plain" version. The enhanced Acrobat Reader has a Search button on the toolbar in addition to the usual Find button. This Search button searches across multiple documents and is available only in the enhanced version of Acrobat Reader. The Find button available in the "plain" version of Acrobat Reader searches only a single document at a time.

If you do not already have the enhanced version of Acrobat Reader 5, the English-language version installer is available in the *Documentation/Adobe* directory on the InterBase CD-ROM or download file and in the *<interbase_home>/Doc/Adobe* directory after the InterBase installation.

The enhanced Acrobat Reader is also available for free in many languages from *http://www.adobe.com/products/acrobat/readstep2.html*. In Step 1, choose your desired language and platform. In step 2, be sure to check the “Include the following options…” box.

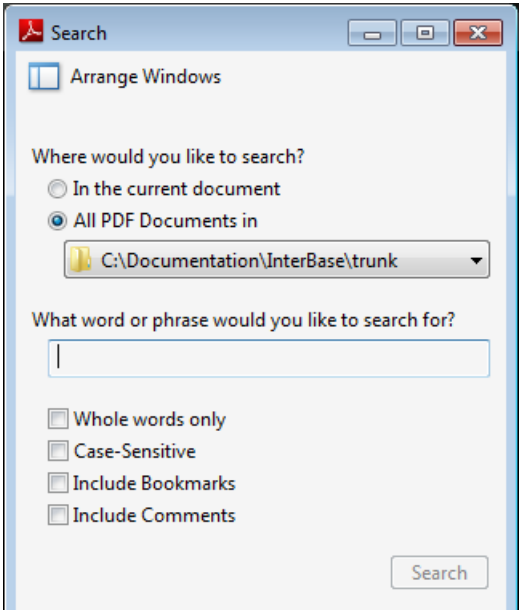
Installing Acrobat

Once you have installed the documentation, you will find the install files for Acrobat Reader With Search will be in the *<interbase_home>/Doc* directory. They are also present on your InterBase CD-ROM or download files. Choose the install file that is appropriate for your platform and follow the prompts. Uninstall any existing version of Acrobat Reader before installing Enhanced Acrobat Reader.

Using Full-Text Search

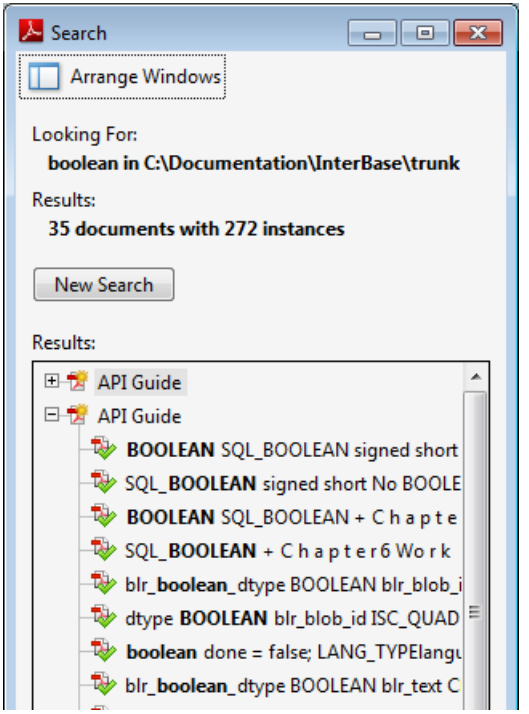
To use full-text searching in the enhanced Acrobat Reader, follow these steps:

1 Click **Edit > Search** on the menu to display the Search dialog.



2 Fill in whichever criteria are useful and meaningful to you. To search all books, select the option of **All PDF Documents in...** and select the directory which contains all the InterBase User Guides.

Acrobat Reader returns a list of books that contain the phrase, ranked by probable relevance.



3 You can sort by: Relevance Ranking, Date Modified, File Name, and Location.

4 Choose the book you want to start looking in to display the first instance.

5 Expand the results and double-click the entry you want to view. The page where the entry is located in the PDF opens in the Adobe Reader.

System Requirements and Server Sizing

InterBase server runs on a variety of platforms, including Microsoft Windows server platforms, Linux, and several UNIX operating systems.

The InterBase server software makes efficient use of system resources on the server node. The server process uses little more than 1.9MB of memory. Typically, each client connection to the server adds approximately 115KB of memory. This varies based on the nature of the client applications and the database design, so the figure is only a baseline for comparison.

The minimal software installation requires disk space ranging from 9MB to 12MB, depending on platform. During operation, InterBase's sorting routine requires additional disk space as scratch space. The amount of space depends on the volume and type of data the server is requested to sort.

The InterBase client also runs on any of these operating systems. In addition, InterBase provides the InterClient Java client interface using the JDBC standard for database connectivity. Client applications written in Java can run on any client platform that supports Java, even if InterBase does not explicitly list it among its supported platforms. Examples include the Macintosh and Internet appliances with embedded Java capabilities.

Terminology: Windows server platforms Throughout this document set, there are references to “Windows server platforms” and “Windows non-server platforms.” The Windows server platforms are Windows Server 2008, Windows 2008 R2 (64-bit), and XP Pro (SP3). Windows non-server platforms are Windows 7 (32-bit and 64-bit), Windows Vista (32-bit and 64-bit), and Windows XP Pro SP3 (32-bit).

Primary InterBase Features

InterBase offers all the benefits of a full-featured RDBMS. The following table lists some of the key InterBase features:

Table 1.1 InterBase Features

Feature	Description
Network protocol support	• All platforms of InterBase support TCP/IP • InterBase servers and clients for Windows support NetBEUI/named pipes
SQL-92 entry-level conformance	ANSI standard SQL, available through an Interactive SQL tool and Embarcadero desktop applications
Simultaneous access to multiple databases	One application can access many databases at the same time
multigenerational 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 records that a client updates, instead of locking an entire database page
Query optimization	Server optimizes queries automatically, or you can manually specify a query plan
Blob datatype and Blob filters	Dynamically sizeable datatypes that can contain unformatted data such as graphics and text
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

Table 1.1 InterBase Features (*continued*)

Feature	Description
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
multidimensional 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; included with the InterBase server license
IBConsole	Windows tool for data definition, query, database backup, restoration, maintenance, and security
isql	Command-line version of the InterBase interactive SQL tool; can be used instead of IBConsole for interactive queries.
Command-line database administrator utilities	Command-line version of the InterBase database administration tools; can be used instead of IBConsole
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 you can use to query standard InterBase example databases on the server
Message file	*interbase.msg*, containing messages presented to the user

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 SQL99 extensions to the SQL standard. These include stored procedures, triggers, SQL roles, and segmented Blob support.

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

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 ***Embedded SQL 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 multigenerational 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 ***Embedded SQL Guide*** for details.

Database Administration

InterBase provides both GUI and command-line tools for managing databases and servers. You can perform database administration on databases residing on Local InterBase or InterBase Server with IBConsole, a Windows application running on a client PC. You can also use command-line database administration utilities on the server.

IBConsole and command-line tools enable the database administrator 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 security later in this chapter, and later chapters describe individual tasks you can accomplish with IBConsole and the 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 InterBase security database (*admin.ib* by default), on the server.

Note InterBase XE implements stronger password protection on InterBase databases. See: **“Implementing Stronger Password Protection”**

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 **"Database User Management."**

Backing Up and Restoring Databases

You can backup and restore a database using IBConsole or command-line **gbak**. 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
- Backing up only a database's metadata to recreate an empty database

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

Maintaining a Database

You can prepare a database for shutdown and perform database maintenance using either IBConsole or the command-line 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 about database maintenance, see **"Database Configuration and Maintenance."**

Viewing Statistics

You can 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 **"Database Statistics and Connection Monitoring."**

About InterBase SuperServer Architecture

InterBase uses SuperServer architecture: a multi-client, multi-threaded implementation of the InterBase server process. SuperServer replaces the Classic implementation used for previous versions of InterBase. SuperServer serves many clients at the same time, using threads instead of separate server processes for each client. Multiple threads share access to a single server process.

Overview of Command-line Tools

For each task that you can perform in IBConsole, 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 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 noninteractive, 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). 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 **"Interactive Query"** for full documentation and reference on **isql** and using **isql** from IBConsole.

gbak

The **gbak** tool provides options for backing up and restoring databases. **gbak** now backs up to multiple files and restores from multiple files, making it unnecessary to use the older **gsplit** command. Only SYSDBA and the owner of a database can back up a database. Any InterBase user defined on the server can restore a database, although the user must be SYSDBA or the database owner in order to restore it *over* an existing database.

Note When you back up and restore databases from IBConsole on Windows platforms, you are accessing this same tool through the IBConsole interface.

See **"Database Backup and Restore"** for full documentation and reference on using **gbak**.

gfix

gfix configures several properties of a database, including:

- 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 access all the functionality of **gfix** through the IBConsole graphical interface. Only SYSDBA and the owner of a database can run **gfix** against that database.

See **"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 IBConsole.

See **"Database User Management"** for full details and reference.

gstat

gstat displays some database statistics related to transaction inventory, data distribution within a database, and index efficiency. You can also view these statistics from IBConsole. You must be SYSDBA or the owner of a database to view its statistics.

See **"Database Statistics and Connection Monitoring"** 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 **“Database Statistics and Connection Monitoring”** 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 **“Using ibmgr to Start and Stop the Server”** for details on using this utility.

Chapter

2

Licensing

This chapter summarizes the licensing provisions and add-ons available for InterBase products. The licensing information in this chapter is not meant to replace or supplant the information in the much more detailed license agreement you receive at the time of purchase. Instead, this chapter summarizes general licensing terms and options.

To activate and use an InterBase product, you must register it when you install it or soon after. For detailed instructions on how to install InterBase products, see the **IBsetup.html** file that you receive upon purchase.

This chapter also provides basic instructions on how to use the **InterBase License Manager** to view existing license information for the products you've purchased, to register those products if you haven't already, and to view additional add-ons licenses.

InterBase License Options

This section summarizes the license provisions and add-on options available for each InterBase product. For the purposes of this chapter, an "add-on" refers to an InterBase feature or option you can purchase to "add-on" to the InterBase product(s) you've already purchased. Each add-on comes with its own license agreement. To purchase an InterBase add-on, see your sales or Value-Added Reseller () representative, or go to the Embarcadero.com website.

To view available InterBase add-ons and licenses, you can use the InterBase License Manager, which installs with your product. For more information, see **"Using the License Manager"**.

Note To distribute the InterBase software to third parties as bundled with your own software application or installed on your hardware, you must contact Embarcadero Technologies and enter into an Original Equipment Manager (OEM) license agreement.

Server Edition

InterBase Server Edition software provides strong, symmetric multiple processing support, and robust database and server monitoring facilities.

The IB Server Edition License allows you to:

- Install Server Edition software on a single computer.
- Make up to four (4) connections to the machine on which the server software is running.
- The provisions for each server license are specific to the single computer for which the server software is licensed. For example, if you purchase two server licenses for 20 users each, you cannot increase the number of licensed users to 40 on a single computer.

Note You can use the same serial number to install and register Server Edition software on a second computer for backup purposes as long as the second computer is not used concurrently with the primary installation.

Add-ons available for the Server Edition

The following add-ons are available for the Server Edition:

Note Add-on licenses continue to be available for XE versions.

- **InterBase Strong Encryption License**

  The InterBase Strong Encryption License enables strong encryption (AES) on the Desktop, Server, and ToGo Editions.

  In the XE3 release, the strong encryption license is built in for the Desktop, Server, and ToGo Editions. So there is longer the need for an add-on license for the "encryption" functionality. It is now automatically part of Server/Desktop/ToGo Edition. However, it is not part of Developer and IBLite and cannot be added to them even as an add-on.

  A strong encryption license is required for each installation that utilizes strong encryption. By default, InterBase allows only the use of weak encryption (DES) if the Strong Encryption License has not been activated. If strong encryption has been activated, you can use either DES or AES. For more information about using InterBase encryption, see the InterBase ***Data Definition Guide***.

- **InterBase Simultaneous User License**

To connect any users to Server Edition software, you must purchase as many Simultaneous User Licenses as you have simultaneous users on that server. For example, to install copies of the server software on ten (10) computers, you must first purchase fifteen (10) Simultaneous User licenses. Each Simultaneous User License allows a single user up to four (4) connections to the Server software.

- **InterBase Unlimited Users License**

  Use this license to allow an unlimited number of users to access the software. To enable your users to connect to the Server Edition software via an unrestricted-access Internet application, you must purchase an Unlimited User License.

- **InterBase Additional CPUs License**

  With the exception of the Unlimited User License, every license certificate you purchase allows you to install and execute the software on one computer with up to eight (8) CPUs for each license. The Additional CPUs license enables all other s/CPUs up to a limit of 32 on the system.

Developer Edition

The Developer Edition License is limited to use of the Server Edition for development purposes only, using solely client applications executing on the same, single computer as the server. This license grants no rights for use in a production environment.

There are no add-ons available for the Developer Server Edition.

Desktop Edition

The InterBase Desktop Edition is an InterBase database server designed to run in a stand-alone environment for use by a single end user. You can deploy the Desktop Edition on a desktop, laptop, or other mobile computer.

The InterBase Desktop Edition license enables you to:

- Install a single copy of the Desktop Edition on a single computer for your internal data processing needs only
- Log into the same computer on which the software is running and make up to eight (8) connections to the InterBase Desktop Edition software.

Add-ons available for the Desktop Edition

The following add-ons are available for the Desktop Edition:

Note Add-on licenses are only available in the XE version and earlier.

- **InterBase Strong Encryption License**

The InterBase Strong Encryption License enables strong encryption (AES) on the Desktop, Server, and ToGo Editions. A strong encryption license is required for each installation that utilizes strong encryption. By default, InterBase allows only the use of weak encryption (DES) if the Strong Encryption License has not been activated. If strong encryption has been activated, you can use either DES or AES. For more information about using InterBase encryption, see the InterBase ***Data Definition Guide***.

- **InterBase Simultaneous Users License**

  For the Desktop Edition, each Simultaneous User License you add provides four (4) additional local connections to Desktop Edition software.

- **InterBase Unlimited Users License**

  Each Unlimited User License is associated with a single Desktop License. Each Unlimited User License allows you to use the Client Software to connect an unlimited number of users to the particular instance of the server for which it is licensed.

- **InterBase Additional CPUs License**

  InterBase Server Edition license comes with 8 cores licensed initially. If your system has more cores you want utilized by InterBase, the "Additional CPUs license" license should be installed. Each license enables InterBase to use an additional 8 cores on your system. Purchase and install as many "Additional CPUs license" as you need. InterBase 32-bit Edition can only use a maximum of 32 cores; 64-bit Edition can use a maximum of 64 cores.

ToGo Edition

The InterBase ToGo Edition is a small, portable version of the Desktop Edition, and is designed to run in a stand-alone environment. The ToGo Edition is available on the following platforms:

Windows

- Microsoft Windows Vista
- Microsoft Windows 8
- Microsoft Windows 7 (32-bit and 64-bit)
- Microsoft Windows XP (SP2)
- Microsoft Windows Server 2003, 2008
- Microsoft Windows Server 2008 R2 (64-bit)
- Microsoft Windows Server 2012
- Microsoft Windows 2000 (SP4)

Mac

- Apple MAC OS X

iOS

- iPhone 4 and 5 or newer running iOS 5.1 or newer
- iPod 4 or 5 or newer running iOS 5.1 or newer
- iPad 2 and 3 or newer running iOS 5.1 or newer
- Mountain Lion, Lion and their latest supported XCode versions. XCode 4.3 for iOS 5.1, 4.5 for iOS 6, 4.6 for 6.1

Android

- Android 2.3.3 and above; specifically Android OS versions Gingerbread, Ice Cream Sandwich, and Jelly Bean. Android devices running earlier versions of the OS are not supported.

The ToGo Edition does not contain all of the options, such as IBConsole, that are available in the Desktop Edition. For information on how to use the ToGo edition, see the InterBase ***Developer's Guide***.

The InterBase ToGo license enables the purchaser to:

- Deploy applications that are embedded with the InterBase ToGo engine (DLL's).
- Deploy ToGo Edition software on a standalone computer for use by a single end user.

Add-ons available for the ToGo Edition

Note Add-on licenses are only available in the XE version and earlier.

The following add-ons are available for the ToGo Edition:

- **InterBase Strong Encryption License**

  The InterBase Strong Encryption License enables strong encryption (AES) on the Desktop, Server, and ToGo Editions. A strong encryption license is required for each installation that utilizes strong encryption. By default, InterBase allows only the use of weak encryption (DES) if the Strong Encryption License has not been activated. If strong encryption has been activated, you can use either DES or AES. For more information about using InterBase encryption, see the InterBase ***Data Definition Guide***.

Using the License Manager

A separate **License Manager** tool installs with the Desktop, ToGo, and Server Editions. to view existing license information for the products you've purchased, to register those products if you haven't already, and to view additional licenses for add-ons.

If your Linux or Solaris environment does not support the GUI installer, you can use the command-line tool to register additional options and licenses. For information on how to do so, see the **IBsetup.html** file.

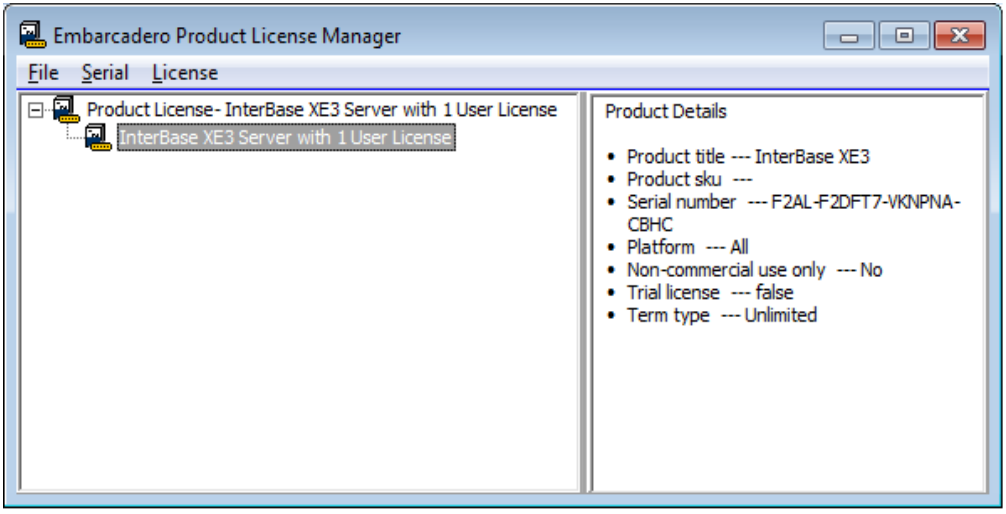
Important In versions of InterBase prior to 2007, you could access an older version of the License Manager from IBConsole. Because the IBConsole version of License Manager does not contain up-to-date licensing information and options, including add-ons, we recommend that you use only the separate InterBase License Manager tool to purchase new licenses.

Accessing the License Manager

To access the separate InterBase License Manager tool, do the following:

1 From the Start menu, select Programs>Embarcadero InterBase XE>License Manager. The License Manager window appears, as shown in Figure 2.1.

Figure 2.1 The License Manager window



2 To view and select the add-ons available for the InterBase product you are using, click on the Serial menu and select **Add**.

3 When the **Add Serial Number** dialog opens, type in your serial number and click **OK**. The License Manager displays licensing information and allows you to register the product you purchased if you have not done so already.

Chapter

3

IBConsole: The InterBase Interface

InterBase provides an intuitive graphical user interface, called IBConsole, with which you can perform every task necessary to configure and maintain an InterBase server, to create and administer databases on the server, and to execute interactive SQL (isql). IBConsole enables you to:

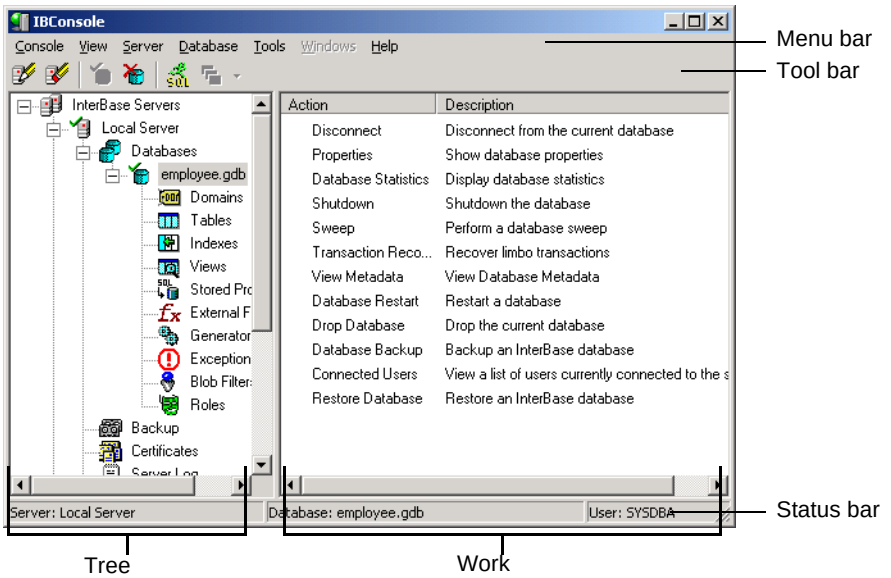
- Manage server security
- Back up and restore a database
- Monitor database performance
- View server statistics
- Perform database maintenance, including:
  - Validating the integrity of a database
  - Sweeping a database
  - Recovering transactions that are “in limbo”

IBConsole runs on Windows, but can manage databases on any server on the local network.

Starting IBConsole

To start IBConsole, choose IBConsole from the Start|Programs|InterBase menu. The IBConsole window opens:

Figure 3.1 IBConsole window



Elements in the IBConsole dialog:

- **Menu bar** Commands for performing database administration tasks.
- **Tool bar** Shortcut buttons for menu commands.
- **Tree pane** Displays a hierarchy of registered servers and databases.
- **Work pane** Displays specific information or allows you to perform activities, depending on what item is currently selected in the Tree pane.
- **Status bar** Shows the current server, user login, and selected database.

IBConsole Menus

The IBConsole menus are the basic way to perform tasks with IBConsole. There are seven pull-down menus.

- **Console menu** enables you to exit from IBConsole.
- **View menu** enables you to indicate whether or not IBConsole displays system and temporary tables and dependencies and to change the display and appearance of items listed in the Work pane.

- **Server menu** enables you to log in to and log out of a server, diagnose a server connection problem, manage user security, view the server log file, and view server properties. For more information, see **“Connecting to Servers and Databases”** .
- **Database menu** enables you connect to and disconnect from a database, create and drop a database, view database metadata, monitor database performance and activities, view a list of users connected to the database, view and set database properties, and perform database maintenance, validation, and transaction recovery. For more information, see **“Connecting to Servers and Databases”**
- **Tools menu** enables you to add custom tools to the Tools menu and start the interactive SQL window. The interactive SQL window has its own set of menus, which are discussed in **"Interactive Query”**. This menu includes a ‘License Manager’ command that you can use to register a server.

Note If you are adding the very first certificate from IBConsole to an older database, the Add Certificate pop-up menu is shown. This will add an old style certificate. However, as of InterBase 2007, certificate information is not displayed, so it will be hidden, after the server version is determined.

- **Windows menu** enables you to view a list of active IBConsole windows and to manage them. See **“Switching Between IBConsole Windows”** for more information.
- **Help menu** enables you to access both IBConsole on-line help and InterBase on-line help.

Context Menus

IBConsole also enables you to perform certain tasks with context sensitive popup menus called *context menus*. Tables 3.1 and 3.2 are examples of context menus.

When you right-click a server icon, a context menu is displayed listing actions that can be performed on the selected server.

Table 3.1 IBConsole Context Menu for a Server Icon

Popup command	Description
Login	Login to the selected server.
Logout	Logout from the current server.
Add Certificate	Add certificate ID/keys for servers connected to older databases (InterBase 7.x or earlier). Use License Manager for newer databases (InterBase 7.x or later).
User Security	Authorize users on the current server.
License Manager	Add, edit, and verify license information. You can also launch the License Manager from the Tools menu.

Table 3.1 IBConsole Context Menu for a Server Icon (*continued*)

Popup command	Description
View Logfile	Display the server log for the current server.
Diagnose Connection	Display database and network protocol communication diagnostics.
Database Alias	Add or delete a database alias.
Properties	View and update server information for the current server.

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

Table 3.2 IBConsole Context Menu for a Connected Database Icon

Popup command	Description
Disconnect	Disconnect from the current database.
Maintenance	Perform maintenance tasks including: view database statistics, shutdown, database restart, sweep, and transaction recovery.
Backup/Restore	Back up or restore a database to a device or file.
Copy Database	Copy the database to a different database and/or server.
Performance Monitor	View database activities including queries, transactions, trigger actions, tables, and memory status. For more information, see “Monitoring Database Performance”.
View Metadata	View the metadata for the selected database.
Connected Users	Displays a list of users connected to the database.
Properties	View database information, adjust the database sweep interval, set the SQL dialect and access mode, and enable forced writes.
Decrypt Database	SYSDSO’s can use this option to encrypt the database password.

IBConsole Toolbar

A toolbar is a row of buttons that are shortcuts for menu commands. The following table describes each toolbar button in detail.

Figure 3.2 IBConsole Toolbar

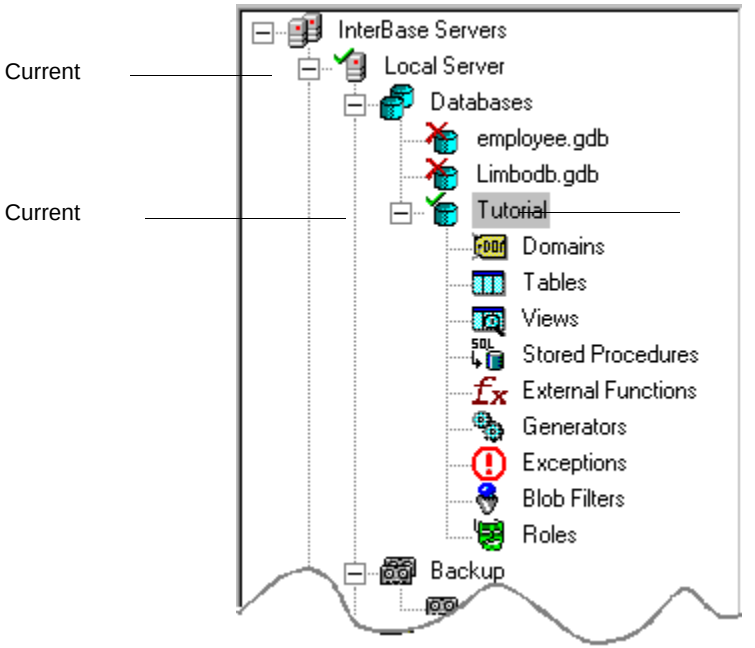
Table 3.3 IBConsole Standard Toolbar

Button	Description
	Register server: opens the register server dialog, enabling you to register and login to a local or remote server.
	Un-register server: enables you to un-register a local or remote server. This automatically disconnects a database on the server and logout from the server. See **"Un-registering a Server"** for more information.
	Database connect: Connects to the highlighted database using the user name and password for the current server connection. See **"Connecting to a Database"** for more information.
	Database disconnect: enables you to disconnect a database on the current server. See **"Disconnecting a Database"** for more information.
	Launch SQL: opens the interactive SQL window, which is discussed in detail in **"Interactive Query"**.

Tree Pane

When you open the IBConsole window, you must register and log in to a local or remote server and then register and connect to the server's databases to display the Tree pane. See **"Connecting to Servers and Databases"** to learn how to register and connect servers and databases.

Figure 3.3 IBConsole Tree pane



Navigating the server/database hierarchy is achieved by expanding and retracting nodes (or branches) that have sub-details or attributes. This is accomplished by a number of methods, outlined in Table 3.4.

To expand or retract the server/database tree in the Tree pane:

Table 3.4 Server/database Tree Commands

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

In an expanded tree, click a database name to highlight it. The highlighted database is the one on which IBConsole operates, referred to as the *current database*. The *current server* is the server on which the current database resides.

The hierarchy displayed in the Tree pane of Figure 3.3 is an example of a fully expanded tree.

- Expanding the InterBase Servers branch displays a list of registered servers.
- Expanding a connected server branch displays a list of server attributes, including Databases, Backup, Users, and the Server Log.
- Clicking on the Database branch displays a list of registered databases on the current server.
- Clicking on Server Log displays the “View Logfile” action in the Work pane.
- Expanding a connected database branch displays a list of database attributes, including Domains, Tables, Views, Stored Procedures, External Functions, Generators, Exceptions, Blob Filters, and Roles.
- Expanding Backup shows a list of backup aliases.

Work Pane

Depending on what item has been selected in the Tree pane, the Work pane gives specific information or enables you to execute certain tasks.

To display a list of backup aliases for the current server, click the Backup icon.

To display a list of users defined on the server, click the Users icon.

To display information about a database attribute, click the database attribute icon.

To display information about a database object in a viewer, click the Work Pane icon for the object (for example, a table name).

See **“Viewing Metadata”** for more information.

Standard Text Display Window

The standard text display window is used to monitor database backup and restoration, to display database statistics and to view server and administration logs.

The standard text display window contains a menu bar, a toolbar with icons for often-used menu commands, and a scrolling text display area. **Figure 8.3, “Database Backup Verbose Output”** is an example of the standard text display window.

Elements in a standard text display window:

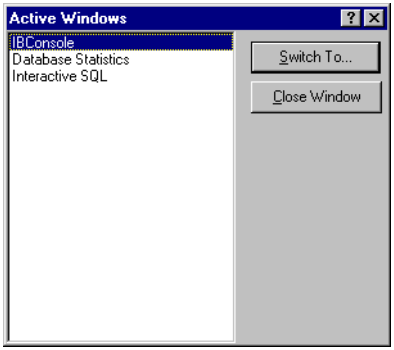
- **Menu bar** The File menu enables you to save the contents of the window and Exit from the window. The Edit menu enables you to copy selected text in the window to the clipboard, select all text in the window, cut and paste text, and find a specified word or phrase within the displayed text.
- **Toolbar** Save and Copy toolbar buttons enable you to save the contents of the text display window as well as copy selected text to the clipboard.

- **Status bar** Shows the cursor location, given by line and column, within the text display window.

Switching Between IBConsole Windows

Use the Active Windows dialog to switch between IBConsole windows, or to close specific windows. To access the Active Windows dialog, click on the Windows menu. The dialog appears:

Figure 3.4 Active Windows dialog

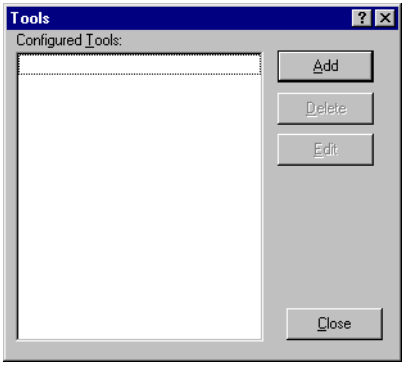


- To switch to a different IBConsole window, select it and click the Switch To button.
- To close a window, select it and click the Close window button.

Managing Custom Tools in IBConsole

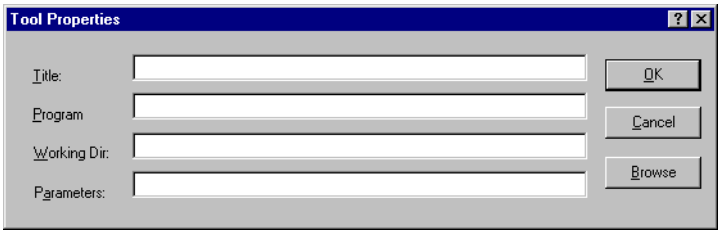
Use the Tools dialog to add, edit, and delete custom tools for the IBConsole interface. To access the Tools dialog, select Tools|Configure Tools from the IBConsole menu. The Tools dialog is displayed:

Figure 3.5 Tools dialog



- To delete a tool, select it and click Delete.
- To modify a tool, select it and click Edit. Change the relevant fields in the **Tool Properties dialog**.
- To add a tool, click Add. The Tool Properties dialog appears.

Figure 3.6 Tool Properties dialog



To add a custom tool:

- Enter the name of your utility in the Title field. This is the name that will be displayed on the Tools menu. Use an ampersand (&) to specify an accelerator key for the menu item. Conflicting accelerator keys are automatically resolved. If you do not specify an accelerator key, one will be chosen automatically.
- Enter the path and the executable to be launched in the Program field.
- Enter the working directory for your utility in the Working Dir field. If no working directory is specified, then it defaults to the current directory.
- Enter any other parameters needed to run your utility in the Parameters field.

Monitoring Database Performance

You can use the IBConsole performance monitor to view a database's status and activities including:

- A summary of database threads, procedures, transactions and memory use
- View memory use by graph, pool or heap
- List all the tables and views in the database
- Display a list of all recent transactions
- View a list of all recent procedures and triggers
- Display the interbase log file and statistics list

Displaying the Performance Monitor

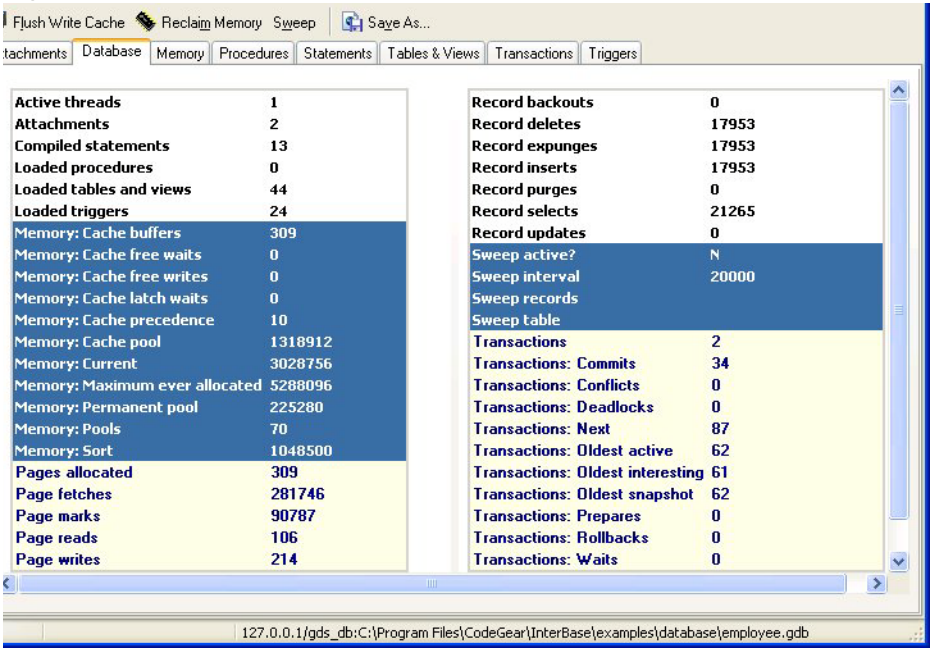
To display the performance monitor:

1. In the left pane, right-click on a database name, and select **Performance Monitor** from the context menu. The Performance Monitor appears.

2 The **Performance Monitor** contains the following tabs: Attachments, Database, Memory, Procedures, Statements, Tables & Views, Transactions, and Triggers. Click on a tab to view the information displayed on that tab.

Figure 3.7 displays the Database tab. Notice the options (Flush Write Cache, Reclaim Memory Space and others) that appear above the tab name. You can click on an option to perform that action on the database. Different options appear for each tab.

Figure 3.7 Performance monitor, Database activities tab



Chapter

4

Server Configuration

This chapter describes the operation and configuration of the InterBase server process, including the following topics:

- Configuring Server Properties
- Using InterBase Manager to Start and Stop InterBase
- Starting and Stopping the InterBase Server on UNIX
- Example initialization script installation on Linux
- Using Environment Variables
- Managing Temporary Files
- Configuring Parameters in ibconfig
- Viewing the Server Log File

Configuring Server Properties

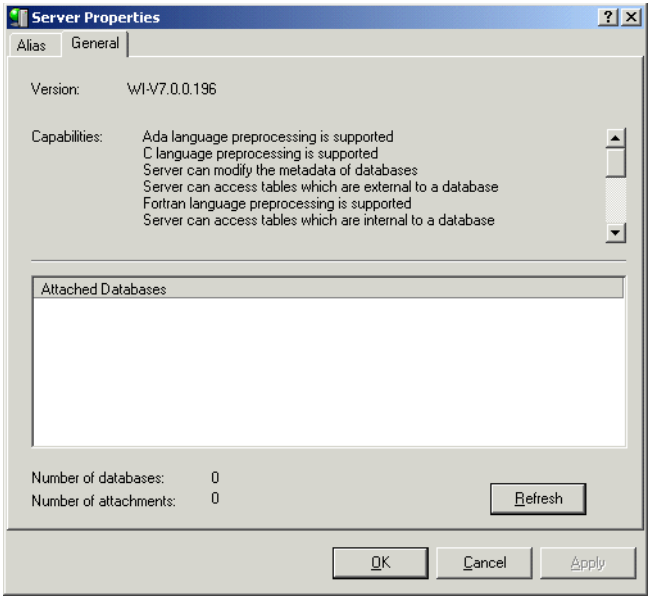
You can use InterBase Manager to change database cache size of client map size. The InterBase Guardian Server Properties dialog enables you to display and configure these server settings. To access InterBase Guardian, right-click the InterBase Guardian icon in the System Tray. You can access the Server Properties dialog by any of the following methods:

- Select a server (or any branch under the server hierarchy) in the Tree pane and choose Server|Server Properties.
- Select a server in the Tree pane and click Server Properties in the Work pane.
- Right-click a server in the Tree pane and choose Server Properties from the context menu.

The Server Properties dialog contains two tabs, Alias and General.

The General Tab

The General tab of the Server Properties dialog is where you can view such server settings as the version, capabilities, number of databases, and number of attachments. You cannot edit the information displayed on this tab.



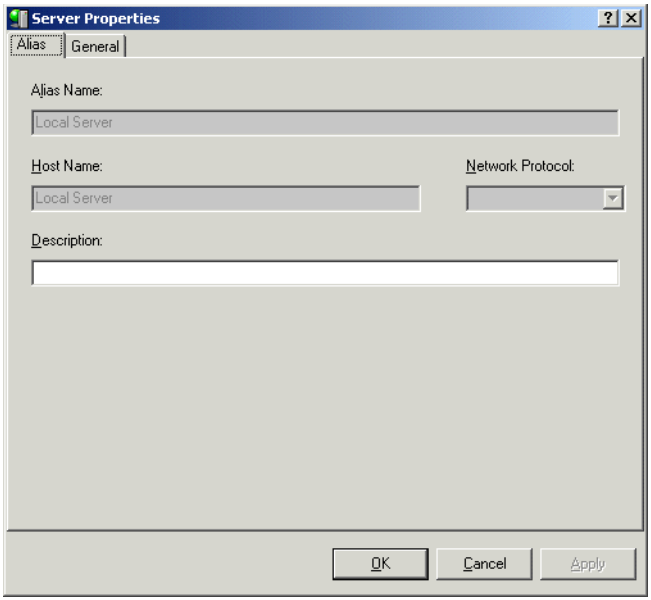
The server properties displayed are:

- **Version**: displays the version number for the InterBase Server.
- **Capabilities**: displays support capabilities for the InterBase Server.
- **Attached databases**: displays the path and filename for each attached database
- **Number of databases**: displays the total number of databases in the InterBase Server.
- **Number of attachments**: displays the total number attachments to the InterBase Server.

You cannot update the information displayed on the General tab; however, you can click Refresh at any time to retrieve the current server property information. If you need to view or configure server settings, click the IB Settings tab.

The Alias Tab

On the Alias tab, you can inspect the host name and network protocol for the server. You can inspect and change the Alias name and description.



- **Alias Name**: the name of the server as it appears in the Tree pane. This setting is editable.
- **Host Name**: the name of the host server. This is determined at the time you create the server connection and cannot be changed in this dialog.
- **Network Protocol**: the protocol that the server is using to communicate. This is determined at the time you create the server connection and cannot be changed in this dialog.
- **Description**: any additional information you wish to add about the server. This field is optional and editable.

Multi-Instance

InterBase 2007 (equivalent to version 8.0) allows multiple instances of InterBase servers to run simultaneously. In versions previous to 7.5, multiple instances of the InterBase server could not be run on the same machine. In these earlier versions, when an application utilized one version of InterBase, another application that utilized another version of InterBase could not be run. With InterBase 7.5, and later versions, you can run one previous version (major release) of InterBase, i.e. InterBase 6.x while you are running a newer version simultaneously.

Windows Server Setup

Start the server as an application with the following switches on a Windows machine.

ibserver ***-a -p*** *service_name* ***-i*** *interbase_env_variable*

The service_name is the entry contained in the services file pointing to the port number which the InterBase server should bind to. Below is an example of a part of the file from the <system directory>\drivers\etc\services file.

```
gds_db 3050/tcp #default interbase port number
ib__a 3051/tcp # A's interbase port number
```

The InterBase environment variable or the -i switch is used for local connections. These values determines which InterBase server a client on the same machine will connect to. The InterBase environment variable for a client and server's -i switch must match to have a successful connection. So if InterBase server is started with the setting:

ibserver ***-a -p*** *ib__a* ***-i*** *C:\Embarcadero\InterBase*

The InterBase server will accept remote connections on the TCP/IP port number 3051 as the service ib__a is set to port 3051. The local connections will be accepted from client on the same machine who have their InterBase environment variable set to C:\Program Files\interbase.

Older versions of InterBase servers (pre-7.5) can still run using the default setting. These pre-7.5 InterBase servers will accept remote connections on TCP/IP port number 3050. The local connections will be accepted when the client uses a pre-7.5 interbase client library.

We recommend using the -i switch to set the local InterBase variable for the server. The order in which InterBase server looks for the InterBase environment variable is as follow; Command line argument '-i', InterBase environment variable setting, InterBase Registry key setting, Server's current directory.

Accessing Remote Databases

Client Side Settings

In order to connect to a specific InterBase server on a machine you need to identify the server you want to connect to.

Remote Servers

In order to access the database database_name.ib located in the directory database_dir. On a remote machine remote_host accepting connections on a port number specified by a service_name on the client machine. The database name specified in isql, the client API or any InterBase driver should be remote_host/ service_name:/database_dir/database_name.ib

Assume that a remote client application wants to access windows server running on a machine called remote_host running 2 servers with the example configuration specified above. The client machine will need to have the same service name set as the server, so the services file will need to have these entries

```
gds_db 3050/tcp #default interbase port number
ib__a 3051/tcp # A's interbase port number
```

In order to access an InterBase server running on the 3051 port number, use the following database connection string (through isql or through the API) remote_host/3051:c:\database_dir\ib80test.ib.

For older clients specify the service name which is bound to the port number on which the older server is listening e.g. remote_host/gds_db:c:\database_dir\ib71test.ib

Accessing Local Databases

Note Windows platform only.

In order to access a database on a local InterBase server, InterBase depends on the InterBase Environment variable to identify the server to be connected to. A pre-7.5 InterBase server running will be connected to if no server with the InterBase environment variable setting is running.

In order to access an older server make sure that your application uses the older gds32.dll. To access a older server using a 7.5 InterBase client library make sure your InterBase environment variable is set to a empty string "".

Applications can also pass in the information regarding the InterBase server they want to attach to as part of the InterBase database parameter block (isc_dpb parameter). Setting the isc_dpb_client_interbase_var followed by the length and the location of the InterBase server will allow the user to specify the InterBase server to be used. The following code demonstrates how a client should set the dpb parameter to attach to a InterBase server running with the InterBase environment variable set to "c:/interbase"

```
#include <ibase.h>
…
char dpb_buffer[256], dpb;
short dpb_length;
char *ib_server = "c:/interbase";
char *str = "employee.ib";
isc_db_handle db1;
ISC_STATUS status_vector[20];
/* construct the dpb, specifing the IB server to attach to
*/
dpb = dpb_buffer;
*dpb++ = isc_dbp_version;
*dpb++ = isc_dpb_client_interbase_var;
*dpb++ = strlen(ib_server);
strcpy (dpb, ib_server);
/* set other dpb parameters */
```

```
…
/* set the dpb_length as usual */
dpb_length = dpb - dpb_buffer;
/* finally call isc_attach or create with the dpb parameters
*/
isc_attach_database(status_vector, strlen(str), str, &db1,
dpb_length, dpb_buffer);
```

Automatic Rerouting of Databases

Once multiple instance of InterBase servers are running on a machine simultaneously, this feature will allow setups where some database connections can be re-routed to a different InterBase server. The user will have to manually start the different instance of InterBase as an application or service.

Server Side Setup

In order to setup simultaneous InterBase servers on a machine follow the instructions specified above. Once these machines are set up, and running, follow the instructions below to setup and use the DB_ROUTE database table in the ADMIN.IB.

The structure of the DB_ROUTE table is as follows:

Table 4.1 DB_ROUTE Table - Structure

Column Name	Data type	Length	Description
DB_NAME	CHAR	1024	Complete name of the database to be rerouted, including the path of the database on the server. e.g. Server_Name:DATABASE_DIR\EMPLOYEE.Ib
SERVICE_NAME	CHAR	30	Service name to look up in the services file for the port number to be re-routed to. The look up takes place at the server side, the client is only aware of the port number and not the service name. e.g. ib__a
ACTIVATE_ROUTE	BOOLEAN		Set to true if this re-routing is active, flase it this re-routing is disabled.

The service name that is entered in the set DB_ROUTE table must exist in the services file:

```
gds_db 3050/tcp #default interbase port number
ib__a 3051/tcp # A's interbase port number
```

Client Side Settings

No client side settings are required. In order to access the database database_name.ib located in the directory database_dir. On a remote machine remote_host accepting connections on a default port number. The database name specified in the client API or any InterBase driver would be REMOTE_HOST:DATABASE_DIR\DATABASE_NAME.IB.

In order to setup the database server AGNI so that it can re-route in coming connections, for the database c:\smistry\employee.ib to an older server running on port number specified by the service ib__a. The ADMIN.IB database on server AGNI will need the following row of information in DB_ROUTE table.

Table 4.2 DB_ROUTE Table Information

DB_ROUTE Column Name	Value
DB_NAME	Server_Name:DATABASE_DIR/ EMPLOYEE.IB
SERVICE_NAME	ib__a
ACTIVATE_ROUTE	TRUE

Since the DB_ROUTE is a regular InterBase table in the ADMIN.IB database you can use regular SQL to enter, modify or remove database re-routing from information.

Startup Parameters

To accommodate multiple instances of InterBase running on the same machine the InterBase Guardian and Server now have label names as part of their Service names.

- InterBase %Service Name% Guardian, i.e. InterBase 8.0 server1 Guardian
- InterBase (%Service Name%) Server, i.e. InterBase 8.0 server1 Server

The InterBase Server and Guardian have two new command line arguments:

```
-i %interbase_INSTALL_DIR% -p %SERVICE NAME%
```

(Command line arguments are called start parameters as far as starting the applications are concerned)

If you write to the Microsoft auto run registry key entry you will need to do the same to the

/SOFTWARE/MICROSOFT/WINDOWS/CURRENTVERSION/RUN REGISTRY KEY SETTING TOO.

Currently the registry key is InterBaseGuardian

Change this to InterBaseGuardian%SERVICE NAME%

The value of this registry key is currently %interbase_INSTALL_DIR%/bin/ibguard ***-a***

Change this to %interbase_INSTALL_DIR%/bin/ibguard -a -i %interbase_INSTALL_DIR% -p %SERVICE NAME%

SMP Support

InterBase provides symmetric multiprocessor (SMP) support for both clients and servers. Older versions of InterBase ran on SMP systems safely by allowing only a single processor at a time to execute within the InterBase components. Current versions of InterBase exploit SMP hardware by running InterBase threads on all processors simultaneously for increased throughput and performance.

Important When you purchase a single server license, you acquire the right to use a single processor. You must purchase one additional license for each additional processor that you wish to use.

On Windows platforms, the CPU_AFFINITY setting in the *ibconfig* configuration file specifies which processors of a multiprocessor system InterBase should use. The default setting, in effect when CPU_AFFINITY is commented out, is to use as many processors as licensing permits. See **"Expanded Processor Control: CPU_AFFINITY"** below for how to specify a subset of processors to use.

Expanded Processor Control: CPU_AFFINITY

On Windows multiprocessor platforms, you can specify which processors InterBase should use by adding the CPU_AFFINITY parameter to the *ibconfig* file. This setting is useful whenever the number of licensed processors is less than the number of actual processors present.

Important Note that when you purchase a single server license, you acquire the right to use a single processor. You must purchase one additional license for each additional processor that you wish to use.

The CPU_AFFINITY parameter populates a bit vector in which each bit represents a processor on the system on which the threads are allowed to run. The maximum number of processors depends on the operating system. To specify which processors should be used, give CPU_AFFINITY a decimal value that corresponds to the binary value that results from setting a bit for each desired machine. For example, to use processors 2 and 3, set CPU_AFFINITY to 6:

```
CPU_AFFINITY6
```

To use these processors	CPU_AFFINITY value	Array setting
1	1	0 0 1
2	2	0 1 0
1 and 2	3	0 1 1
3	4	1 0 0
2 and 3	6	1 1 0
1, 2, and 3	7	1 1 1

ibconfig Parameter: MAX_THREADS

Setting the MAX_THREADS parameter in *ibconfig* controls the maximum number of threads that can be active at one time within the InterBase engine. The default setting is 100:

```
MAX_THREADS100
```

This configuration parameter controls the number of active threads in the engine. The ideal setting for this number depends partly on the nature of the work being performed by your clients. If you have many clients performing very similar tasks, you may want to lower the MAX_THREADS setting to reduce contention. On the other hand, if simultaneous activity is highly diverse, setting this to a higher value may increase throughput.

Note that this setting does not affect the maximum possible threads that can be created by the InterBase server but only the number that can be active in the engine at one time.

ibconfig Parameter: THREAD_STACK_SIZE_MB 2

This parameter controls the stack size of various threads in InterBase. The value is in multiple of MBs per thread. The valid range is 2MB to 32MB. If it is set beyond the range, the value defaults to 2MB.

You should not have to change this parameter for normal use of InterBase. In extremely rare cases of abnormal termination of the process, the reason might be thread stack space constraints due to high levels of recursive calls (of stored procedures and such). Feel free to increase this value then.

- The default setting on 32-bit Editions is 2 (2 MB)
- The default setting on 64-bit Editions is 4 (4 MB)

Hyper-threading Support on Intel Processors

InterBase can support hyper-threading on Intel processors that support logical processors using Intel's hyperthreading technology. To enable this support in the InterBase server, you must make a setting in the InterBase configuration file, *ibconfig*. If you are running the InterBase server on a machine with hyper-threaded processors, edit the ENABLE_HYPERTHREADING parameter in the configuration file. By default, this parameter is set to zero. Set the value to 1 to allow the InterBase server to use hyperthreaded processors.

Using InterBase Manager to Start and Stop InterBase

The InterBase Server and InterBase Guardian must be started before you enable database connections. On Windows platforms, you can use the InterBase Manager to start and stop the InterBase Server and Guardian. In previous versions of InterBase the InterBase Manager is a Windows Control Panel applet. Now the InterBase Manager is an application installed for each instance of the InterBase Server installed. To start the InterBase Manager, choose Start|Programs|<InterBase install directory>. You can use InterBase Manager to do the following:

- Choose the server startup mode: whether to start the InterBase server manually, or have it start automatically at system boot time
- Change the path to the server: if you click the Change option, you can browse and select a different directory
- Change how InterBase Server operates. By default, InterBase runs automatically as a service on Windows platforms, though it is possible (but not recommended) to run it as an application.

Note To start InterBase Server as an application from a command prompt or in a batch file, invoke InterBase Guardian with the following options:

ibguard ***-a -p*** *service_name* ***-i*** *interbase_env_variable*

Options Commands are:

Table 4.3 Start and Stop InterBase Commands

Command/ option	Description
-a	Start as application
-i	Environment variable; identifies the Server location for clients that want to connect locally to the Server
-p	Port number; where the *service_name* is the entry in the services file pointing to the port number where InterBase Server listens for connection requests from clients

InterBase Guardian starts the server, and places an icon in the System Tray.

- **Start** InterBase Server and InterBase Guardian, via a Start/Stop button. Click **Start** in the InterBase Manager Status area to start InterBase Server (and InterBase Guardian). The server status changes, and an InterBase Guardian icon appears in the system tray. Once you have started the InterBase Server, you can exit InterBase Manager, and both InterBase Server and InterBase Guardian will continue to run. The InterBase Guardian icon remains in the System Tray until you stop the server.
- **Stop** InterBase Server and InterBase Guardian, via a Start/Stop button. Click **Stop** in the InterBase Manager Status area to stop InterBase Server (and InterBase Guardian). Or, right-click the InterBase Guardian icon in the System Tray and choose **Shutdown**.

Starting and Stopping the InterBase Server on UNIX

The following sections describe how to start and stop the InterBase server on UNIX.

Using ibmgr to Start and Stop the Server

The InterBase Server process **ibserver** runs as a daemon on UNIX systems. Use **ibmgr** to start and stop the server process. To use **ibmgr**, you must be logged on to the server machine.

Syntax ibmgr -*command* [-*option* [*parameter*] ...]

or

ibmgr u
```
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 on to the machine on which the server is running to use **ibmgr**.

InterBase may be started as the root user or the interbase user. To correctly start InterBase under the interbase user account take the following steps:

1. Create the interbase user.
2. Give the interbase user read and write access to //run.
3. cd to /opt/interbase.
4. chown -R interbase * to make the interbase user the owner of all files in the interbase directory tree.
5. Start InterBase using ibmgr and the InterBase daemon will start without error.

Note The *ibmgr32.exe* file that is present in older 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, port and user
user *user_name*	Supplies SYSDBA
password *password*	Supplies SYSDBA password
help	Prints help text
quit	Quits prompt mode

Starting the Server

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

ibmgr ***-start -p*** *service_name*

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.” If the -p option is not specified, the default of gds_db is used.

Stopping the Server

Note For safety, make sure all databases have been disconnected before you stop the server.

The command switches **-user** and **-password** can 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 -password password
```

or

```
ibmgr u
IBMGR> shut -password password
```

or

```
imbgr u
IBMGR> password password
IBMGR> shut
```

Note The **-shut** option 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 **“Shutting Down and Restarting Databases”**.

Starting the Server Automatically

To configure a UNIX server to start the InterBase Server automatically at server host boot-time, you must 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 daemon

# Set these environment variables if and only if they are
not set.
: ${InterBase:=/usr/interbase}
# 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

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.
  $InterBase/bin/ibmgr -shut -user SYSDBA -password password
}

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 Linux

1. Log in as root.
   ```
   $ su
   ```
2. Enter the Linux example script (given below) into the initialization script directory.
   ```
   # cp ibserver.sh /etc/rc.d/init.d/ibserver.sh
   # chmod 700 /etc/rc.d/init.d/ibserver.sh
   ```
3. Link the initialization script into the *rc* directories for the appropriate run levels for starting the InterBase server.

```
  # ln -s /etc/rc.d/init.d/ibserver.sh /etc/rc.d/rc0.d/
S85ibserver
```

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

```
  # ln -s /etc/rc.d/init.d/ibserver.sh /etc/rc.d/rc0.d/
K30ibserver
```

5 Make sure you have host equivalence:

```
  # touch /etc/gds_hosts.equiv
  # echo "+" >> /etc/gds_hosts.equiv
```

6 Make sure you don't have an **inetd** entry for InterBase Classic:

```
  # echo -e "/gds_db/s/^/#/\nwq" | ed /etc/inetd.conf
  # killall -HUP inetd
```

Example Linux initialization script

```
#!/bin/sh
# ibserver.sh script - Start/stop the InterBase daemon
# Set these environment variables if and only if they are
not set.
: ${InterBase:=/usr/interbase}
# 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

ibserver_start() {
  # This example assumes the InterBase server is
  # being started as user "InterBase".
  su - InterBase -c "$InterBase/bin/ibmgr -start -forever"
  RETVAL=$?
  [ $RETVAL -eq 0 ] && touch //lock/subsys/ibserver
}

ibserver_stop() {
  # No need to su.
  $InterBase/bin/ibmgr -shut -user SYSDBA -password password
  RETVAL=$?
  [ $RETVAL -eq 0 ] && rm -f //lock/subsys/ibserver
}

if [ ! -d "$InterBase" ] ; then
  echo "$0: cannot find InterBase installed at $InterBase"
>&2
  exit 1
fi
if [ ! -x "$InterBase/bin/ibmgr" ] ; then
```

```
  echo "$0: cannot find the InterBase SuperServer manager as

            $InterBase/bin/ibmgr" >&2
if [ ! -x "$InterBase/bin/gds_inet_server" ] ; then
  echo "$0: this is InterBase Classic; use inetd instead of
            ibserver daemon" >&2
  fi
  exit1
fi

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

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

'restart')
  ibserver_stop ; ibserver_start
  echo "InterBase restarted" ;;
'restart_msg')
  echo 'InterBase Server restarting...\c' ;;

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

exit 0
```

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 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.
```

Using Environment Variables

This section describes the environment variables that InterBase recognizes. When defining environment variables, keep these rules in mind:

- Environment variables must be in the scope of the **ibserver** process.
- On Windows, define environment variables as *system variables* in the Windows Control Panel|System dialog.
- On UNIX, the easiest way to define environment variables is to add their definitions to the system-wide default shell profile.

ISC_USER and ISC_PASSWORD

If you do not provide a user name and password when you connect to a database or when you run utilities such as **gbak**, **gstat**, and **gfix**, InterBase looks to see if the ISC_USER and ISC_PASSWORD environment variables are set; if they are, InterBase uses that user and password as the InterBase user.

Although setting these environment variables is convenient, do not do so if security is at all an issue.

The INTERBASE Environment Variables

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 install directory. The INTERBASE environment variable is used on Windows for local connections. The INTERBASE environment variable is used by the client to identify the local instance of InterBase Server to attach to.

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"**.

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.

IB_PROTOCOL
The IB_PROTOCOL is used to specify which port you want the InterBase Server to use during runtime. It is also used by the InterBase Manager to identify which InterBase Server to manage. It is used by the InterBase clients to identify the instance of InterBase server to connect to.

Important The environment variables must be in the scope of the **ibserver** process. On Windows platforms, define the variables as *system variables* in the 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.

UNIX and Linux Host Equivalence

On UNIX and Linux machines, you must provide a host equivalence for localhost, since even local connections go through TCP/IP. To do this, place a line in the */etc/hosts* file:

```
127.0.0.1  localhost
```

If your local machine has additional names, include them in the line:

```
127.0.0.1  localhost  mymachine_name
```

Managing Temporary Files

InterBase 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 **isql** client creates 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

You should make sure to have plenty of free space available for temporary sorting operations. The maximum amount of temporary space InterBase needs might be larger than the database itself in some cases.

Temporary sort files are always located on the server where the database is hosted; you should specify temporary directories on disk drives that are physically local to the server (not on mapped drives or network mounted file systems).

There are two ways to specify directories for sort files:

- You can add an entry to the *$InterBase/ibconfig* file to enable directory and space definition for sort files. The syntax is:

  ```
  TMP_DIRECTORY size "pathname"
  ```

  Important The pathname **must** be in double quotes, or the config file will fail.

  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 5,000,000 bytes, then specify *dir2* with 10,000,000 bytes, followed by *dir1* with 2,000,000 bytes, InterBase uses *dir1* until it reaches the 5,000,000 limit, then uses *dir2* until it has filled the 10,000,000 bytes allocated there, and then returns to *dir1* where it has another 2,000,000 bytes available. Below are the *ibconfig* entries that describe this configuration:

  ```
  TMP_DIRECTORY 5000000 "C:\dir1"
  TMP_DIRECTORY 10000000 "D:\dir2"
  TMP_DIRECTORY 2000000 "C:\dir1"
  ```

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

If you specify temporary directories in *ibconfig*, the server uses those values for the sort files and ignores the server environment variable values. If you don't specify configuration of temporary directories in *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, default to the */tmp* directory (UNIX) or *C:\temp* (Windows)

Configuring Parameters in *ibconfig*

You specify configuration parameters for InterBase Server by entering their names and values in the configuration file, *ibconfig*. Entries are in the form:

```
parameter <whitespace> value
```

- *parameter* is a string that contains no white space and names a property of the server being configured.
- *value* is a number or string that is the configuration of the specified property.

Each line in *ibconfig* is limited to 80 characters, including the parameter name and any whitespace.

If you specify a value of zero or less for a parameter, the server ignores the setting and uses the default value for that parameter. However, there is no upper limit applied to these parameters.

The server reports the values for each of these parameters in the *InterBase.log* file on startup.

When a parameter is commented out in the *ibconfig* file, the server uses the default value.

The following is a summary of allowable entries in *ibconfig*:

Table 4.4 Contents of *ibconfig*

Parameter	Description
ADMIN_DB	• Platforms: All • Specifies the name of the InterBase security database • Needed only if the security database is not named *admin.ib* • The security database must always be in the InterBase home directory
APPDATA_DIRECTORY	• The directory path MUST be enclosed in double quotes. • The choice entered may vary based on the underlying OS platform version. • It is application for all platforms. • It specifies a directory where InterBase data files (requiring read-write) access are located. This includes database files admin.ib, examples/ sub-folder, ext/ sub-folder, secure/ sub-folder to support SSL/OTW configuration, license/ sub-folder to deliver licenses required for InterBase runtime, and, other temporary files InterBase creates to support its runtime engine (.lck and .env files, interbase.log etc.) • This has been introduced to support Windows UAC driven requirements when installing under "Program Files" area, but can also be applied to isolate InterBase "writable" files into a separate location on other supported Editions and/or OS platforms.
CPU_AFFINITY	• Platform: Windows only. • 32-bit values only allowed. Only the first 32 processors can be set • On a SMP system, sets which processors can be used by the InterBase server. The value is taken from a bit vector in which each bit represents a processor. Thus, to use only the first processor, the value is 1. To use both the first and second processors, the value is 3. To use the second and third, the value is 6. • The default is to use all processors (when entry is commented out).
CONNECTION_TIMEOUT	• Platforms: All • Seconds to wait before concluding an attempt to connect has failed • Default: 180

Table 4.4 Contents of *ibconfig*

Parameter	Description
DATABASE_CACHE_PAGES	• Platforms: All • Server-wide default for the number of database pages to allocate in memory per database • Can be overridden by clients • See **"Configuring the Database Cache"** for more information • Default: 2048
DATABASE_ODS_VERSION	• Platforms: All • Version: starting in InterBase XE7 • The database server/engine will automatically create/restore databases to this major ODS version number, if specified. • Valid ODS major versions are in the range of 13 to 16 • Default major ODS version is the latest version supported by the product • Default: 16
DEADLOCK_TIMEOUT	• Platforms: All • Seconds before an ungranted lock causes a scan to check for deadlocks • Default: 10
DUMMY_PACKET_INTERVAL	• Platforms: All • Seconds to wait on a silent client connection before the server sends dummy packets to request acknowledgment • The default value of 0 will not send any dummy keepalive packets to the client from the server. • Versions of InterBase before 7.1 used to have this set at 60 seconds, by default. This has now been modified to have a default of 0 (zero) seconds due to a memory leak bug in one of the Windows system drivers for socket connections. • Alternatively, you can set the interval value for a specific client by using the isc_dpb_dummy_packet_interval DPB parameter while doing a connection.
ENABLE_HYPERTHREADING	• Platforms: Windows, for InterBase 32-bit Edition(s) only • Specifies whether Intel Hyper-threading technology enabled logical processors should be enabled • Valid values are: 1 (enable), 0 (disable) • Default: 0 (disable)

Table 4.4 Contents of *ibconfig*

Parameter	Description
EXTERNAL_FILE_DIRECTORY	• Platforms: All • The default directory for an external table's file is <interbase_home>/ext. • Using this parameter, you can list additional directories where external files can reside. • For security reasons, do not put other files in this directory. • Directory path MUST be enclosed in double quotes. For e.g. EXTERNAL_FILE_DIRECTORY "C:\Temp" EXTERNAL_FILE_DIRECTORY "/tmpdir" • List multiple entries one per line; directories are used in the order specified.
EXTERNAL_FUNCTION_DIRECTORY	• Platforms: All • The default directory for UDF libraries is <interbase_home>/UDF. • Using this parameter, you can list additional directories where UDF libraries should be loaded from. • For security reasons, do not put other files in this directory. • Directory path MUST be enclosed in double quotes. For e.g. EXTERNAL_FUNCTION_DIRECTORY "C:\Temp" EXTERNAL_FUNCTION_DIRECTORY "/tmpdir" • List multiple entries one per line; directories are used in the order specified.
LOCK_ACQUIRE_SPINS	• Platforms: All • Number of spins during a busy wait on the lock table mutex • Relevant only on SMP machines • Default: 0
LOCK_HASH_SLOTS	• Platforms: All • 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

Table 4.4 Contents of *ibconfig*

Parameter	Description
MAX_DB_VIRMEM_USE	• Platforms: Windows • Define an upper percentage limit of how much of the available virtual memory InterBase can use for its large memory allocations like those of database cache etc. This is set as a percentage of total virtual memory available to the process. • This can be used to set the limit to a lower level if you observe that other critical memory allocations such as file I/O buffers, file handles, socket buffers etc. need the user space. • Valid range that can be set is between 50 and 100 • Please note that setting this to 100 indicates that InterBase can use up all available virtual memory for its database cache etc. This is not recommended. For e.g. setting this to 90 (the default), indicates that on a 2GB virtual address space, InterBase process will only use up 1.8GB of the available virtual memory for database cache allocations; the rest will be available for other critical memory allocations.
MAX_THREADS	• Platforms: All • Controls the maximum number of threads that can be active at one time within the InterBase engine. The listed value applies to a system with multiple licensed CPUs. For a single CPU system, the value defaults to 1 which eliminates the synchronization overhead required by multiple CPUs. • Default: 1000000
MAX_ASSISTANTS	• Platforms: All • Controls the maximum number of threads that can be active assisting other threads with their tasks. This number should be less than the number of CPUs on which InterBase can run. • Default: 1
MEMORY_RECLAMATION	• Platforms: All • Number of seconds between attempts to return unused memory to OS. • This parameter enables better co-existence with neighboring processes by not monopolizing memory. It cannot guarantee that other processes will behave in kind. • 0 = disable memory reclamation • 1 = memory reclamation on attach of first database • 2 = memory reclamation on detach of last database • 3 = 1 + 2 above • > 15 = number of seconds between memory reclamations • Default: 300

Table 4.4 Contents of *ibconfig*

Parameter	Description
PAGE_CACHE_EXPANSION	• Platforms: All • Number of MB of memory to expand page buffer cache until upper limit is reached. The upper limit can be a database-specific setting or the DATABASE_CACHE_PAGES parameter. The page buffer cache is expanded when there are no more free buffers, which occurs when all buffers have been loaded with a database page. • This parameter is used to smooth the memory allocation of page buffers to match demand generated by client load. • 0 = allocate all page buffers on first database attach • > 0 = expand this many MBs at a time until all page buffers allocated • Default: 256
SERVER_CLIENT_MAPPING	• Platform: Windows • Size in bytes of one client's portion of the memory mapped file used for interprocess communication • Valid values are 1024, 2048, 4096, 8192 • Default: 4096
SERVER_PRIORITY_CLASS	• Platforms: Windows • Priority of the InterBase service on Windows • The value 1 is low priority, 2 is high priority • Relevant only on Windows NT/2000/XP • Default is 1
SERVER_WORKING_SIZE_MAX	• Platforms: Windows • Threshold above which the OS is requested to swap out all memory • Default is 0 (system-determined)
SERVER_WORKING_SIZE_MIN	• Platforms: Windows • Threshold below which the OS is requested to swap out no memory • Default is 0 (system-determined)
SOLARIS_BOUND_THREADS	• Platform: Solaris • When set to 1 each user-level thread is bound to a LWP thread. • The default of 0 creates user-level threads unbound and causes a user-level scheduler to map them to available LWPs. • Default: 0

Table 4.4 Contents of *ibconfig*

Parameter	Description
SOLARIS_SYNC_SCOPE	• Platform: Solaris • When set to 1 threads use process-level synchronization variables. • The default of 0 causes thread-level synchronization variables to be used. • Default: 0
SORTMEM_BUFFER_SIZE	• Platforms: All • Specifies the size of a sort buffer in memory • Versions of InterBase before 7.1 used to have a static value of approx. 128 KB. This is now configurable per server using this parameter. The default value is just about 1 MB. • Setting this to a higher value will enable better performance in large sort-merge queries. • Default: 1048500
SQL_COMPILER_RECURSION	• Platforms: All • Specifies the call depth that the recursive descent parsing algorithm will try during SQL compilation phase of statement preparation. If the call depth would exceed this limit than a stack overflow is declared and returned as an error. Note that this is an artificial stack overflow detection and not a hardware detected stack overflow. • Default: 2000
STARTING_TRANSACTION_ID	• Platforms: All • Version: starting in InterBase XE7 • ODS version: 16+ • The database server/engine will automatically create/restore databases with this transaction ID as the starting number. • If you want to restore your database(s) to test with very high transaction ID numbers, to evaluate 48-bit transaction ID support, set this value and restart InterBase. • Valid values include any value >= 0, that can be represented in a 48-bit Integer data type. • Default starting transaction ID on database create/restore is 0 • Default: 0
SWEEP_QUANTUM	• Platform: All • Specifies the maximum number of records that a garbage collector thread or a sweeper thread is allowed to work before yielding control back to the worker threads. • Default:10

Table 4.4 Contents of *ibconfig*

Parameter	Description
SWEEP_YIELD_TIME	• Platforms: All • Specifies the time, in milliseconds, the sweeper or garbage collector thread sleeps. • Default:1 millisecond
TCP_REMOTE_BUFFER	• Platforms: All • TCP/IP buffer size for send and receive buffers. This applies to both client and server programs. • Valid range is 1448 to 32768 • Default: 8192 bytes
TMP_DIRECTORY	• Platforms: All • Directory to use for storing temporary files (such as sort files). • Default is the value of environment variables INTERBASE_TMP or TMP, in that order; otherwise /tmp on UNIX or C:\temp on Windows NT/2000/XP. • If you have lots of space in your default temporary folders as above, then there is no need to mention any further directory listings here. • Specify directory path and number of bytes available in the directory. • Directory path MUST be enclosed in double quotes. • Format for entry is as follows. TMP_DIRECTORY <MaxBytes> <AbsoluteDirectoryPath> MaxBytes should be a number indicating maximum space in bytes to be used in the directory, of type 32-bit unsigned integer. Valid range: 1MB - 4GB (in bytes) • For e.g. TMP_DIRECTORY 500400300 "C:\Temp" TMP_DIRECTORY 1500400300 "/tmpdir" • List multiple entries, one per line; directories are used in the order specified.
USER_QUANTUM	• Platforms: All • Specifies the maximum number of records that a worker thread (thread running an user query) is allowed to work before yielding control back to other threads. • Default: 1000
V4_EVENT_MEMSIZE	• Platforms: All • Bytes of shared memory allocated for event manager. • Default: 32768

Table 4.4 Contents of *ibconfig*

Parameter	Description
V4_LOCK_GRANT_ORDER	• Platforms: All • 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; can result in lock request starvation. • Default: 1
V4_LOCK_MEM_SIZE	• Platforms: All • Bytes of shared memory allocated for lock manager. • Default is 98304 on Linux and Solaris, 256K on Windows.
V4_LOCK_SEM_COUNT	• Platforms: All • Number of semaphores for interprocess communication (Classic architecture only). • Default: 32
V4_LOCK_SIGNAL	• Platforms: All • UNIX signal to use for interprocess communication • (Classic architecture only). • Default: 16
V4_SOLARIS_STALL_VALUE	• Platform: Solaris • Number of seconds a server process waits before retrying for the lock table mutex. • Default is 60

Viewing the Server Log File

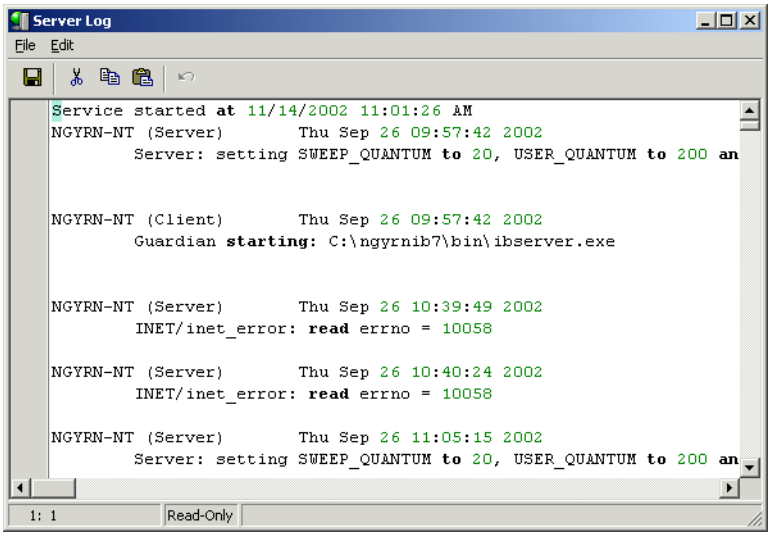
InterBase Server logs diagnostic messages in the file *InterBase.log* in the InterBase install directory. Any messages generated by **ibserver** are sent to this file. This can be an important source of diagnostic information if your server is having configuration problems.

Refer to the *Language Reference* for a list of error messages that can appear in this file.

IBConsole displays this log file in a standard text display window. To display the Server Log dialog:

- Select a server and expand it if it is not already expanded, click Server Log and then double-click View Logfile in the Work pane.
- Right-click a server in the Tree pane and choose View Logfile from the context menu.
- Select a server and then choose View Logfile from the Server menu.

Figure 4.1 Server Log dialog



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 **“Standard Text Display Window”.**

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, encrypting the data that passes between servers, and network troubleshooting tips.

Network Protocols

InterBase supports TCP/IP for all combinations of client and server platforms. Additionally, InterBase supports NetBEUI on Windows server platforms and for all Windows clients, and a local connection mode (involving inter-process communication but no network interface) for Windows 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.

In the following table, Windows non-server platforms are Windows 7 (32-bit and 64-bit), Windows Vista (32-bit and 64-bit), Windows XP Pro (32-bit). Windows servers are Windows 2008 and Windows 2008 R2 (64-bit).

Table 5.1 Matrix of Connection Supported Protocols

	InterBase server platform		
Client platform	**Windows non-server**	**Windows server**	**UNIX**
Windows non-server	TCP/IP, Local	TCP/IP, NetBEUI	TCP/IP
Windows server	TCP/IP	TCP/IP, NetBEUI, Local	TCP/IP
UNIX/Linux	TCP/IP	TCP/IP	TCP/IP

Connecting to Servers and Databases

Before performing any database administration tasks, you must first register and log in to a server. Once you log in, you can register and connect to databases residing on the server. You can switch context from one connected database to another by selecting the desired database from the IBConsole Tree pane. The selected database in the Tree pane is referred to as the *current database*. The selected server or the server where the current database resides is referred to as the *current server*.

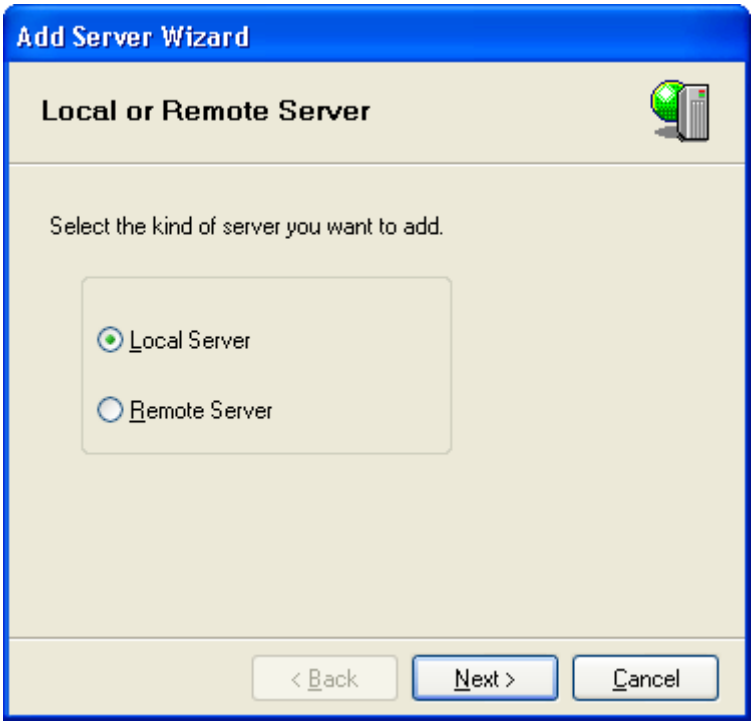
Adding a Server

You can access the Add Server Wizard in IBConsole by one of the following methods:

- Choose Server|Add or click the Add a New InterBase server toolbar button.
- Double-click InterBase Servers in the Tree pane.
- Right-click InterBase Servers and choose Add from the context menu.

When you add a server and there is no local server yet added, the following dialog appears:

Figure 5.1 Local or Remote Server dialog

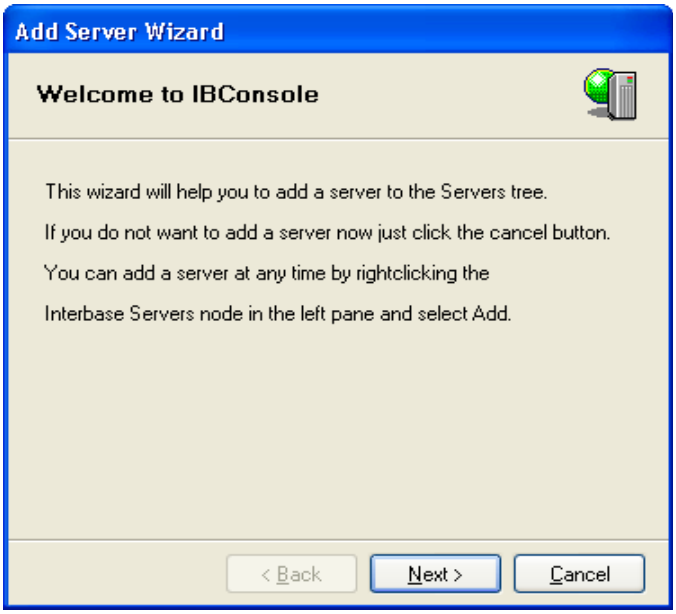


If a local server already exists, the Remote Server Setup dialog opens. For more information, see: **"To add a local server"** and **"To add a remote server".**

To add a local server

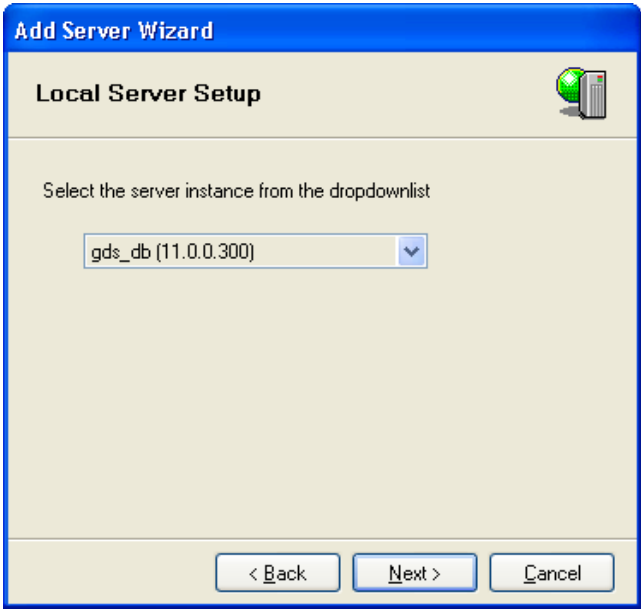
1 Once you have clicked Server|Add, click **Next** in the Welcome dialog to advance to the Local Server Setup panel.

Figure 5.2 Welcome to IBConsole



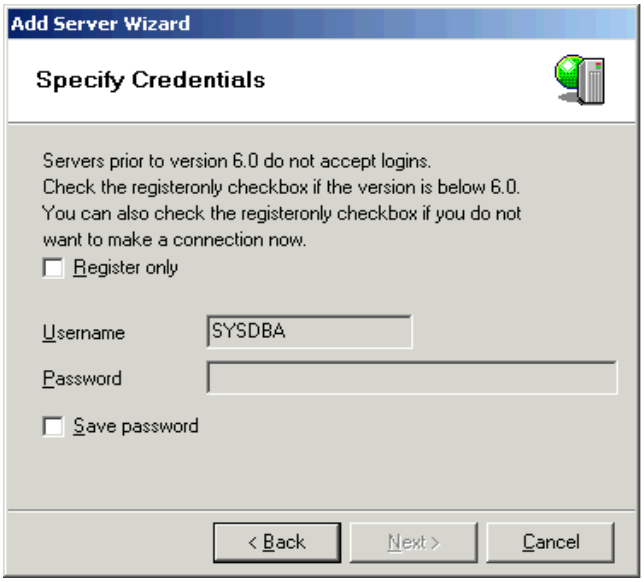
2 Select a server instance from the drop down list.

Figure 5.3 Local Server Setup dialog



3 Click Next and Specify Credentials opens.

Figure 5.4 Specify Credentials dialog



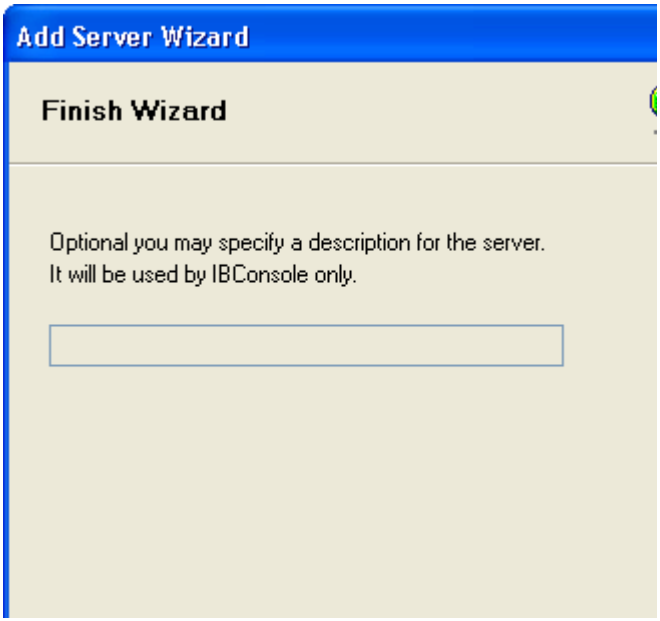
4 You can choose to just add the server (without logging in) or you can choose to add and connect to the server simultaneously.

- If you want to just add the server you can ignore the Login Information and click Next.
- If you want to add and connect to the server simultaneously, enter a username and password in the corresponding text fields and click Next.

Note The usernames and passwords must be the InterBase usernames and passwords stored in the InterBase security database (*admin.ib* by default) on the server.

5 Click Next to advance to Finish Wizard. Here you have the option to enter a description name for the server.

Figure 5.5 Finish Wizard dialog



6 Click Finish and once a server is added, IBConsole displays it in the Tree pane.

To add a remote server

1 Choose Server|Add or click the Add a New InterBase server toolbar button and the Remove Server Setup dialog appears.

Figure 5.6 Remove Server Setup dialog

2 Browse the name of the server.

3 Specify the instance name (the default is gds_db)

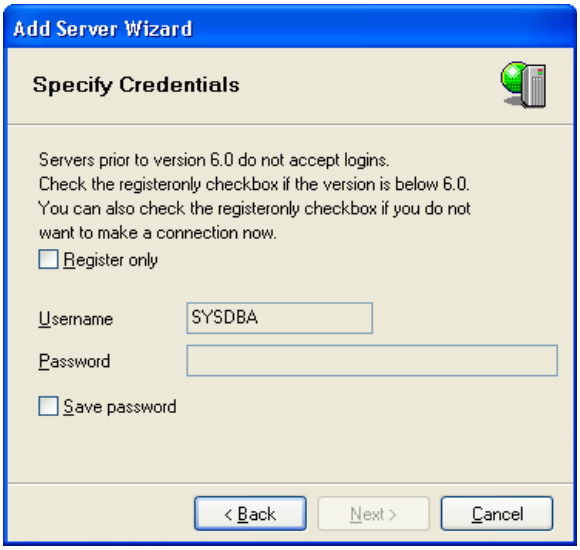
The InterBase server name is the name of the database server machine. There is not a specific name for the InterBase server process itself. For example, if the server is running on the NT server “venus”, you enter this name in the Server Name text field.

4 Enter an alias name in the Optional Specify an Aliasname text field.

5 The network protocol you select can be TCP/IP on any platform. On Windows, it can also be NetBEUI or local. Protocols are valid only when they are supported by both the client and the server.

6 Click Next and Specify Credentials opens.

Figure 5.7 Specify Credentials



7 You can choose to just add the server (without logging in) or you can choose to add and connect to the server simultaneously.

- If you want to just add the server you can ignore the Login Information and click Next.
- If you want to add and connect to the server simultaneously, enter a username and password in the corresponding text fields and click Next.

Note The usernames and passwords must be the InterBase usernames and passwords stored in the InterBase security database (*admin.ib* by default) on the server.

8 Click Next to the Finish Wizard dialog. Here you have the option to enter a description name for the server.

Figure 5.8 Finish Wizard dialog

9 Click Finish and once a server is added, IBConsole displays it in the Tree pane.

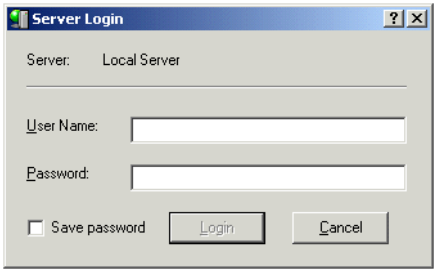
Logging in to a Server

You can access the Server Login dialog in IBConsole by one of the following methods:

- In the Tree pane, select a registered server that is not already logged in. Choose Server|Login or double-click Login in the Work pane.
- In the Tree pane, double-click a registered server that is not already logged in.
- In the Tree pane, right-click a registered server that is not already logged in and choose Login from the context menu.

The Server Login dialog appears:

Figure 5.9 Server Login dialog



To log in to a server

1. Verify that the server displayed in the Server field is correct.
2. Enter a username and password in the corresponding text fields. For convenience, IBConsole defaults the UserName text field to the last username that was used to log in (successfully or unsuccessfully).

   The usernames and passwords must be the InterBase usernames and passwords that are stored in the InterBase security database (*admin.ib* by default) on the server.

   The username is significant to 31 bytes and is not case-sensitive. The 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 username and password are verified against records in the security database. If a matching record is found, the login succeeds.
3. Click Login to log in to the server.

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 IBConsole"**.

Logging Out from a Server

Logging out from a server automatically disconnects all databases but does not un-register any databases on the server.

You can log out from a server in IBConsole by one of the following methods:

- Select a connected server in the Tree pane (you can also select any branch under the desired server hierarchy) and choose Server|Logout.
- Select a connected server in the Tree pane and double-click Logout in the Work pane.
- Right-click a connected server in the Tree pane and choose Logout from the context menu.

A confirmation dialog asks you to confirm that you wish to close the connection to the selected server. Click Yes if you want to log out from the server, otherwise click No.

Un-registering a Server

You can un-register a disconnected server in IBConsole by one of the following methods:

- Select a server in the Tree pane and choose Server|Un-register or click the Unregister Server toolbar button

- Select a server in the Tree pane and double-click Un-register Server in the Work pane.
- Right-click a server in the Tree pane and choose Un-register from the context menu.

A confirmation dialog asks you to confirm that you wish to un-register the selected server. Click Yes if you want to un-register the server, otherwise click No.

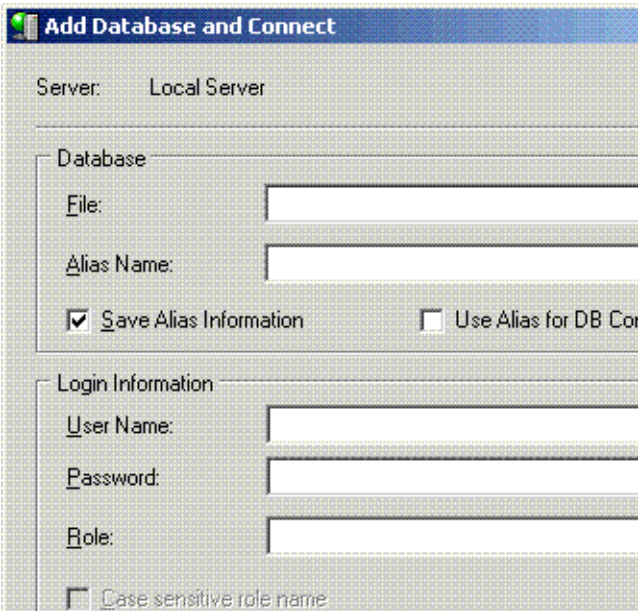
Note Un-registering a server removes that server from the Tree pane and automatically logs you out of the current server as well as disconnects and un-registers any databases on the server.

Adding a Database

You can access the Add Database and Connect dialog in IBConsole by one of the following methods:

- Choose Database|Add.
- Expand a connected server branch. Right-click Databases in the Tree pane and choose Add from the context menu.
- Select a disconnected database in the Tree pane and double-click Add in the work pane, or right-click the database and choose Add from the context menu.

Figure 5.10 Add Database and Connect dialog



To Add a Database

1. Make sure the server displayed in the Server field is correct.
2. Enter the database filename, including the path where the file is located, in the File text field. For databases that reside on the local server, you also have the option of clicking the Browse button to locate the file you want. The Browse button is disabled for all remote servers.
3. Type an alias name for the database in the Alias Name text field. This is the name that will appear in the IBConsole window. If you omit this step, the alias defaults to the filename that you select in step 2.
4. Check the Save Alias Information check box if you wish to permanently register the database. This saves the database alias name in the Windows registry.
5. Check the Use alias DB Connect check if you have specified an Alias Name and you want to connect using it instead of using the File entry.
6. You also have the option to save your password so it is automatically supplied each time you log in.
7. At this point you can choose to just add the database without connecting, or you can choose to add and connect to the database simultaneously.

   If you only want to add the database, ignore the Login Information and click OK.
8. If you want to add and connect a database simultaneously, type the username, password and optional role and default character set for the database in the corresponding text fields and click OK.

   If you want to connect using a role, specify the role in the Role text field. This is optional. Connecting using 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. For more information on roles, refer to**"SQL Roles"**.

Once you add a database, it appears in the Tree pane.

Connecting to a Database

IBConsole provides two methods for connecting to a database. The first method is a quick connect using the username and password that were supplied with the login to the server to instantaneously connect the database. The second method allows you to connect to the database using a different username and password by accessing the Database Connect dialog.

Connect

If you want to perform an automatic connect, using the username and password supplied for the server login to instantaneously connect the database, you can do so by one of the following methods:

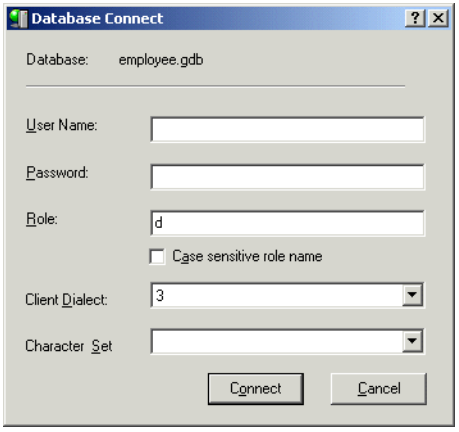
- Select a disconnected database in the Tree pane. Choose Database|Connect, choose Connect in the Work pane, or click on the Database Connect toolbar button.
- Right-click a disconnected database in the Tree pane and choose Connect from the context menu.
- Double-click a disconnected database in the Tree pane.

Once you connect to a database, the database tree expands to display the database hierarchy.

Connect As

If you want to access the Connect Database dialog in IBConsole to connect to the database using a different username and password from that which was supplied in the server login, you can do so by one of the following methods:

- Select a disconnected database in the Tree pane. Choose Database| Connect As or choose Connect As in the Work pane.
- Right-click a disconnected database in the Tree pane and choose Connect As from the context menu. This displays the Database Connect dialog box:



To Connect to a Database

1. Verify that the database displayed in the Database field is correct.
2. Type the username and password for the database in the corresponding User Name and Password text fields.
3. If you want to connect as a role, specify the role in the Role text field. This is optional. 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. Once you have typed a character in the Role field, the Case

Sensitive Role Name field becomes active. Check this box if you want the server to consider case in the role name. Role names are case insensitive by default.

For more information on roles, refer to **"SQL Roles"**

4. Select the SQL Client dialect. The dialect for the database connection will default to the lower value of the client or server. For more information on SQL dialects, refer to "***"Understanding SQL Dialects"***" in the migration appendix of the InterBase *Operations Guide*.
5. Optionally, you can choose a character set to use. If you do not specify one here, the server uses the default set that was specified at creation time.
6. Click Connect.

Once you connect to a database, the database tree expands to display the database hierarchy.

Disconnecting a Database

You can disconnect a database in IBConsole by one of the following methods:

- Select a connected database in the Tree pane (you can also select any branch under the desired database hierarchy) and choose Database|Disconnect or click the Disconnect Database toolbar button
- Select a connected database in the Tree pane and double-click Disconnect in the Work pane.
- Right-click a connected database in the Tree pane and choose Disconnect from the context menu.

A confirmation dialog asks you to confirm that you wish to close the connection to the selected database. Click OK if you want to disconnect the database, otherwise click Cancel.

Un-registering a Database

Un-registering a database automatically disconnects the current database and removes it from the Tree pane.

You can un-register a disconnected database in IBConsole by one of the following methods:

- Select a database in the Tree pane (you can also select any branch under the desired database hierarchy) and choose Database|Un-register.
- Select a database in the Tree pane and double-click Un-register in the Work pane.
- Right-click a database in the Tree pane and choose Un-register from the context menu.

A confirmation dialog asks you to confirm that you wish to un-register the database. Click Yes if you want to un-register the database, otherwise click No.

Connection-specific Examples

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

- For a Windows server, the database path name must contain the appropriate drive letter designation. For example, to connect to a local database:

  ```
  D:\users\accting\fin\accred.ib
  ```

- To connect to a database on a remote server using the TCP/IP protocol:

  ```
  ntserver:D:\users\accting\fin\accred.ib
  ```

- To connect via NetBEUI (Windows server platforms only), use UNC notation:

  ```
  \\ntserver\D:\users\accting\fin\accred.ib
  ```

- For a UNIX or Linux server, you must enter the complete and absolute directory path for the database. For example:

  ```
  server:/usr/accting/fin/accred.ib
  ```

Encrypting Network Communication

Information sent to an InterBase remote client from a database server is unencrypted during the transmission process, even when the data was encrypted prior to transmission. This creates an opportunity for a security breach. To protect against this, you can use InterBase's Over-the-Wire (OTW) encryption feature to encrypt data during the transmission process.

In the XE3 release, a Strong Encryption license was incorporated into the server license that allows AES encryption which applies to OTW. This is no longer an add-on license, but is automatically part of Server/Desktop.ToGo Edition.

You can also use InterBase to encrypt a database and/or individual columns in a database table. For information on how to do so, see the ***Data Definition Guide***.

Note OTW functionality is designed to be used in conjunction with the InterBase database user name and password. It is not meant to replace it.

Requirements and Constraints

InterBase OTW encryption is provided using SSL v3 and TLS v1 security protocols. SSL uses the X.509 standard for its public key infrastructure. Both the InterBase client and server must have the X.509 files installed to use OTW encryption.

InterBase uses the following conventions on both the client and server sides:

- All the X.509 PKI (public key infrastructure) files, which include the certificate file and the CA files, **must be in the Privacy Enhanced Mail (PEM) format**.
- The parameters clientCertFile and IBSSL_SERVER_CERTFILE always refer to the PEM formatted file that contains the CA signed certificate and the private key. These files should not be distributed.
- The parameters serverPublicPath and serverPublicFile on the client, and IBSSL_SERVER_CAFILE and IBSSL_SERVER_CAPTH on the server, always refer to the public key certificate.
- InterBase XE3 supports both stronger (AES) and weak (DES) encryptions out of the box. InterBase XE and earlier supports the use of weak encryption (DES) out of the box, but to use stronger encryption (AES), you must, due to U.S. export regulations, obtain a strong encryption license from InterBase and install it on the server machine.

Note The InterBase JDBC driver now supports OTW functionality.

Setting up OTW Encryption

InterBase OTW encryption consists of two parts: one resides on the server side and the other resides on the client side and works to secure the server. The sections below explain how to set up OTW on both the server and client sides. Sample OTW configurations follow the instructions.

Note For information on specifying JDBC properties for OTW, see the **SSL File Properties** topic in the ***Developer's Guide***. Also in the ***Developer's Guide***, table 4.10 defines the new extended properties.

Before setting up OTW encryption on the server or client side, you must first obtain the necessary security certificates, provided by your Certificate Authority (CA) vendor. InterBase uses these certificates to verify identity information.

Generating Security Certificates

OTW requires the generation and use of the following certificates:

- **A public key certificate for the server.** For example, ibserverCAfile.pem
- **The server's private key and server certificate**. For example, ibserver.pem

You can use any SSL tool to generate these certificates, or contact your IT department or CA vendor. To learn how to create SSL certificates using OpenSSL, see the following website:

- http://www.openssl.org/docs/apps/openssl.html

Setting up the Client Side

The client application indicates to the InterBase client library that it needs to perform OTW encryption via the connection string. The connection string takes the OTW parameters before the database path/name, as specified in the syntax below.

Note The existing OTW properties have been changed to the new JDBC OTW properties.

Important It is strongly recommended that the old native OTW properties no longer be used. However, the new native client and server supports both the old and new names.

It is likely that this will be the last release of InterBase Client which supports the old parameters. It is important that you start using the new parameters.

Table 5.2 New Parameters

Old OTW Client Properties:	New OTW Properties Starting with InterBase XE:
OTWENABLE	ssl=true
PASSPHRASE	clientPassPhrase
PASSPHRASEFILE	clientPassPhraseFile
CERTFILE	clientCertFile
CAFILE	serverPublicFile
CAPATH	serverPublicPath

Syntax To enable OTW on the client side, use the following syntax for your database connection string:

```
  <secure server host name>[/secure server port name |
secure server port
number]?ssl=true[?serverPublicFile=complete location of the
CA file | ?serverPublicPath=name of directory containing the
CA certificates |?clientCertFile=name of the client
certificate file][?clientPassPhraseFile=pass phrase filename
|?clientPassPhrase=pass phrase]??:<database path>/<database
name>
```

The starting '?ssl=true' and the ending '??' are mandatory, and work as demarcations for the OTW encryption parameters. Table 5.3 lists the descriptions of the options used in the syntax sample.

Table 5.3 Client-side OTW Options

Options	Description
secure server host name	The hostname, Fully Qualified Domain name, or IP address of the machine that is hosting the secure InterBase server. If the machine has multiple network interface cards, define the hostname/IP of the interface that is used by the InterBase server.
secure server port name	If you have a local service name associated with the socket port of the remote server, mention that name here; this is typically defined in the /etc/services file on Unix or on the <windows system directory>/drivers/etc/services file on Windows. The client library will automatically translate this to the socket port number at connection time.
secure server port number	The socket port number where the remote server is listening for secure connection requests. You provide this OR the port name above; not both.

Table 5.3 Client-side OTW Options

Options	Description
ssl	This must be set for any of the other OTW parameters to be accepted and for the connection to be secure. **Note** This options replaces the “OTWENABLE” option. It is strongly recommended that the OTWENABLE option no longer be used.
serverPublicFile	Location of the certificate file. By default, InterBase searches for ibserverCAfile.pem in the user’s home directory. Both the serverPublicFile and serverPublicPath (defined below) must follow the PEM format as per standard SSL requirements. The client will not be expected to create this file. This file will be created by the database administrator of the server you are connecting to. If you are the person generating this file, please ensure that the public certificate file has been created binding the certificate to the server’s DNS. This DNS must match the <secure host name> used by the client. **Note** This options replaces the “CAFILE” option. It is strongly recommended that the CAFILE option no longer be used.
serverPublicPath	If you use and mention the serverPublicPath, then each file in the directory must contain only a single CA certificate and the files must be named by the subject name’s hash and extension of “.0”. There are no default for this option. It is recommended that you use the serverPublicFile as opposed to serverPublicPath; if you specify both, serverPublicFile will be used. The organization hosting the server will provide the CA file. This file is used to verify that the connection is being made to a legitimate and verified server. Care must be taken to ensure that you are referencing the CA you want to use. Please see **“About the “c_rehash” command”** for more information. **Note** This options replaces the “CAPATH” option. It is strongly recommended that the CAPATH option no longer be used.

In addition, if you need **to enable server verification of the client**, you can use the parameters described in Table 5.4. An example follows the table.

Table 5.4 Client Server Verification Parameters

Parameter name	Description
clientCertFile	Location and name of the client certification file. This certificate will be presented to the server during the SSL connection phase. The clientCertFile must be in the PEM format and must contain both the client certificate and the private key. **Note** This options replaces the “CERTFILE” option. It is strongly recommended that the CERTFILE option no longer be used.
clientPassPhraseFile	Name and location of the text file containing the client private key passphrase. You can use either the clientPassPhrase File parameter, or the clientPassPhrase parameter. **Note** This options replaces the “PASSPHRASEFILE” option. It is strongly recommended that the PASSPHRASEFILE option no longer be used.
clientPassPhrase	Specify a private key PassPhrase. You can use either the clientPassPhrase parameter or the clientPassPhraseFile parameter. **Note** This options replaces the “PASSPHRASEFILE” option. It is strongly recommended that the PASSPHRASEFILE option no longer be used.

Following is a sample of how to use these parameters in an isql command:

```
  isql> connect  'localhost/
gds_ssl?ssl=true?clientPassPhrase=clientkey?
  clientCertFile=c:\ib_builds\InterBase\secureserver\client\
client.pem?
  serverPublicFile=c:\ib_builds\InterBase\secureserver\
client\serverCAfile.pem??:c:/foo.ib'
```

About the “c_rehash” command
Use this command if you want to use the serverPublicPath parameter instead of the serverPublicFile on the client or the IBSSL_SERVER_CAPATH instead of the IBSSL_SERVER_CAFILE parameter on the server. For more information on how to set up this directory please go to the OpenSSL website and look for the c_rehash command.

.

“c_rehash” is a command provided by OpenSSL. This script automatically creates symbolic links to a directory of certificates. For example, suppose that you have a directory called some/where/certs, which contains several CA certificates, and that you want to prepare this directory for use as a serverPublicPath directory. In this case, you can use the following commands:

```
cd /some/where/certs
c_rehash .
```

Setting up the Server Side

After you have enabled the client side for OTW, you must change the configuration parameters in the SSL configuration file called “ibss_config.” This file is located in “<install_directory>\secure\server” directory. The configuration file contains information required by the server. Instructions on how to set up this file are provided below.

In addition, the InterBase server requires two DH (Diffie-Hellman) parameter files to operate. For more information about the dhparameter files, see **“Generating the dhparameter files”**.

Changing the ibss_config file
Following is sample ibss_config file:

```
  IBSSL_SERVER_HOST_NAME=localhost
  IBSSL_SERVER_PORT_NO=3065
  IBSSL_SERVER_PHASSPHRASE=serverkey
  IBSSL_SERVER_clientCertFile=<install_directory>/secure/
server/ibserver.pem
  #IBSSL_SERVER_PASSPHRASEFILE=c:/secure/pass.txt
  #example comment line
  #only needed for client verification
  #IBSSL_SERVER_VERIFY_CLIENT
  #IBSSL_SERVER_CAFILE=<install_directory>/secure/server/
root.pem
```

Table 5.5 provides a description of each parameter in the sample above.

Table 5.5 Server-side Configuration Parameters

Parameter	Description
IBSSL_SERVER_PORT_NO and IBSSL_SERVER_HOST_NAME	Port number and the hostname of the InterBase server's SSL port number and SSL machine name (can be localhost) the InterBase Server is running on. The defaults are machine name or host name and '3065.' In most cases the IBSSL_SERVER_HOST_NAME need not be set.
IBSSL_SERVER_CERTFILE	Location of the private key stored in a file.This will be used by the server for encryption. (Default location and filename: will the <install_directory>/ secure/server/ibserver.pem. The IBSSL_SERVER_CERTFILE must be in PEM format and must contain both the private key and the certificate.
IBSSL_SERVER_PASSPHRASEFILE	Location of the file containing the passphrase. This must be secure. Make sure you have the correct permissions for this file; the server only needs read access to the file during start up time. The log file will indicate via a message that the passphrase is not loaded. This means you can have the pass phrase on a removable media and once the server has started the media (and hence the passphrase) maybe safely removed.
IBSSL_SERVER_PASSPHRASE	Contains the server pass phrase to be used in conjunction with the server certificate file. Use this instead of the IBSSL_SERVER_PASSPHRASEFILE. If both are set the IBSSL_SERVER_PASSPHRASE is used instead of IBSSL_SERVER_PASSPHRASEFILE. If both are not set, InterBase assumes that the private key does not contain a pass phrase.

Table 5.5 Server-side Configuration Parameters

Parameter	Description
IBSSL_SERVER_VERIFY_CLIENT	If this parameter is set, then the server will ensure that the client has sent us a certificate. This certificate will be verified against the file specified in the IBSSL_SERVER_CAFILE (or the directory specified in the IBSSL_SERVER_CAPTH).
IBSSL_SERVER_CAFILE	Location of the file containing the CA file, which can be used to verify the client certificate.There is no default for this file. However it is recommended that you locate the file in <install_directory>/secure/server/ and call it ibrootcert.pem. The file must be in PEM format and is needed only if the IBSSL_SERVER_VERIFY_CLIENT flag is set.
IBSSL_SERVER_CAPATH	Used for the same purpose as the IBSSL_SERVER_CAFILE. However, in this case, the parameter points to a directory containing the CA certificates in PEM format.The files each contain one CA certificate, and are only needed if the IBSSL_SERVER_VERIFY_CLIENT flag is set. The files are looked up by the CA subject name hash value, which must be available. See **"About the "c_rehash" command"** for information about this command, which can be used to convert multiple PEM files into a IBSSL_SERVER_CAPATH-accessible directory.

In addition, InterBase following information is assumed about the **ibss_config** file:

- General format of the file is <parameter_name>=value.
- Lines starting with "#" are assumed to be comments.
- Lines greater than 1023 characters are truncated to 1023 characters.
- Spaces at the end of the line are considered part of the name or number, so do not put spaces at the end of a line. In case of a filename, enclose the filename in straight quotation marks to avoid problems with unseen space characters at the end of the line.

Generating the dhparameter files

As mentioned above, to use OTW, the server also requires two DH (Diffie-Hellman) parameter files. These are located at <install_directory>/secure/server and are called dh512.pem and dh1024.pem, respectively. InterBase uses the DH key exchange protocol to establish a SSL connection, be it DSA- or RSA-based. InterBase also uses ephemeral mode to ensure forward secrecy.

You are encouraged to generate your own DH parameter files, if you want these files to be unique to your installation. Otherwise, the default ones provided by InterBase will be used. In order for the InterBase server to make successful SSL connections, these files are required.

To create the dhparameter files, use the following commands:

```
openssl dhparam -check -text -5 512 -out dh512.pem
openssl dhparam -check -text -5 1024 -out dh1024.pem
```

After generating the files, copy them to the <install_directory>/secure/server directory.

Sample OTW Configurations

The following sample configurations were designed to help you effectively enable and implement OTW across your network.

Sample 1: Setting up the Client and Server Without Client Verification by the Server

This is the setup that most InterBase customers will use. In this setup, the server's identity is provided by the server's certificate and the client verifies that the server matches what the client wanted to connect to. The client also authenticates the server certificate based on a CA file located on the client.

Sample 1: Setting up the server

To set up the sample server for OTW, take the following steps:

1. Create the ibserverCAfile.pem and the ibserver.pem files.
2. Copy the ibserver.pem file to <install_directory>/secure/server/ibserver.pem.
3. Create or copy the ibss_config in the <install_directory>/secure/server/ directory from the ibss_config default file.
4. Setup and create the 2 dhparam files in the <install_directory>/secure/server directory, if you want unique ones for your location.
5. Start the server, which should be set up for receiving SSL connections on port 3065 (default).

Sample 1: Setting up the client

To set up the sample client for OTW:

1. Copy the ibserverCAfile.pem provided by the server DBA to the user's home directory.
2. Using isql, make a connection using the following as your URL. Assume your server and client are on the same machine then the hostname is "localhost".

```
isql> connect "localhost/3065?ssl=true??:c:/foo.ib";
```

You are now set up to use OTW. This example used default locations for all the certificate and CA files used. If you do not use the defaults and decide to change the location of the server files, you must change the IBSSL_SERVER_CERTFILE parameter in the ibss_config file to point to your PEM formatted Certificate (plus private key) file.

If you locate the CA file (on the client machine) in a directory other than your home directory use the following command on connect:

```
  isql> connect "localhost/
3065?ssl=true?serverPublicFile=<your CA file location and
name>??:c:/foo.ib";
```

Sample 2: Setting up the Client and Server for Verifying the Client

To setup InterBase with client side verification, you must first perform all the steps in Sample 1 for both server and client setup. For this example, we will assume that InterBase is installed in C:\InterBase.

Sample 2: Setting up the server

To set up the sample server:

1. Copy the ibrootcert.pem file to the <install_directory>/secure/server directory. This is the public key certificate used by the server to identify the client.
2. The ibss_config file must be modified to indicate to the server that client verification has been enabled, and that the public key certificate location. This is done by adding the following to the <install_directory>/secure/server/ibss_config file:

```
  IBSSL_SERVER_VERIFY_CLIENT
  IBSSL_SERVER_CAFILE=c:\InterBase\secure\server\
ibrootcert.pem
```

Sample 2: Setting up the client

To set up the sample client:

1. Copy the ibclient.pem file, which is a PEM formatted file that contains the client certificate and private key, to your HOME directory on the client. Assume that your HOME directory is C:\smistry, then the complete path for the file will be c:\smistry\ibclient.pem.
2. Specify the location of your client certificate and private key on the connection URL. For example, if you are connecting to c:/foo.ib using isql, the command would be:

```
  isql> connect "localhost/3065?ssl=true?clientCertFile=C:\
smistry\ibclient.pem??:c:/foo.ib";
```

Sample 3: Setting up a JDBC Client and InterBase Server for Verifying the Client

These instructions are only needed if you need your JDBC client connection verified by the server. Use the Sun provided keytool.

You can use the "keytool -genkey" to generate a new self signed private key and public key pair. This password is to be used when making a connection via JDBC (clientPassPhrase).

Examples

```
[C:/ib_svn_build/certificates] keytool -genkey -keystore
smclient.jks
Enter keystore password:  client
What is your first and last name?
  [Unknown]:  Shaunak Mistry
What is the name of your organizational unit?
  [Unknown]:  InterBase
What is the name of your organization?
  [Unknown]:  Embarcadero
What is the name of your City or Locality?
  [Unknown]:  Scotts Valley
What is the name of your State or Province?
  [Unknown]:  CA
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=Shaunak Mistry, OU=InterBase, O=Embarcadero, L=Scotts
Valley, ST=CA, C=US
correct?
  [no]:  yes
```

Enter key password for <mykey>

```
  RETURN if same as keystore password):
```

These commands created a new keystore called smclient.jks. It contains your private and public key and a self signed certificate.

If you follow this example then the following values need to be appended to your JDBC connection URL to make a JDBC connection using client side verification.

```
?clientPrivateFile=c:/smistry/
smclient.jks?clientPassPhrase=client
```

Next you can use the keytool -export -rfc to export you public key. This public key must be added to the server,and pointed to by the server using the IBSSL_SERVER_CAFILE option in the ibss_config file.

```
[C:/ib_svn_build/certificates] keytool -export -rfc -
keystore smclient.jks
```

Enter keystore password: client

```
-----BEGIN CERTIFICATE-----
MIIDHzCCAtwCBEpt7k4wCwYHKoZIzjgEAwUAMHUxCzAJBgNVBAYTAlVTMQsw
CQYDVQQIEwJDQTEW
MBQGA1UEBxMNU2NvdHRzIFZhbGxleTEUMBIGA1UEChMLRW1iYXJjYWRlcm8x
EjAQBgNVBAsTCUlu
dGVyQmFzZTEXMBUGA1UEAxMOU2hhdW5hayBNaXN0cnkwHhcNMDkwNzI3MTgx
MzM0WhcNMDkxMDI1
.....
utRZT+ZxBxCBgLRJFnEj6EwoFhO3zwkyjMim4TwWeotUfI0o4KOuHiuzpnWR
bqN/C/ohNWLx+2J6
ASQ7zKTxvqhRkImog9/hWuWfBpKLZl6Ae1UlZAFMO/
7PSSoDgYQAAoGAOOavhpQAOLHr/Yw59LrA
SOflcsA15BaAy1NUEl65cqb1/TO/
jWroKjlG8dv1uNdsc2kZ4ptmM0L2RjksLxcrqUBm9qjedan9
X8cjEnTeU2hOrmARoZeFhlvtw4CfiuXwnFeagF2IxrETyVLEXMV1A5ATRzrd
TqQcfnwPCua0F3Ew-----END CERTIFICATE-----
CwYHKoZIzjgEAwUAAzAAMC0CFQCJtK/
qpIw0ahuIYqYP5d1D90UbdAIUEeU4nXvZAUxZv5SPcFFP
uowm7bI=
-----END CERTIFICATE-----
```

or use the command

```
[C:/ib_svn_build/certificates] keytool -export -rfc -keystore
smclient.jks -file mycert.pem
```

Enter keystore password: client

Certificate stored in file <mycert.pem>

Now the file mycert.pem contains your public certificate. Move this to the server and make sure this is included in the file pointed to by the IBSSL_SERVER_CAFILE.

If you want to get your private key validated by a certification authority, the client need to use the "keytool -certreq" command to generate a certificate signing request for a Certificate signing authority. Once this request is validated you would add this certificate reply to your keystore via a "keytool - import" command. This is followed by a "keytool -export" command to get the certificate to authenticate your public key. This exported certificate will then be moved to the InterBase server, so the InterBase server can "trust" and verify the client private key.

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 dialog. See **“Communication Diagnostics”** for more information.

Connection Refused Errors

If the client fails to reach the server host at all, or the *gds_db* service fails to answer, you might 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 could rule out improper network configuration on the client.

Test connectivity from another client to the InterBase server host; if it succeeds, this could 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, except in some Linux distributions. 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 could be restricted and the client might 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 command-line **isql**.
2. Attempt to connect to a database without specifying a hostname: list just the path.

The Communications Diagnostic dialog also has a local database attachment test. See **"DB Connection Tab"** for details.

Note Local connection mode is not available on UNIX 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 command-line **isql** or InterBase IBConsole 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.

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 InterBase Manager on Windows. See **"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
```

This file is found in the following locations:

Windows server platforms: *C:\system32\drivers\etc\services* or *C:\Windows\system32\drivers\etc\services* on new Windows platforms
On Windows non-server platforms: *C:\windows\services*.
On UNIX: */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.

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 Windows, 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”**.

Is the database on a networked file system?

A database file must not reside on an NFS file system 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 File Systems”** for more details.

To correct this situation, move your database to a file system 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 the InterBase security database (*admin.ib* by default).

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 the InterBase security database (*admin.ib* by default) that contains users and passwords must also be writable by the **ibserver** process.

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

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

Disabling Automatic Internet Dialup

Microsoft Windows operating systems offer a networking feature that is convenient for users who use a modem to connect to the Internet: any TCP/IP request that occurs on the system activates an automatic modem dialing program. This is helpful for users who want to connect quickly as they launch a web browser or email client application.

This convenience feature is unnecessary on systems that use a client/server application to access an InterBase server on a local network. The TCP/IP service request that the client invokes triggers the Windows automatic modem dialer. This interferes with quick network connections from client to server.

This section describes several methods to suppress the automatic modem dial feature of Windows operating systems. No more than one of these methods should be necessary to accomplish the networking configuration you need.

Reorder network adapter bindings

You probably have a dialup adapter and an ethernet adapter for your local network. On Windows, you can reverse the bindings order for your two adapters to force the ethernet adapter service the TCP/IP request before the dialup adapter tries. You can do this in the Control Panel by choosing Networking|Bindings|All Adapters|Move Down.

The local ethernet adapter satisfies TCP/IP requests it can, and those requests that can't be done locally—such as Internet requests—are passed on to the next adapter in the list, the dialup adapter.

Disabling autodial in the registry

Perform the following:

1 Start the registry editor with **regedit.exe**

2. Move to the registry key HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Internet Settings: EnableAutoDial
3. Change the value from 0 to 1

Preventing RAS from dialing out for local network activity

Perform the following if you are using Windows NT RAS:

1. Start the registry editor, with **regedit.exe**
2. Move to the registry key HKEY_CURRENT_USER\Software\Microsoft\RAS Autodial\ Addresses

   A better way to view these is to type **rasautou -s** from the command prompt
3. In the sub-keys, look for the local address and name; select the key and select **Delete** from the **Edit** menu
4. Close the registry editor

You might also wish to add addresses to the disabled list:

5. Start the registry editor with **regedt32.exe**, not **regedit.exe**
6. Move to the registry key HKEY_CURRENT_USER\Software\Microsoft\RAS Autodial\Control
7. Double click Disabled Addresses and add the address on a new line; click **OK** when you are finished
8. Close the registry editor

You must reboot the machine in both of the above cases.

Other Errors

Unknown Win32 error 10061

This error is often associated with a 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.

Unable to complete network request to host

This error occurs in cases when the InterBase client cannot establish a network connection to the server host. This can occur for a variety of reasons. Below is a list of common causes:

- The BDE Administrator requires that you specify the InterBase connect string in the SERVER NAME alias property. You must use this property and must not use the PATH alias property, or else you receive the network error message.
- The InterBase client attempts to translate the server portion of your connect string to an IP address, by calling gethostbyname(). If you supplied an IP address, gethostbyname() is likely to fail to resolve it. Some modern TCP/IP

drivers— including Winsock 2 and Linux TCP/IP—do resolve strings that look like IP addresses. If you are on Windows, specify hosts by name, or else upgrade your TCP/IP driver to Winsock 2.

- The InterBase client must look up the InterBase network *service* by name. If the client doesn't find the entry for **gds_db** in the *services* file, it might fail to connect to the server, and give the network error. You can create the entry in the *services* file manually, or reinstall InterBase to perform this task.
- The server you specify must be running on the network that you use. If the hostname corresponds to a host that is inaccessible because of network interruption, or the host is not running, then the connection request fails with the network error.
- The syntax of the InterBase connect string determines the network protocol the client uses to connect to the server host (see **"Connection-specific Examples"**). Different server platforms support different subsets of network protocols. If your server does not support the protocol indicated by your connect string, the connection attempt fails with the network error. For example, the NetBEUI connection syntax (*server**C:**path**database.ib*) works only if your server is a windows 2008, or XP server. The syntax does not work if your server is running UNIX or Linux.
- A network connection request succeeds only if the InterBase server is installed and active on the server host, and that the InterBase server is licensed to receive remote connection requests. If there is no process listening for connection requests, the client's connection requests with the network error. You should check that the InterBase server is installed on the server, that it is running, and that the license includes the Server capability.

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 mis-configured, it is not always evident where the problem lies. InterBase Communication diagnostics helps to identify the source of the problem by testing each layer progressively for existing or potential network problems.

You can access the Communication Diagnostics dialog by one of the following methods:

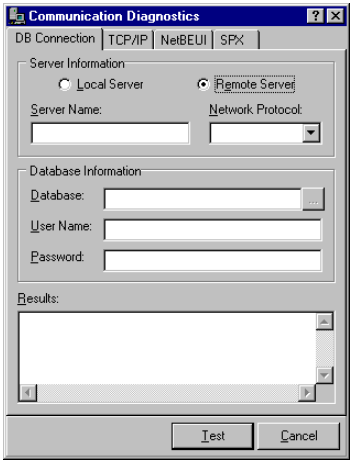
- Select a disconnected server in the Tree pane. Choose Server|Diagnose Connection**.**
- Right-click InterBase Servers or any disconnected server in the Tree pane and choose Diagnose Connection from the context menu.
- Select a disconnected server from the Tree pane and double-click Diagnose Connection in the Work pane.

There are four types of diagnostics that you can perform. The Communications Diagnostics dialog has separate tabs for each diagnostic type.

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.

Figure 5.11 Communications Dialog: DB Connection



To Run a DB Connection Test

1. Select either the Local Server option or the Remote Server option.
2. If you choose Local Server, the Server Name and Network Protocol information is not required. These text fields are disabled. You can proceed to step 5.
3. If you choose Remote Server, type the name of the server in the Server Name text field.

   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 Name text field.
4. If you choose Remote Server, select a network protocol from the drop-down list: either TCP/IP, NetBEUI, named pipe, or local. Protocols are valid only when they are supported by both the client and the server.
5. Enter the database filename, including the path where file is located, in the Database text field. If you selected the Local Server option in step 1 you can also click the browse button ... to locate the file you want. If you selected the Remote Server option, however the browse button is disabled.

6 Type the username and password for the database in the corresponding User Name and Password text fields.

7 Click Test to display the results of the connectivity test in the Results text area.

Sample output (local connection)

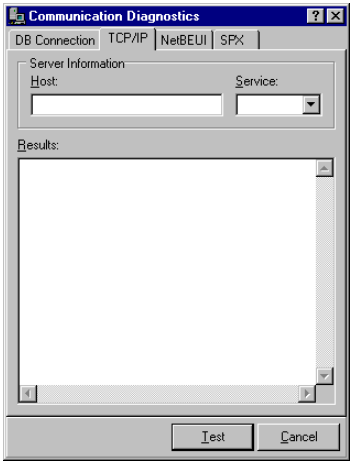
```
Attempting to attach to:
  C:\Program Files\Embarcadero\InterBase\examples\Database\
employee.ib
  Attaching ...Passed!
  Detaching ...Passed!

InterBase Communication Test Passed!
```

TCP/IP Tab

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

Figure 5.12 Communications dialog: TCP/IP



To run a winsock TCP/IP connectivity test

1 Enter either a network host name or IP address in the Host text field.

2 Select a service name or number from the drop-down Service list. Possible service selections are: 21, Ping, 3050, ftp, **gds_db**.

Select Ping from the Service drop-down list to display a summary of round-trip times and packet loss statistics.

3 Click Test to display the results of the connectivity test in the Results text area.

Sample results (ftp)

```
Initialized Winsock.

Attempting connection to DBSERVE.
```

```
Socket for connection obtained.

Found service 'FTP' at port '21'.
Connection established to host 'DBSERVE' on port 21.

TCP/IP Communication Test Passed!
```

Sample results (ping)

```
Pinging DBSERVE [200.34.4.5] with 32 bytes of data.

Reply from 200.34.4.5: bytes=32 time=1ms TTL=128
Reply from 200.34.4.5: bytes=32 time=1ms TTL=128
Reply from 200.34.4.5: bytes=32 time=1ms TTL=128
Reply from 200.34.4.5: bytes=32 time=0ms TTL=128

Ping statistics for 200.34.4.5:
    Packets: Send = 4, Received = 4, Lost = 0 (0%),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

Table 5.6 Using Communication Diagnostics to Diagnose Connection Problems

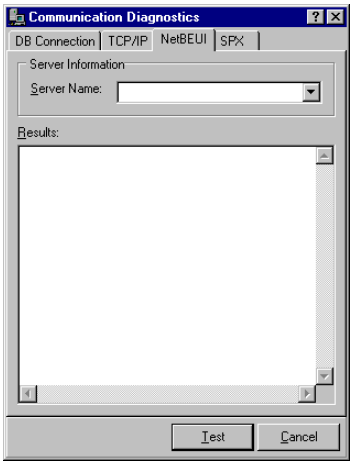
If the error message is	Then check
Failed to find named port	Your *services* file to be sure there is an entry for *gds_db* in the form: **gds_db 3050/tcp**
Failed to connect to host	• Hostname, port 3050 • The InterBase Server to make sure it is installed properly, is running, and is configured for 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

NetBEUI Tab

NetBEUI is supported on all Windows clients, but only Windows server platforms support NetBEUI as a server.

Use this property sheet to test NetBEUI connectivity between the client and the server.

Figure 5.13 Communications Dialog: NetBEUI



To run a NetBEUI connectivity test

1. Select a Windows server on which InterBase has been installed from the Server Name drop-down list. If the desired server does not exist in this list, you can type the server name in the edit portion of the drop-down list.
2. Click Test to display the results of the connectivity test in the Results text area.

Sample output (NetBEUI connection)

```
Attempting to attach to DBSERVE using
the following named pipe:
  \\dbserve\pipe\interbas\server\qds.db.

NetBEUI Communication Test Passed!
```

The connection may fail if a Microsoft Windows network is not the default network for the client. You should also be logged into the Windows network with a valid user name and password.

Chapter

6

Database User Management

InterBase provides several methods for configuring and enforcing security by controlling how a database is accessed and used. Server security enables you to:

- Add a user to the security database
- Delete a user from the security database
- Modify user information in the security database
- Display a list of users in the security database
- Enable embedded user authentication
- Create database alias
- Delete a database alias
- Display a list of all database alias

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 security database for each server host. This database, *admin.ib* by default, contains a record for each legitimate user who has permission to connect to databases and InterBase services on that host. Each record includes the user login name and the associated encrypted password. The entries in this security database apply to all databases on that server host.

The username is significant to 31 bytes and is not case sensitive. When a stronger password protection is implemented, the password is now significant to 32 bytes instead of 8 and is case sensitive.

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

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 account that can bypass normal SQL security and perform tasks such as database backups and 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, delete, or modify user configurations. SYSDBA can use either gsec or IBConsole to authorize a new user by assigning a username and password in the security database.

Important We *strongly* recommend you change the password for SYSDBA as soon as possible after installing InterBase. If you do not alter the SYSDBA password, unauthorized users have easy access and none of your databases are secure.

Other Users

The SYSDBA account can create other users on a per-server basis. Use gsec or IBConsole to create, modify, or remove users from the InterBase 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 person 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 could 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 could use the same InterBase user account.

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 4-17.

Notes

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

The InterBase Security Database

The InterBase server stores the names and passwords of its authorized users in a special security database that resides in the InterBase home directory. By default, it is named *admin.ib*.

Note InterBase XE implements stronger password protection on InterBase databases. See: **see "Implementing Stronger Password Protection," in this chapter.**

You can use another name for the security database if you wish. If you change this name, you must add an entry to the *ibconfig* file, setting ADMIN_DB to the new name.

```
ADMIN_DB newname.ib
```

Note In older versions of InterBase, the security database was named *isc4.gdb*. Because files with a *gdb* extension automatically get backed up whenever they are touched in some versions of Windows XP, using this extension degrades database performance. Therefore, InterBase recommends using a different extension for database names.

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

Table 6.1 Format of the InterBase Security Database

Column	Required?	Description
User name	Yes	The name that the user supplies when logging in; maximum length is 31 bytes
Password	Yes	The user's password • Case sensitive • Only the first eight bytes are significant • Maximum length: 32 bytes.
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)

Implementing Stronger Password Protection

Stronger password protection on InterBase databases can be implemented with InterBase XE. This additional functionality supports a longer effective password length, resulting in stronger password protection.

Requirements/Constraints

- This design supports server-wide user authentication as manifested by the USERS table of the security database, configured with the IBCONFIG.ADMIN_IB property parameter, which defaults to the file ***admin.ib***.
- The design also supports EUA (Embedded User Authentication) databases. As with the non-EUA databases, it also has to be explicitly enabled by the owner/administrator. Please note that the USERS table in admin.ib has RDB$USERS as the counterpart in EUA databases; so the earlier references have to be compatible with EUA database references.
- A user account in the USERS table can only accommodate a single password hash value. This restriction means that once the user account password is changed to use SHA-1, the user has to use the new IB client to log into the new IB server.
- A plaintext password length of 32 bytes is supported in this release, up from 8 bytes in earlier versions of InterBase.

- An updated version of IBConsole is present in the kit. This version does not show the “Default” buttons in the database/server login screens.
- A batch script (*changepassword.bat*) is now provided in the <interbase>/bin directory to update the SYSDBA account password *post-install*.

Getting Started

The DES-CRYPT password algorithm has been replaced with a modern cryptographic hash function that is more widely accepted by organizations in private industry and government. The design uses SHA-1, which generates a fixed length 160-bit hash.

1. Before starting, it is strongly recommended that you backup your *old* admin.ib from the current installation before installing the new InterBase. This allows you to restore it, if needed.
2. After new IB has been installed on the server, run the following against ***admin.ib***:

```
isql admin.ib -user SYSDBA -pass xxxxxxx

sql> ALTER DATABASE SET PASSWORD DIGEST 'SHA-1';

sql> CREATE DOMAIN PASSWORD_DIGEST AS CHAR(16) CHARACTER
SET ASCII;

sql> ALTER TABLE USERS ADD PASSWORD_DIGEST
PASSWORD_DIGEST;

sql> UPDATE USERS SET PASSWORD_DIGEST = 'DES-CRYPT';

sql> COMMIT;
```

Note The ALTER DATABASE command can only be run by the database owner or SYSDBA. This command modifies RDB$DATABASE.RDB$PASSWORD_DIGEST to the string value "SHA-1". This means that all new password hash generation for new or existing user accounts in the USERS table will use the SHA-1 hash function.

The password hash function can be reset to DES-CRYPT using the same DDL:

```
ALTER DATABASE SET PASSWORD DIGEST 'DES-CRYPT';
```

The admin database is now prepped so that new user accounts or modifying the password of existing accounts will generate SHA-1 password hashes against plaintext passwords up to an untruncated length of 32 significant bytes.

GSEC [add | modify], IBConsole, and the IB Services API support the SHA-1 password hash algorithm. Any of these tools can be used to maintain the passwords of server-wide user accounts. If an existing user account has had its password changed then that user must log in to the server using the new IB client library.

Important There will be backward compatibility problems if the converted ***admin.ib*** database is backed up and restored by an older IB engine after the password hashes have been converted to SHA-1. Older IB engines will not understand the different password hashes and will cause unrecoverable login errors.

Enabling Embedded User Authentication

Embedded user authentication (EUA) stores database user name and password information directly in the database. When user authentication is embedded in a database, database metadata IP is better protected from outside inspection. EUA also makes transportable databases more secure.

Only the database owner is allowed to administer embedded user authentication. A regular user may alter the password for their own user account.

Having a SYSDBA user account under embedded user authentication is optional. If there is a SYSDBA account, it has most of the same privileges for the database in which it is embedded that any admin.ib would have. The sole exception is that the SYSDBA cannot maintain admin control for EUA if it has been implemented by another user.

Important EUA must be enabled to use InterBase's encryption feature, which facilitates the encryption of database pages and columns. Access to encrypted databases and columns can be given to specified users when EUA has been enabled. For more information about the InterBase encryption feature, see the ***Data Definition Guide***.

Check if EUA is Active with isc_database Info API

If isc_databaseinfo() is invoked with info item isc_info_db_eua_active it returns:

- 1 if EUA is active for the database

- 0 if EUA is not active

Only the owner or SYSDBA can query for this information, once connected to the database. For all other users, the info request is ignored.

Enabling EUA Using iSQL

You can enable EUA using the following commands:

- When creating a new database, use:

  CREATE DATABASE <database name> [WITH ADMIN OPTION]

  The admin clause automatically inserts name and password information for the user creating the database into the RDB$USERSsystem table.

- When altering an existing database, use:

  ALTER DATABASE <database name> [ADD ADMIN OPTION]

Alternatively, the **gsec** command-line utility has a new option, -user_database [database_name], which allows that tool to maintain user accounts for embedded user authentication enabled databases.

To disable EAU, use the following syntax:

```
ALTER DATABASE <database> [DROP ADMIN OPTION]
```

Once EUA is disabled, access to the database will be authenticated via the server's centralized user authentication database ADMIN.IB.

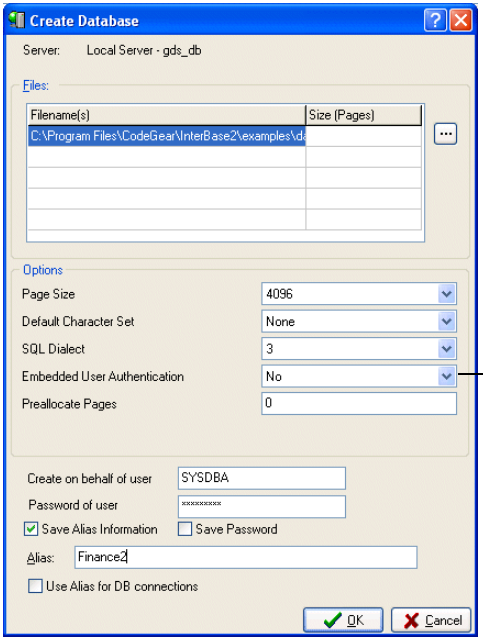
Enabling EUA Using IBConsole

You can enable EUA using the IBConsole when you use the IBConsole interface to create a new database.

To enable EUA from IB Console:

1. Right-click on Databases and choose Create Database from the context menu.
2. On Create Database, shown in Figure 6.1, in the **Embedded User Authentication** field, change the default, No, to **Yes**.

Figure 6.1 Create Database Dialog



3. Change the other settings as needed, and choose **OK** to create the database. EUA is now enabled.

Adding and Modifying Users in a EUA-enabled Database

To add users to a EUA-enabled database, use the isc_spb_user_dbname service parameter block (SPB) with the isc_action_svc_add_user service action. The service actions allowed are isc_action_svc_xxx_user, where you replace xxx with add/modify/delete/display for each respective action.

The following code sample illustrates how to use this SPB to add a user to EUA-enabled database:

```
#ifdef EUA_DATABASE
*thd++ = isc_spb_user_dbname;
ADD_SPB_LENGTH (thd, strlen(target_db));
for (x = target_db; *x;)
  *thd++ = *x++;
#endif
```

For more information about using this and other service parameter blocks and service actions, see the InterBase ***API Guide***.

System Table Security

InterBase stores the database metadata in its system tables. These tables have an intricate set of dependencies between them, and writing to one without sufficient knowledge can corrupt the database. For this reason, the system tables have the following *default* security access applied to them:

- By default, PUBLIC users have only SELECT privileges on the system tables.
- The database owner, the SYSDBA user, and the operating system administrator (root on UNIX and Administrator on Windows server platforms) have full access to the system tables, including write permission. These users can, if desired, assign write privileges to individual users or to PUBLIC, using the GRANT statement.

Older Databases

InterBase applies this default security (no write access for PUBLIC) to older databases whenever possible:

- The gbak backup/restore utility applies the default security to any database when it is restored to ODS 10.1 (InterBase 6.5) or later.
- When an InterBase server that is version 6.5 or later attaches an older database, it applies the default privileges to that database if they are not already present, even if the database is ODS 10.0 or earlier.

Scripts for Changing Database Security

Three SQL scripts are included in *<ib_install>/examples/security* directory: *readmeta.sql*, *writemeta.sql* and *blindmeta.sql.* These scripts can be run against databases with isql to make wholesale changes to a database's system tables access privileges, except or rdb$users for security purposes.

- *readmeta.sql* applies the default PUBLIC access privileges: PUBLIC can only select from the system tables, but the database owner, system administrator, and SYSDBA user have full access. This script can be used to return a database that has customized system table privileges back to this default.
- *writemeta.sql* grants all system table privileges to PUBLIC. This is the behavior that existed in InterBase 6.0 and earlier.
- *blindmeta.sql* revokes all system table privileges from PUBLIC. This prevents any PUBLIC user from querying the system tables, including InterBase and third-party utilities run by PUBLIC users. For example, gstat, gbak, QLI and IBConsole would not be able to query system metadata. This script allows developers to protect their intellectual property by hiding the database design of tables, stored procedures and triggers from the general public and competitors. Blind access makes it difficult, if not impossible, for a general user to generate ad hoc queries against a database.

  A database with blind access does not prevent any user from using InterBase Data Definition Language (DDL) to define new database objects. It just prevents a user from querying or writing to the system tables directly.

  isc_blob_lookup_desc() and isc_array_lookup_bounds() Two client-side APIs, *isc_blob_lookup_desc*() and *isc_array_lookup_bounds*(), cannot execute without SELECT metadata privileges, because the APIs directly query some InterBase system tables. Thus databases that have had *blindmeta.sql* run against them are not able to execute these APIs for any users except the owner, the system administrator, and SYSDBA.

  Older InterBase clients InterBase 6.0 and previous InterBase kits cannot access a database on behalf of a user if that user has no privileges to the system tables. Thus an InterBase developer who runs *blindmeta.sql* on an InterBase database cannot ship that database to customers with InterBase 6.0 or older runtime kits and expect those users to be able to access the database. The developer would have to run *readmeta.sql* against a copy of the database and ship that to customers who have older InterBase runtimes.

Migration Issues

The InterBase engine automatically installs the default (SELECT-only) SQL privileges for PUBLIC on the system tables when attaching ODS 10.0 or older databases. Thus if all users must be able to write to database metadata, *writemeta.sql*will have to be run against each database to restore that behavior.

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 PUBLIC apply to all users.

For full description of syntax of SQL privileges, see entries for GRANT and ROLE in the *Language Reference* and ***Data Definition Guide***.

Groups of Users

InterBase 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.

SQL Roles

InterBase 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.

1. Declare the role with CREATE ROLE.

```
CREATE ROLE sales;
```

2. Assign privileges on specific tables and columns to the role using the GRANT statement.

```
GRANT UPDATE ON table1 TO sales;
```

3. Grant the role to users, again with the GRANT statement.

```
GRANT sales TO user1, user2, user3;
```

4. Finally, to acquire the privileges assigned to a role, users must specify the role when connecting to a database.

```
CONNECT 'foo.ib' USER 'user1' PASSWORD 'peanuts' ROLE
sales;
```

User1 now has update privileges on TABLE1 for the duration of the connection.

A user can belong to only one role per connection to the database and cannot change role while connected. To change role, the user 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 IBConsole 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 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* Guide.**

Note Applications using BDE version 5.02 or later, including Delphi, JBuilder, and C++Builder, have a property by which they can specify a role name. Also, the ODBC driver that currently ships with InterBase also recognizes 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 the InterBase security database. 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 a SQL standard feature.

Other Security Measures

InterBase provides some restrictions on the use of InterBase tools in order to increase security. In addition, there are things that you can do to protect your databases from security breaches. This section describes these options.

Restriction on Using InterBase Tools

As a security measure, InterBase requires that only the owner of a database or SYSDBA can execute gbak, gstat, and gfix.

- Only the database owner or SYSDBA can use gbak to back up a database. Anyone can restore a database, because there is no concept of an InterBase user for a backup file. However, only the owner or SYSDBA can restore a database over an existing database. For security purposes, make sure that your backup files are stored in a secure location. This prevents unauthorized persons from restoring databases and gaining access to them.
- On UNIX platforms, there is a further constraint on gstat: to run gstat, you must have system-level read access to the database file. To access the database with gstat, you must either be logged into the account running the InterBase server (“InterBase” or “root”) or someone must change the permissions on the database file to include read permission for your Group.

Protecting your Databases

You can take several steps to increase the security of your databases and other files on your system:

- UNIX and Linux systems: Before starting the InterBase server, log in as user “InterBase” (or “interbas”, if user names longer than eight characters are not allowed), rather than “root” (only these users can start the server). This restricts the ability of other users to accidentally or intentionally access or overwrite sensitive files such as the password file. Start the InterBase server while you are logged on as user “InterBase”.
- Windows server platforms: When the InterBase server is run as a service, you can protect a database against unauthorized access from outside InterBase (such as by a **copy** command), by making the database files readable only by the system account, under which services run. However, if you make the database readable only by the system account, remote access to the database must be by TCP/IP, not by NetBEUI.
- Because anyone can restore a backed up database, it is wise to keep your backup files in a directory with restricted access. On UNIX, only the backup file itself, not the directory in which it resides, needs to have permissions restricted to prevent reading by unauthorized persons.

  For example, if all of the following are true:

  - the backup file has permission 600 (rw-------) or 640 (rw-r-----)
  - only trusted persons belong to the groups
  - the directory has permission rwxr-xr-x

  then persons other than the responsible owner and group can see that the backup file is there, but they cannot get at it. If the user or backup script issues the command umask 077 (or 027, as appropriate) before running **gbak**, unauthorized persons will not be able to access the backup file, no matter what the permissions on the directory. (Of course the directory should not be writable by “other”; that would permit other persons to delete the backup file.)

User Administration with IBConsole

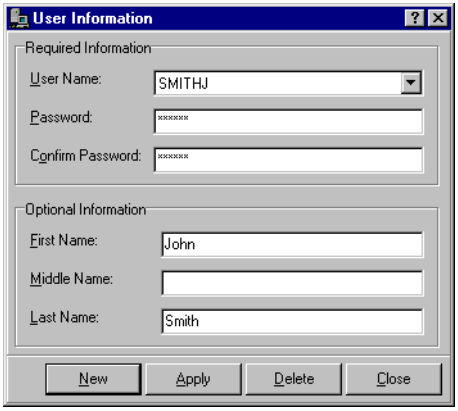
User administration is accomplished through the User Information dialog where you are able to add, modify, view and delete users. User administration can only be performed after logging in to the server.

Displaying the User Information Dialog

You can use any of the following methods to access the User Information dialog:

- Select a logged in server or any branch under the server hierarchy from the list of registered servers in the Tree pane; choose Server|User Security.
- Select a logged in server from the list of registered servers in the Tree pane. Double-click User Security in the Work pane or right-click the selected server and choose User Security from the context menu.
- Select Users under the desired server in the Tree pane to display a list of valid users in the Work pane. Double-click a user name to display the User Information dialog.

Figure 6.2 User iInformation Dialog



Adding a User

Use the User Information dialog to add new users. To access this dialog follow one of the methods described in **“Displaying the User Information Dialog”**.

To add a new user

1 Display the User Information dialog in one of the following ways:

- Select a logged in server or any branch under the server hierarchy from the list of registered servers in the Tree pane; choose Server|User Security.
- Select a logged in server from the list of registered servers in the Tree pane. Double-click User Security in the Work pane or right-click the selected server and choose User Security from the context menu.
- Select Users under the desired server in the Tree pane to display a list of valid users in the Work pane. Double-click a user name to display the User Information dialog.

2 Click New. The New and Delete buttons are disabled and the Close button changes to a Cancel button.

3 Type the new username in the User Name text field.

4 Type the user's password in both the Password and the Confirm Password text fields.

5 Add any desired optional information in the corresponding text fields. Each of the optional text fields can be up to 32 bytes.

6 Click Apply to add the new user to the security database or click Cancel to abandon your changes.

Note Usernames can be up to 31 bytes long and are *not* case sensitive. Passwords *are* case-sensitive and only the first eight characters are significant. InterBase does not allow you to create usernames or passwords containing spaces.

Modifying User Configurations

Use the User Information dialog to modify user configurations. To access this dialog follow one of the methods described in **"Displaying the User Information Dialog"**.

To modify a user's details

1 Display the User Information dialog in one of the following two ways:

- Select a logged in server or any branch under the server hierarchy from the list of registered servers in the Tree pane; choose Server|User Security to display the User Information dialog.
- Select a logged in server from the list of registered servers in the Tree pane. Double-click User Security in the Work pane or right-click the selected server and choose User Security from the context menu.
- Select Users under the desired server in the Tree pane to display a list of valid users in the Work pane. Double-click a user name to display the User Information dialog.

2 From the User Name drop-down list, select the user whose configuration you wish to modify. The user's details display. You can also type the first letter of the desired username in the User Name drop-down list to quickly scroll to usernames beginning with that letter. By repeatedly typing that same letter, you can scroll through all usernames that begin with that letter.

3 Change any of the text fields except the User Name. If you change the password, you must enter the same password in the Password text field and the Confirm Password text field.

4 Click the Apply button to save your changes.

You cannot 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

Use the User Information dialog to removed users from the security database. To access this dialog follow one of the methods described in **"Displaying the User Information Dialog"**.

1. Display the User Information dialog in one of the following two ways:
   - Select a logged in server or any branch under the server hierarchy from the list of registered servers in the Tree pane; choose Server|User Security.
   - Select a logged in server from the list of registered servers in the Tree pane. Double-click User Security in the Work pane or right-click the selected server and choose User Security from the context menu.
2. Select the user you wish to delete from the User Name drop-down list. You can also type the first letter of the desired username in the User Name drop-down list to quickly scroll to usernames beginning with that letter. By repeatedly typing that same letter, you can scroll through all usernames that begin with that letter.
3. Click Delete. A confirmation dialog inquires, "Do you wish to delete user *username*?" If you choose OK, the user is removed and is no longer authorized to access databases on the current server.

Important Although it is possible for the SYSDBA to delete the SYSDBA user, it is strongly not recommended because it will no longer be possible to add new users or modify existing user configurations If you do delete the SYSDBA user, you must reinstall InterBase to restore the InterBase security database (*admin.ib* by default).

User Administration With the InterBase API

The InterBase API includes three functions that permit authors of InterBase applications to add, delete, and modify users programmatically using three API functions: *isc_add_user*( ), *isc_delete_user*( ), and *isc_modifiy_user*( ). These functions are deprecated in InterBase 6 and later, however, because they are replaced by functions in the InterBase Services API.

The InterBase Services API provides a much broader and more robust set of tools for programmatically managing users in the security database. See "Working with Services" in the ***API Guide*** for details and examples of using the Services API functions.

For programmers using Delphi and C++ Builder, the IBX components include components for managing users. For more information on using the IBX components, refer to the ***Developer's Guide***.

Using gsec to Manage Security

The InterBase command-line security utility is gsec. This utility is used in conjunction with the InterBase security database (*admin.ib* by default) to specify user names and passwords for an InterBase server. This tool duplicates the functionality of Server|User Security in IBConsole for Windows.

The security database 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 the security database. If a matching row is found, the connection succeeds.

Important Only the SYSDBA 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.

To use gsec interactively, type gsec at the command prompt. The 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 InterBase security database. For example:

```
gsec -database jupiter:/usr/InterBase/admin.ib
```

Running gsec with Embedded Database User Authentication

You can gsec to database which enabled embedded user authentication. Use the -user_database option with embedded user authentication database specification to connect to a database which enabled embedded user authentication.

For example:

```
gsec -user_database jupiter:/usr/InterBase/employee.ib
```

Using gsec Commands

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

Table 6.2 Summary of gsec Commands

Command	Description
di[splay]	Displays all rows of the InterBase security database (*admin.ib* by default)
di[splay] *name*	Displays information only for user *name*
a[dd] *name* -pw password [*option argument*] [*option argument ...*]	Adds user *name* to the security database with password *string*. Each option and corresponding argument specifies other data associated with the user, as shown in Table 6.3, “Options for gsec”
mo[dify] *name* [*options*]	Like add, except that *name* already exists in the security database
de[lete] *name*	Deletes user *name* from the security database
alias_add *path name*	Adds a database alias. The path is the location of the database, and *name* is the name given for the alias
alias_del *name*	Deletes database alias *name* from the security database
alias_dis	Displays all database alias
alias_dis *name*	Displays information only for alias *name*
h[elp] or ?	Displays gsec commands and syntax
q[uit]	Quits the interactive session

Displaying the Security Database

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

```
GSEC> display
user nameuidgidfull name
------------------------------------------
JOHN  123 345John Doe
JANE  124 345Jane Doe
RICH  125 345Richard Roe
```

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 password [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 the InterBase security database (*admin.ib* by default).

Table 6.3 Options for gsec

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
-user_database ***string***	Name of user database
-database ***string***	Name of remote security database

Note The -pa switch specifies the root or the SYSDBA account password; -pw specifies the password for the 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 the contents of the security database:

```
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 the security database, then enter the new user name and re-enter the other information.

Deleting Entries from the Security Database

To delete a user’s entry from the security database, 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 a Windows Command Prompt

To use gsec from the Windows 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 the InterBase security database, enter:

```
C:> gsec -display
```

Using gsec to Manage Database Alias

Database Alias eliminates the need of knowing the exact location of the database file by the client application as long as the client application refers to the database by its alias.

Adding database alias to the security database

To add database alias to the security database, use the alias_add command:

```
alias_add alias name
```

and alias_dbpath *path name*

where path is the location of the database.

For example, to add the database alias "emp" with the path "C:\Embarcadero\InterBase\examples\database\employee.ib", enter:

```
GSEC> alias_add emp -alias_dbpath "C:\Embarcadero\InterBase\
examples\database\employee.ib"
```

Note Quotes are necessary for paths that contain spaces.

Use alias_dis to verify the entry:

```
GSEC> alias_dis emp C:\Embarcadero\InterBase\examples\
database\employee.ib
```

Deleting database alias from the security database

To delete a database alias from the security database, use the alias_del command:

```
alias_del name
```

For example, to delete the database alias “emp”, enter:

```
GSEC> alias_del emp
```

gsec Error Messages

Table 6.4 gsec Security 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.

Table 6.4 gsec Security Error Messages

Error Message	Causes and Suggested Actions to Take
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.
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 InterBase security database does not exist or cannot be located by the operating system.

Chapter

7

Database Configuration and Maintenance

This chapter describes configuration and maintenance issues for individual databases, including the following topics:

- Database Files
- On-disk Structure (ODS)
  - To upgrade the ODS of an older database, you must back it up using the backup utility for the version of the existing database and then restore it using the current version of InterBase.
- Creating Databases
- Backup File Properties
- Shadowing
- Setting Database Properties
- Sweep Interval and Automated Housekeeping
- Configuring the Database Cache
- Forced Writes vs. Buffered Writes
- Validation and Repair
- Shutting Down and Restarting Databases
- Limbo Transactions
- gfix Command-line Tool

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.

You can 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, and so on.

Database File Size

InterBase database file size is the product of the number of database pages times the page size. The minimum page size is 1 KB, the default page size is 4KB, and the maximum page size is 16KB. Each page can store records only from a single table. You set the database page size when you create a database by using the PAGE SIZE clause of the CREATE DATABASE statement, or its equivalent in IBConsole. You can change the page size when you restore a database using **gbak** or IBConsole.

InterBase supports 64-bit file IO, so the size of a database file is effectively limited only by the operating system.

Note Using **gbak** is the *only* way to reduce the size of the primary database file. When you restore a database, you can specify multiple files without reference to the original file sizes.

Dynamic File Sizing

InterBase dynamically expands the last file in a database as needed. This applies to single-file databases as well as to the last file of multifile databases. Specifying a LENGTH for the last or only file in a database has no effect.

Database File Preallocations

The InterBase SQL statement CREATE DATABASE includes a preallocation clause to specify extra database space for the new database. The space is actually allocated when the user detaches from the connection that was established by the CREATE DATABASE statement. The database preallocation feature supports secondary database files in that the preallocation will be spread across all secondary files in accordance with their file size specifications.

To specify preallocation, use the following syntax:

Example

```
... [[NO] PREALLOCATE [=]  int  [PAGES]]
```

By default, creating a database does not preallocate additional database pages, so it is as if NO PREALLOCATE had been specified. IB provides this syntax so that a DDL script can explicitly state and document that preallocation has not been specified. Database preallocation is always specified in units of database pages to be consistent with other related features (i.e., length of secondary database files or shadow sets).

Important If a preallocation exceeds available disk space, the IB thread making the write request when the device fills will timeout after 1 minute of waiting for the I/O to complete. It makes 4 additional I/O attempts, waiting 1 minute each time, to complete the write (results written to the InterBase log). If space is not freed to allow the preallocation operation to continue, the space requested will not be allocated.

isql -extract PREALLOCATE

The CREATE DATABASE command now includes the isql -extract PREALLOCATE clause to the formatted CREATE DATABASE statement if there is a non-zero preallocation value for the database. The isql *extract* operation can be invoked with the -a|-x options.

GSTAT

GSTAT displays the database preallocation information, which is stored on the database header page. Following is a sample from a GSTAT -H command:

Example

```
variable header data:
    Preallocate pages: 5000
    Sweep interval: 25000
    *END*
```

API DPB Parameter

At the InterBase API-level, there is a DPB parameter, isc_dpb_preallocate, that takes a 4-byte integer to specify database preallocation. It is only recognized and processed by isc_create_database(). isc_attach_database() silently ignores isc_dpb_preallocate. You can use the isc_info_db_preallocate database info parameter to request database preallocate information stored on the database header page.

With the InterBase service API, actions isc_action_svc_backup (isc_action_svc_restore) take new parameters, isc_spb_bkp_preallocate (isc_spb_res_preallocate), respectively. Both parameters take a 4-byte argument to specify the database preallocation in units of database pages. The service parameters have the same numeric value but two symbolic constants are provided for source code clarity to show the proper intent.

Note See “Working with Databases” in the ***API Guide*** for more information about DPB parameters.

External Files

InterBase permits external files to be used as *external tables*. These tables are limited in their functionality:

- From a database that is in read-write mode, you can execute only SELECT and INSERT statements on external tables. From a read-only database, you can execute only SELECT statement on external tables.
- You cannot define indexes on external tables; they are outside of the control of the multigenerational architecture.
- The 2GB external file size limit has been removed from InterBase XE onward.

The default location for external files is <*InterBase_home*>/*ext*. InterBase can always find external files that you place here. If you want to place them elsewhere, you must specify the location in the *ibconfig* configuration file using the EXTERNAL_FILE_DIRECTORY entry.

Important For security reasons, it is extremely important that you not place files with sensitive content in the same directory with external tables.

Migration note: If you are migrating from InterBase 6.x or older to InterBase 7.x or newer, and your database includes external table files, you must either move these files to <*InterBase_home*>/*ext* or note their locations in *ibconfig* using the EXTERNAL_FILE_DIRECTORY entry

Temporary Files

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

File Naming Conventions

In earlier versions, InterBase database files were given a file extension of *gdb* by convention. InterBase no longer recommends using *gdb* as the extension for database files, since on some versions of Windows ME and Windows XP, any file that has this extension is automatically backed up by the System Restore facility whenever it is touched. On those two platforms, using the *gdb* extension for database names can result in a significant detriment to performance. Linux and Solaris are not affected. InterBase now recommends using *ib* as the extension for database names.

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.

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 statements, file specification is denoted as '*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 cannot include a node name, but can specify a full path name.

Multifile Databases

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

The Database Restore task in IBConsole and in the **gbak** command-line utility permit you to create a multifile database. The only way to alter the file size allocation of an existing database is to back up and restore the database file.

Adding Database Files

You have the option of specifying the size of secondary files in either of two ways: specify the page on which each secondary file starts, or specify the length in database pages of each file. When you specify the size using the LENGTH keyword, do not specify the length of the final file. InterBase sizes the final file dynamically, as needed.

The following **isql** example adds files using STARTING AT syntax:

```
CONNECT 'first.ib';

ALTER DATABASE
  ADD FILE 'second.ib' STARTING AT 50000;
```

Altering Database File Sizes

You cannot use ALTER DATABASE to split an existing database file. For example, if your existing database is 80,000 pages long and you issue the command above, InterBase starts the new database file at page 80,001. The only way to split an existing database file into smaller files is to back it up and restore it. When you restore a database, you are free to specify secondary file sizes at will, without reference to the original file sizes.

The following **isql** example adds a file using LENGTH syntax. *second.ib* will begin on the page following the final page of *first.ib* and will grow to 50,000 database pages. Then InterBase begins writing to *third.ib* and dynamically increases the size as necessary.

```
CONNECT 'first.ib';
ALTER DATABASE ADD FILE 'second.ib' LENGTH 50000
  ADD FILE 'third.ib';
```

InterBase starts writing data to *third.ib* only after *second.ib* file fills up. In the example above, *second.ib* is 50,000 pages long, and begins following the original file. InterBase will begin filling the *third.ib* file after *second.ib* reaches 50,000 pages. Database pages are 4KB each by default and have a maximum size of 8KB.

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 131,000 database files, including shadow files. Note that your operating system might have a lower limit on the number of simultaneous open files than 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 can 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 change the sizes of the files of a multifile database when using **gbak** to restore a database. If you need to move a multi-file database to a different disk or directory, use **gbak** to back up the database, then specify the new locations of all secondary files as you restore the database. See **"Performing backups and restores using the gbak command"** .

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 tends to accumulate without bounds, and some day in the future your database might exceed your file system 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.

Networked File Systems

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) cannot reside on networked or remote file systems (called *mapped drives* on Windows and *NFS file systems* on UNIX). External tables and UDF libraries can reside on networked file systems, but this practice is not recommended because networked file systems can suffer from intermittent availability.

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

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 database file. In general, major ODS versions (those incrementing the number to the left of the decimal point) introduce features that are not backward compatible with earlier ODS versions.

The InterBase XE format is ODS 15, but still supports existing databases with ODS 13. InterBase XE may work with ODS 12, 11, and 10 but is not supported. It is strongly recommended to backup and restore your database so it can be upgraded to ODS 15 and benefit from the newer features.

When you create a new database or restore a backup file in the current version of InterBase, the resulting database file has the current ODS version.

Important To upgrade the ODS of an older database, you must back it up using the backup utility for the version of the existing database and then restore it using the current version of InterBase.

Read-write and Read-only Databases

InterBase databases have two modes: read-only and read-write. At creation, all databases are both readable and writable: they are in *read-write mode*.

Read-write Databases

To function in read-write mode, databases must exist on writable media and the **ibserver** process must have write access to the database file. For databases that are in read-write mode, this is true even when they are used only for reading because the transaction states are kept in an internal inventory data structure within the database file. Therefore any transaction against the database requires the ability to write to the transaction inventory.

Under both Windows and UNIX, read-write database files must be writable by the user ID for the **ibserver** process. However, the operating environment or file system can be configured to create files that have limited file privileges by default. If you attempt to attach to a database and get an error of “unavailable database,” first check to see if the database file’s permissions are such that the user ID of the **ibserver** process does not have write privilege on the database file.

Read-only Databases

You can change InterBase databases to *read-only mode*. This provides enhanced security for databases by protecting them from accidental or malicious updates and enables distribution on read-only media such as CDROMs. Databases are always in read-write mode at creation time. This feature is independent of dialect. Any ODS 10 or higher database can be set to read-only mode.

You can use **gbak**, **gfix**, or IBConsole to change a database to read-only mode. (See **“Making a Database Read-only”** below.)

Properties of Read-only Databases

- In read-only mode, databases can be placed on CD-ROMs or in read-only file systems as well as on read-write file systems.
- Attempted INSERT, UPDATE, and DELETE operations on a read-only database generate an error. See the “Error Codes and Messages” chapter of the **Language Reference Guide.**
- No metadata changes are allowed in read-only databases.

- Generators in a read-only database do not increment and are allowed only to return the current value. For example, in a read-only database, the following statement succeeds:

  ```
  SELECT GEN_ID(generator_name, 0) FROM table_name;
  ```

  The following statement fails with the error “attempted update on read-only database.”

  ```
  SELECT GEN_ID(generator_name, 1) FROM table_name;
  ```

- External files accessed through a read-only database open in read-only mode, regardless of the file’s permissions at the file system level.

Making a Database Read-only

To change the mode of a database between read-write and read-only, you must be either its owner or SYSDBA and you must have exclusive access to a database.

From within InterBase, you can change a read-write database to read-only mode in any of three ways:

- In IBConsole, select the database, display its properties, and edit the mode. For more information, refer to **“Setting Database Properties”** .
- Use **gbak** to back up the database and restore it in read-only mode:

  ```
  gbak -create -mode read_only foo.ibk foo.ib
  ```

- Use **gfix** to change the mode to read-only:

  ```
  gfix -mode read_only foo.ib
  ```

Important To set a database to read-only mode from any application that uses BDE, ODBC, or JDBC, use the *isc_action_svc_properties*() function in the InterBase Services API.

Tip To distribute a read-write database on a CD-ROM, back it up and put the *database.ibk* file on the CD-ROM. As part of the installation, restore the database to the user’s hard disk.

Read-only with Older InterBase Versions

- A pre-6 InterBase client can access a read-only database to perform SELECTs. No other operation succeeds.
- If a current InterBase client tries to set a pre-6 database to read-only mode, the server silently ignores the request. There is no way to make older databases read-only. You must upgrade them.

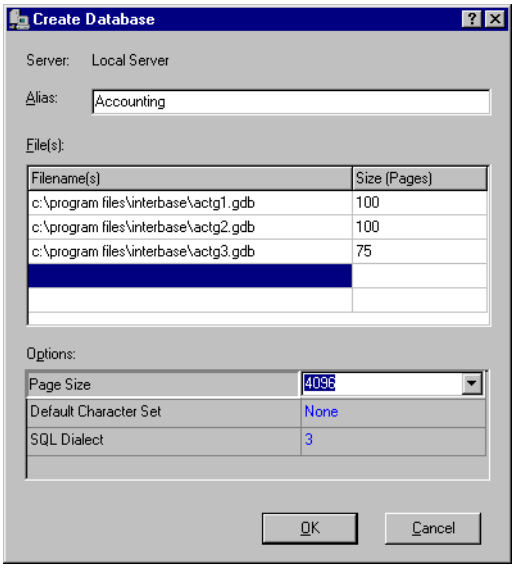
Creating Databases

You can create databases on local and remote servers using IBConsole with the Create Database dialog.

You can use any of the following methods to access the Create Database dialog:

- In the Tree pane, select a server or anywhere in the branch under the desired server and choose Database|Create Database.
- In the Tree pane, right click the Databases branch under the desired server, and select Create Database from the context menu.

Figure 7.1 Create Database Dialog



To Create a Database

1. Ensure that the server indicated is correct. If it is not, cancel this dialog and re-initiate it under the correct server.
2. Type an Alias name for the new database in the Alias text field.
3. Enter one or more filenames which will make up the database, specifying the number of pages required for each file. To insert a new row into the Files table, move to the last row and column of the table and type w-z.

   When entering a filename, make sure to include the file path unless you wish to default the file to the working directory.

   Note Database files must reside on a local drive.
4. You can specify create options by entering a valid value, by clicking the option value and choosing a new value from a drop-down list of values or by double-clicking the option value to rotate its value to the next in the list of values. For more information, see **“Database Options”** below.

   To create a basic database without any options, leave all options blank.
5. Click OK to create the specified database.

Important The alias name that you specify when creating a database references the necessary database file information associated with the database. When performing database configuration and maintenance, you need only specify the alias name, not the actual database filename. If the database spans multiple files, the server uses the header page of each file to locate additional files.

Database Options

The database options that you can set are Page Size, Default Character Set, and SQL dialect.

Page Size

InterBase supports database page sizes of 1024, 2048, 4096, 8192, and 16384 bytes. The default is 4096 bytes.

Default Character Set

See **"Character Set"** in Table 9.2 for a detailed explanation of character sets.

For more information about creating databases, see the *Language Reference*. See the ***Data Definition Guide*** for an explanation of character sets.

SQL Dialect

An InterBase database SQL dialect determines how double quotes, large exact numerics, and certain datatypes such as SQL DATE, TIME, and TIMESTAMP are interpreted. In most cases you should create databases in dialect 3 in order to have access to all current InterBase features.

Changing a database dialect from 1 to 3 may require some preparation if it contains DATE data types, DECIMAL or NUMERIC data types with precision greater than 9, or has strings that are in double quotes rather than single quotes. For more information about dialects, refer to *"Understanding SQL Dialects"* in the migration appendix of the *Operations Guide.*

To Change the Database Dialect

1. Highlight the database in the Tree pane and perform one of the following actions:
   - Choose Database|Properties.
   - Right-click and choose Properties from the context menu.
   - Double-click Properties in the Work pane.
2. Click the General tab and change the SQL dialect in the Options field.

Tip To suppress the display of system tables in IBConsole, deselect System Data from the View menu.

Dropping Databases

You can drop databases using IBConsole. Dropping a database deletes the current database and database alias, removing both data and metadata.

A database can be dropped only by its creator or SYSDBA.

To Drop a Database

1. Select the database you wish to drop in the Tree pane.
2. Choose Database|Drop Database or select Drop Database from the Work pane.
3. A dialog asks you to confirm that you wish to delete the database. Click Yes if you want to drop the selected database, otherwise click No.

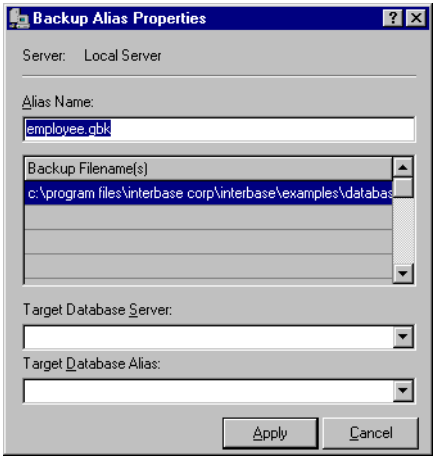
Important Dropping a database deletes all data and metadata in the database.

Backup File Properties

You can view and modify backup file information in IBConsole with the Backup Alias Properties dialog. You can access this dialog with either of the following methods:

- Expand Backup in the Tree pane, select a backup alias, and double-click Modify Backup Alias from the Work pane.
- Right-click a backup alias in the Tree pane and choose Modify Backup Alias from the context menu.

Figure 7.2 Backup Alias Properties



To Edit Backup File Properties

1. Enter a new backup alias name in the Alias Name text field.

2. Add, remove, or modify the backup filenames and corresponding file sizes associated with the backup in the backup files table. When specifying filenames, be sure to include the file path where the file is located.

   To add a new row to the backup files table, move to the last row and column of the table and type w-z. To remove a file from the backup file list, delete the values from the table.

3. Select a server from the Target Database Server drop-down list. You can also type the server name in the edit portion of the drop-down list.
4. Select a database alias from the Target Database Alias drop-down list. You can also type the alias name in the edit portion of the drop-down list
5. Click Apply to save your changes.

Removing Database Backup Files

You can remove database backup files in IBConsole with either of the following methods:

- Expand Backup in the Tree pane and select a backup alias and double-click Delete Alias from the Work pane.
- Right-click a backup alias in the Tree pane and choose Delete Alias from the context menu.

A dialog asks you to confirm that you wish to remove the selected backup file. Click Yes if you want to delete the backup file, otherwise click No.

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:

1. 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”**.

2 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 database administrator, depending on how the shadow was defined. For more information about activating a shadow, see **"Activating a Shadow"**.

3 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"**.

4 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"**.

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 file system 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 database administrator creates a shadow either in auto mode or in manual mode. For more information about these modes, see **“Auto Mode and Manual Mode”**.

- Automatic shadow creation

  To ensure that a new shadow is automatically created, create a conditional shadow. For more information, see **“Conditional Shadows,”** 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.ib*, 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.ib
   Shadow 1: '/usr/InterBase/examples/employee.shd' auto
  PAGE_SIZE 4096
  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 your database is large, you can shadow it to a multifile shadow, spreading the shadow files over several disks. To create a multifile shadow, specify the name and size of each file in the shadow set. As with multifile databases, you have the option of specifying the size of secondary files in either of two ways: specify the page on which each secondary file starts, or specify the length in database pages of each file. When you specify the size using the LENGTH keyword, do not specify the length of the final file. InterBase sizes the final file dynamically, as needed.

For example, the following example creates a shadow set consisting of three files. The primary file, *employee.shd*, is 10,000 database pages in length. The second file is 20,000 database pages long, and the final file grows as needed.

```
SQL> CREATE SHADOW 1 'employee.shd' LENGTH 10000
CON> FILE 'emp2.shd' LENGTH 20000
CON> FILE 'emp3.shd';
```

Instead of specifying the page length of secondary files, you can specify their starting page. The following example creates the same shadows as the previous example:

```
SQL> CREATE SHADOW 1 'employee.shd'
CON> FILE 'emp1.shd' STARTING AT 10000
CON> FILE 'emp2.shd' STARTING AT 30000;
```

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.ib
   Shadow 1: '/usr/InterBase/examples/employee.shd' auto
length 10000
   file /usr/InterBase/examples/emp1.shd length 2000
starting 10000
```

```
   file /usr/InterBase/examples/emp2.shd length 2000
starting 30000
  PAGE_SIZE 4096
  Number of DB pages allocated = 392
  Sweep interval = 20000
```

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 database administrator intervenes. The benefits of auto mode and manual mode are compared in the following table:

Table 7.1 Auto vs. Manual Shadows

Mode	Advantage	Disadvantage
Auto	Database operation is uninterrupted	Creates a temporary period when the database is not shadowed The database administrator 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 database administrator

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 might 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"**.

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, the database owner or SYSDBA must 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 creates 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, log in as SYSDBA or the database owner and use **gfix** with the **-activate** 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.ib* has a shadow named *employee.shd*, enter:

```
gfix -activate 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.

Setting Database Properties

The Database Properties dialog enables you to display and configure certain database settings. You can access the Database Properties dialog by any of the following methods:

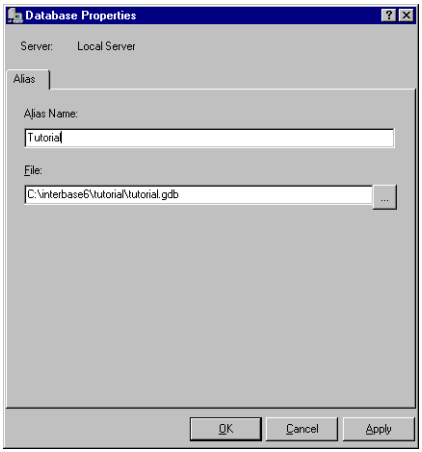
- Select a connected database (or any branch under the database hierarchy) in the Tree pane and choose Database|Properties.
- Select a connected database in the Tree pane and double-click Properties in the Work pane.
- Right-click a connected database in the Tree pane and choose Properties from the context menu.

The Database Properties dialog contains two tabs, Alias and General.

Alias Tab

The Alias tab of the Database Properties dialog is where you can specify an alias name for a database as well as the file path and file name of the selected database.

Figure 7.3 Database Properties: Alias tab



To edit database alias settings

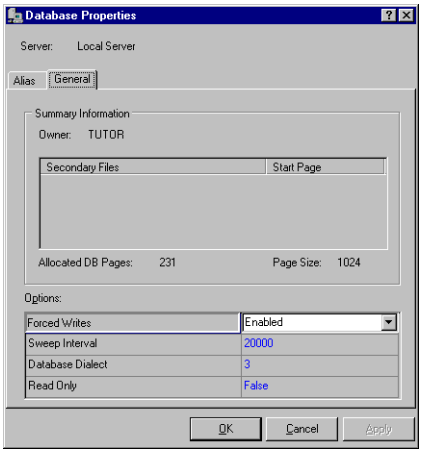
1. Enter the alias name of the database in the Alias Name text field.
2. Enter database file name, including the path where the file is located, in the File text field. If you prefer, you can also click the browse button ... to locate the file you want.

   If you want to change the database file name, the database must be disconnected before you access the Database Properties dialog.
3. If you need to view or configure the general database settings, click the General tab and see **"General Tab"** below for further information.
4. Once you are finished making changes to the database properties click Apply to save your changes, otherwise click Cancel.

General Tab

The General tab of the Database Properties dialog is where you can view such database settings as the database owner, secondary files and their start pages, the number of allocated database pages and the page size. You can also set such options as Forced Writes, Sweep Interval, SQL Dialect and Read Only.

Figure 7.4 Database Properties: General Tab



To edit database general options

1. Choose option values in the Options table. You can specify options by clicking the option value and entering a new value, by choosing a new value from a drop-down list of values or by double-clicking the option value to rotate its value to the next in the list of values.
2. If you need to view or configure the database alias settings, click the Alias tab and see **“Alias Tab”** above for further information.
3. Once you are finished making changes to the database properties click Apply to save your changes, otherwise click Cancel.

Table 7.2 General Options

Option	Value
Forced Writes	Option values are Enabled and Disabled. See **“Forced Writes vs. Buffered Writes”** for further information on forced writes.

Table 7.2 General Options

Option	Value
Sweep Interval	The sweep interval is the number of transactions that will occur before an automatic database sweep takes place. You can enter any positive number for the sweep interval, or zero to disable the automatic sweep. See **"Sweep Interval and Automated Housekeeping"** for further information on setting the sweep interval.
Database dialect	An InterBase database SQL dialect determines how double quotes, large exact numerics, and certain datatypes such as SQL DATE, TIME, and TIMESTAMP are interpreted. In most cases you should choose dialect 3 in order to have access to all current InterBase features.
Read Only	Option values are True and False. To make the database read only set the Read Only option to True. This prevents users from performing any DML or updates to the database. The default setting for this option is False. See **"Making a Database Read-only"** for more information.

Sweep Interval and Automated Housekeeping

Sweeping a database is a systematic way of removing outdated records. Periodic sweeping prevents a database from growing too large. In the past sweeping slowed system performance and users disabled the automatic database sweep function because of the impact on product operations.

InterBase databases periodically need to be swept. Otherwise the main memory allocated for each transaction's bitmap increases to the point where performance becomes unacceptable. The longer sweep takes to complete, the more main memory requirements increase for starting new transactions.

With the implementation of the fast sweep optimization in InterBase XE, the memory allocation issue has been mitigated. The user has the option to configure their databases for automatic sweep. In cases where large databases have large archival or infrequently modified tables, a database sweep will have minimal impact on the performance of running transactional operations.

Only ODS 15 and later databases can perform fast database sweeps. The effectiveness of a fast sweep is directly proportional to the fraction of database data pages that have been modified since the last sweep. If every data page has been changed, fast sweep is no faster than the former methodology. If very few pages are changed, fast sweep is nearly instantaneous. If half the pages were updated, fast sweep is then half the former sweep time.

There is no new user interface or action required by the user to enable this functionality. Manual sweep initiated by the GFIX command line tool, IBConsole, or programmatically, as well as automatic sweep configuration on a database, use

the fast sweep mechanism.

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

Overview of Sweeping

InterBase uses a multigenerational 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 by sweeping the database. This process frees up space allocated to outdated record versions. Whenever a transaction accesses a record, outdated versions of that record are collected. Records that were rolled back are not collected. To guarantee that all outdated records are 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). Automatic housekeeping occurs when the difference between the OIT and the oldest active transaction (OAT) is greater than the sweep interval. By default, this sweep interval is 20,000, but it is configurable (see **"Setting the Sweep Interval"**).

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 OAT 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 **"About InterBase backup and restore options"**.

Configuring Sweeping

You are able 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 a sweep menu command and is explained in **"Performing an Immediate Database Sweep"** .

Setting the Sweep Interval

To set the automatic sweep threshold to *n* transactions:

```
gfix -h n
```

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 can also increase. Lowering the sweep interval can help reduce the time for transaction start-up.

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

To set the sweep interval with IBConsole, refer to **"Setting Database Properties"** .

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 database administrator 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 threshold 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.

To disable automatic sweeping with IBConsole, refer to **"Setting Database Properties"** .

Performing an Immediate Database Sweep

You can perform an immediate database sweep with any of the following methods:

- Right click a connected database in the Tree pane and choose Maintenance| Sweep from the context menu.
- Select a connected database in the Tree pane and double-click Sweep in the Work pane.
- enter the following command:

```
gfix -sweep
```

This operation runs an immediate sweep of the database, releasing space held by records that 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 Database Cache

The *database cache* consists of all database pages (also called buffers) held in memory at one time. *Database cache size* is the number of database pages. You can set the default size of the database cache at three levels:

- Server level: applies to all databases
- Database level: applies only to a single database (using **gfix** or **ALTER DATABASE SET PAGE CACHE** to set the size for a specific database)
- Connection level: applies only to a specific **isql** connection

We recommend setting cache size at the database level rather than at the server level. This reduces the likelihood of inappropriate database cache sizes.

Every database on a server requires RAM equal to the cache size (number of database pages) times the page size. By default, the cache size is 2048 pages per database and the page size is 4KB. Thus, a single database running at the default setting requires 8MB of memory, but three such databases require 24MB of memory.

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 2048 pages.

The default size for a database can also be set using the ALTER DATABASE statement:

```
ALTER DATABASE SET PAGE CACHE n
```

To run **gfix** or **ALTER DATABASE**, you must be either SYSDBA or the owner of the database.

Both gfix and ALTER DATABASE immediately attempt to expand the cache buffers to the number of pages requested.

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* is trimmed to the database-specific cache setting if it is greater than that value.

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

InterBase API: use the *isc_dpb_num_buffers* parameter to set cache size in a database parameter buffer (DPB).

IBX: use the *num_buffers* parameter to set cache size in the TIBDatabase's parameter list. For example: *num_buffers*=250. For the parameter to be parsed correctly, there must be no spaces around the = sign.

The number of buffers passed by the InterBase API or IBX is trimmed to the database-specific cache setting if it is greater than that value.

Default Cache Size Per Server

For SuperServer installations, you can configure the default number of pages used for the database caches. By default, the database cache size is 2048 pages per database. You can modify this default by changing the value of DATABASE_CACHE_PAGES in the *ibconfig* configuration file. When you change this setting, it applies to every active database on the server.

You can also set the default cache size for each database using the **gfix** or **SET PAGE CACHE** utilities. This approach permits greater flexibility, and reduces the risk that memory is overused, or that database caches are too small.

We strongly recommend that you use **gfix** or **SET PAGE CACHE** to set cache size rather than DATABASE_CACHE_PAGES.

Verifying Cache Size

To verify the size of the database cache currently in use, execute the following commands in **isql**:

```
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 = 2048
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.

Note The example command listing shows "**Buffers = 2048**" for user to verify that cache setting has been changed. This is no longer strictly true. For very large cache buffer settings (>256MB), InterBase incrementally allocates additional cache buffers on-demand. So it is possible that the listed command will show a **Buffers** value that is a lower number. The actual value can always be determined by running **gstat -h** and examining the **Page buffers** entry or querying column RDB$PAGE_CACHE from system table RDB$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 that involve data modification are slower.

Forced writes are enabled or disabled in the Database Properties dialog. For more information, refer to **“Setting Database Properties”** .

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 cause data structures such as database pages and indexes to become broken or lost. These corrupt data structures can make committed data unrecoverable.

Validating a Database

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.

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:

```
bad parameters on attach or create database
- secondary server attachments cannot validate databases
```

To shut down a database, refer to the directions in **“Shutting Down a Database”** .

Validating a Database Using gfix

To validate a database using gfix, follow these steps:

1. Enter the following command:

   ```
   gfix -v
   ```

2. If you suspect you have a corrupt database, make a copy of your database using an OS command (**gbak** will not back up corrupt data).

3. Use the **gfix** command to mark corrupt structures in the copied database:

   ```
   gfix -mend
   ```

4. If **gfix** reports any checksum errors, validate and repair the database again, ignoring any checksum errors:

   ```
   gfix -validate -ignore
   ```

Note *Note*: InterBase supports true checksums only for ODS 8 and earlier.

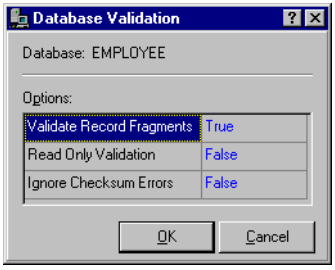
It may be necessary to validate a database multiple times to correct all the errors.

Validating a Database using IBConsole

To validate a database using IBConsole, access the Database Validation dialog by any of the following methods:

- Select a disconnected database in the Tree pane and double-click Validation in the Work pane.
- Right-click a disconnected database in the Tree pane and choose Validation from the context menu.
- Select Database|Maintenance|Validation.

Figure 7.5 Database Validation Dialog



To validate database

1. Check that the database indicated is correct. If it is not, cancel this dialog and re-initiate the Database Validation dialog under the correct database.
2. Specify which validation options you want by clicking in the right column and choosing True or False from the drop-down list. See **Table 7.3, "Validation Options"** for a description of each option.
3. Click OK if you want to proceed with the validation, otherwise click Cancel.

When IBConsole validates a database it verifies the integrity of data structures. Specifically, it does the following:

- Reports corrupt data structures
- Reports misallocated data pages
- Returns orphan pages to free space

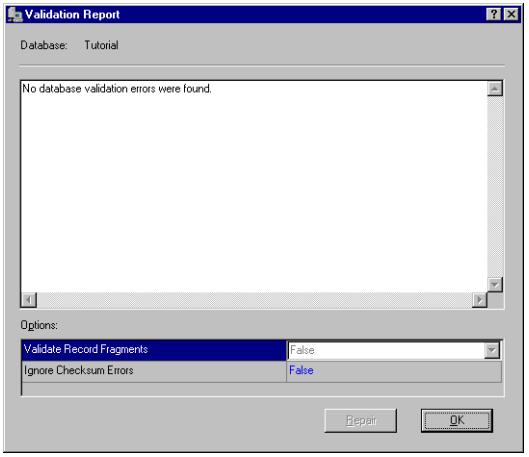
Table 7.3 Validation Options

Option	Value
Validate Record Fragments	Option values are True and False. By default, database validation reports and releases only page structures. If the Validate Record Fragments option is set to True, validation reports and releases record structures as well as page structures.
Read Only Validation	Option values are True and False. By default, validating a database updates it, if necessary. To prevent updating, set the Read Only Validation option to True.
Ignore Checksum Errors	Option values are True and False. 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, set the Ignore Checksum Errors option to True. This enables IBConsole to ignore checksums when validating a database. Ignoring checksums allows successful validation of a corrupt database, but the affected data may be lost. **Note**: InterBase supports true checksums only for ODS 8 and earlier.

Repairing a Corrupt Database

If a database contains errors, they are displayed in the following dialog:

Figure 7.6 Validation Report Dialog



The errors encountered are summarized in the text display 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 marks corrupt structures. In subsequent operations (such as backing up), InterBase ignores the marked records.

Some corruptions are too serious for IBConsole to correct. These include corruptions to certain strategic structures, such as space allocation pages. In addition, IBConsole cannot fix certain checksum errors that are random by nature and not specifically associated with InterBase.

Note 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.

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 IBConsole Backup utility or the **gbak** command, because they cannot back up a database containing corrupt data. If IBConsole reports any checksum errors, validate and repair the database again, setting the Ignore Checksum Error option to True. *Note*: InterBase supports true checksums only for ODS 8 and earlier.
2. It may be necessary to validate a database multiple times to correct all the errors. Validate the database again, with the Read Only Validation option set to True.

3. Back up the mended database with IBConsole or **gbak**. At this point, any damaged records are lost, since they were not included during the back up. For more information about database backup, see **"Database Backup and Restore."**
4. Restore the database to rebuild indexes and other database structures. The restored database should now be free of corruption.
5. To verify that restoring the database fixed the problem, validate the restored database with the Read Only Validation option set to True.

Shutting Down and Restarting Databases

Maintaining a database often involves shutting it down. Only the SYSDBA or the owner of a database (the user who created it) can shut it down. The user who shuts down the database then has exclusive access to the database.

Exclusive access to a database is required to:

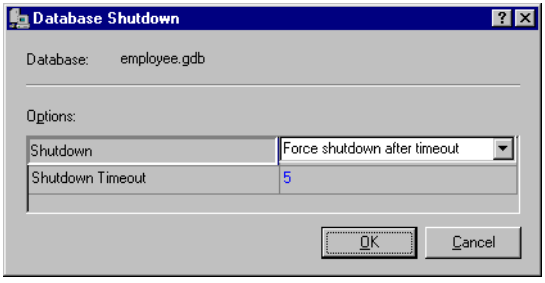
- Validate and repair the database.
- Add or drop a foreign key on a table in the database.
- Add a secondary database file.

After a database is shut down, the database owner and SYSDBA are still able to connect to it, but any other user attempting to connect gets an error message stating that the database is shut down.

Shutting Down a Database

To shut down a database, select a connected database from the Tree pane and double-click Shutdown in the Work pane or choose Database|Maintenance|Shutdown to display the Database Shutdown dialog:

Figure 7.7 Database Shutdown Dialog



Shutdown Timeout Options

You can specify a timeout value by selecting a new value from the drop-down list of values or by typing the value in the edit portion of the drop-down list. Timeout values can range from 1 minute to 500 minutes.

Shutdown Options

You can specify shutdown options by selecting a new value from the drop-down list of values. Shutdown option values include: Deny New Connections While Waiting, Deny New Transactions While Waiting, and Force Shutdown After Timeout.

Deny new connections while waiting

This option allows all existing database connections to complete their operations unaffected. IBConsole shuts down the database after all processes disconnect from the database. At the end of the time-out period, if there are still active connections, then the database is not shut down.

This 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.ib* 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.ib* 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.

Deny new transactions while waiting
This option allows existing transactions to run to normal completion. Once transaction processing is complete, IBConsole shuts down the database. Denying new transactions also denies new database connections. At the end of the time-out period, if there are still active transactions, then the database is not shut down.

This 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.ib*. 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.ib* 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 is happening at 6 p.m and instructs 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, IBConsole cancels the shutdown. Since the SYSDBA needs to perform database maintenance, and has sent out numerous warnings that a shutdown is about to occur, there is no choice but to force a shutdown.

Force Shutdown After Timeout
With this option, there are no restrictions on database transactions or connections. As soon as there are no processes or connections to the database, IBConsole shuts down the database. At the end of the time-out period, if there are still active connections, IBConsole rolls back any uncommitted transactions, disconnects any users, and shuts down the database.

If critical database maintenance requires a database to be shut down while 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.

This option does not deny new transactions or connections during the time-out period. If, at any time during the time-out period, there are no connections to the database, IBConsole shuts 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.

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, select a previously shut down database from the Tree pane and choose Database|Maintenance|Database Restart or double-click Database Restart in the Work pane. 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.

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 IBConsole. *Recovering* a limbo transaction means committing it or rolling it back. Use **gfix** to recover transactions.

Recovering Transactions

You can recover transactions by any of the following methods:

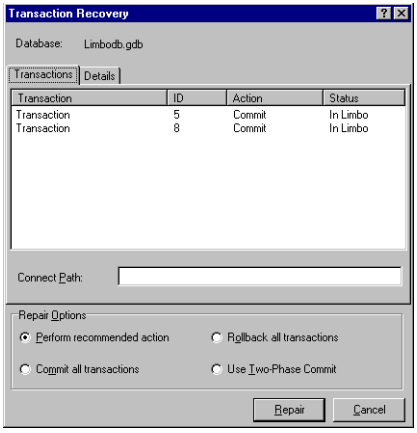
- Select a connected database in the Tree pane and double-click Transaction Recovery in the Work pane or choose Database|Maintenance|Transaction Recovery.
- Right-click a connected database in the Tree pane and choose Maintenance| Transaction Recovery from the context menu.

The Transaction Recovery dialog contains two tabs, Transactions and Details. The Transactions tab displays a list of limbo transactions that can then be recovered—that is, to committed or rolled back. You can also seek suggested recovery actions and set current actions to perform on the selected limbo transactions. The Details tab displays detailed information about a selected transaction.

Transaction Tab

All the pending transactions in the database are listed in the text area of the Transactions tab. You can roll back, commit, or perform a two-phase commit on such transactions.

Figure 7.8 Transaction Recovery: Limbo Transactions



To recover Limbo Transactions

1. Select a limbo transaction in the table.
2. The Connect Path text field displays the current path of the database file for the selected transaction, if it is a multi-database transaction. You can change the target database path, if necessary, by overwriting the current path.

   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 IBConsole. Before attempting to roll back or commit any transaction, confirm the path of all involved databases is correct.

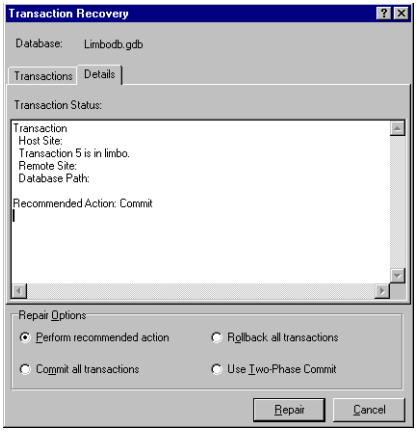
When entering the current path, be sure to include the server name and separator indicating communication protocol. To use TCP/IP, separate the server and directory path with a colon (:). To use NetBEUI, precede the server name with either a double backslash (\\) or a double slash (//), and then separate the server name and directory path with either a backslash or a slash.

3 If you want to continue with the transaction recovery process select a repair option and click Repair, otherwise click Cancel. To determine the recommended action, click on the transaction and select the Details tab. For further information about transaction recovery suggestions, see **"Details Tab"** below.

Details Tab

The Details tab displays the host server, the remote server, database path, and recommended action: either commit or rollback. If you want to continue with the transaction recovery process select a repair option and click Repair, otherwise click Cancel.

Figure 7.9 Transaction Recovery: Details

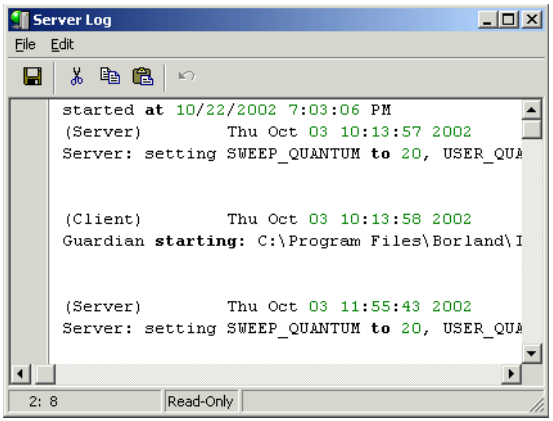


Viewing the Administration Log

IBConsole displays the administration log file in a standard text display window, the Administration Log dialog, which can be accessed by any of the following methods:

- Select a server (or any branch under the server hierarchy) in the Tree pane and choose Server|View Logfile.
- Right-click the desired server in the Tree pane and choose View Logfile from the context menu.
- Under the desired server, select Server Log in the Tree pane and then double-click View Logfile in the Work pane.

Figure 7.10 Administration Log Dialog



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 **“Standard Text Display Window” .**

gfix Command-line Tool

The **gfix** tool performs a number of maintenance activities on a database, including the following:

- Database shutdown
- Changing database mode to read-only or read-write
- Changing the dialect of a database
- Setting cache size at the database level
- Committing limbo transactions
- Mending databases and making minor data repairs
- Sweeping databases
- Displaying, committing, or recovering limbo transactions

To run **gfix**, you must attach as either SYSDBA or the owner of the database. Most of these actions can also be performed through IBConsole.

Syntax `gfix [options] db_name`

Options In the OPTION column of the following table, only the characters outside the brackets ([ ]) are required. You can specify 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.

Table 7.4 gfix Options

Option	Task	Description
-ac[tivate]	Activate shadows	Activate shadows when the database dies. NOTE: syntax is gfix -ac (no database name)
-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	Sets default cache buffers for the database to *n* pages
-c[ommit] {ID | all}	Transaction recovery	Commits limbo transaction specified by **ID** or commit all limbo transactions
-force ***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
-fu[ll]	Data repair	Used with **-v** to check record and page structures, releasing unassigned record fragments
-h[ousekeeping] ***n***	Sweeping	Changes automatic sweep threshold to *n* transactions • Setting *n* to 0 disables sweeping • Default threshold is 20,000 transactions (see “Overview of Sweeping”) • Exclusive access not needed
-i[gnore]	Data repair	Ignores checksum errors when validating or sweeping; InterBase supports true checksums only for ODS 8 and earlier.
-k[ill]	Drop shadows	• Drops unavailable shadows. • Syntax is **gfix -k** (no database name)
-l[ist]	Transaction recovery	Displays IDs of each limbo transaction and indicates what would occur if **-t** were used for automated two-phase recovery
-m[end]	Data repair	Marks corrupt records as unavailable, so they are skipped (for example, during a subsequent backup)
-mo[de] [read_write|read_only}	Set access mode	• Sets mode of database to either read-only or read-write • Default table mode is read_write • Requires exclusive access to the database

Table 7.4 gfix Options

Option	Task	Description
-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	Checks for password *text* before accessing a database
-pr[ompt]	Transaction recovery	Used with **-l** to prompt for action during transaction recovery
-r[ollback] {ID | all}	Transaction recovery	Rolls back limbo transaction specified by **ID** or roll back all limbo transactions
-sh[ut]	Shutdown	• Shuts down the database • Must be used in conjunction with **-attach**, **-force**, or **-tran**
-sq[l_dialect] ***n***	Database dialect	Changes database dialect to *n* • Dialect 1 = InterBase 5.*x* compatibility • Dialect 3 = Current InterBase with SQL92 features
-sw[eep]	Sweeping	Forces an immediate sweep of the database • Useful if automatic sweeping is disabled • Exclusive access is not necessary
-tr[an] ***n***	Shutdown	Used with **-shut** to prevent new transactions from starting during timeout period of *n* seconds; cancels shutdown if there are still active transactions after *n* seconds
-tw[o_phase] {ID | all}	Transaction recovery	Performs automated two-phase recovery, either for a limbo transaction specified by **ID** or for all limbo transactions
-user ***name***	Remote access	Checks for user *name* before accessing a remote database

Table 7.4 gfix Options

Option	Task	Description
-v[alidate]	Data repair	Locates and releases pages that are allocated but unassigned to any data structures; also reports corrupt structures
-w[rite] {sync | async | direct}	Database writes	Database writes Enables or disables forced (synchronous) writes. **sync** enables forced writes; **async** enables buffered writes; **direct** enables direct I/O.
-z		Shows version of **gfix** and of the InterBase engine

Examples The following example changes the dialect of the *customer.ib* database to 3:

```
gfix -sql 3 customer.ib
```

The following example changes the customer.ib database to read-only mode:

```
gfix -mo read_only customer.ib
```

gfix Error Messages

Table 7.5 gfix Database Maintenance 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.
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.

Table 7.5 gfix Database Maintenance Error Messages

Error Message	Causes and Suggested Actions to Take
Transaction number or “all” required	You specified **-commit**, **-rollback**, or **-two_phase** without supplying the required argument.
-mode read_only or read_write	The **-mode** option takes either read_only or read_write as an option.
“read_only” or “read_write” required	The **-mode** option must be accompanied by one of these two arguments.

Chapter

8

Database Backup and Restore

The purpose of a **database backup** is to protect and preserve data in case of database or machine failure. A **database restore** uses the backup file to recreate the database.

A database backup saves a database to a file on a hard disk or other storage medium. InterBase provides full and incremental backup capability, as well a number of options that allow you to tailor your backups to suit a iety of scenarios.

To most effectively protect your database from power failure, disk crashes, or other potential data loss, perform backups on a regular basis. For additional safety, store the backup file in a different physical location from the database server.

This chapter explains how to perform full and incremental backups on InterBase machines. It also explains how to restore InterBase databases.

About InterBase backup and restore options

You can use InterBase to backup data to the following locations:

- To a second drive on the InterBase server.
- To another machine on the local network.
- To a machine at a remote location using a VPN or WAN.

InterBase backups can **run concurrently** while other users are using the database. You do not have to shut down the database to run a backup. However, any data changes that clients commit to the database after the backup begins are not recorded in the backup file.

InterBase backup and restore tools

You can use either of the following InterBase tools to backup and restore InterBase databases:

- The **gbak** command-line tool

  Use the InterBase gbak command to specify and execute backup and restore operations from a Windows or Unix command line. Familiarity with isql, InterBase's version of SQL is recommended. isql provides a number of options to help tailor your backup and restore to suit different circumstances and environments.

- The **IBConsole**

  The IBConsole's intuitive user interface allows you to use your mouse, context menus, and dialog boxes to specify the type of backup and restore you want to perform. The same backup and restore options that are available using gbak are available through the IBConsole user interface. You do not need to be familiar with command-line operations or with SQL or isql to use IBConsole.

This chapter explains how to use both tools to perform backups and restores.

The difference between logical and physical backups

InterBase uses the gbak command to perform backups. The gbak command makes the following distinctions between backup types.

- **Logical:**

  The full backup typically performed by gbak is a "logical" backup. It extracts every record in the database and stores it in a different format. Thus, it is not an exact replica of the database file. Logical backups **reclaim space** occupied by deleted records, thereby reducing database size.

  A logical backup, performed by a gbak client, can save the target backup file anywhere on the network; you do not have to have an InterBase server on the client machine. If the backup is performed using the Services API, then the backup file can only be written to file systems that are accessible to the server (since the server is performing the backup operation).

  When executing a logical backup with gbak, use the following syntax:

  gbak [-b] [options] *database target*

  If you choose the **Backup** option using IBConsole, this is the type of backup InterBase executes.

  Restoring from logical backups gives you the option of changing the database page size and distributing the database among multiple files or disks.

- **Physical:**

InterBase's physical backup, also referred to as an online dump, copies the database at the page level and saves it in its original format. Thus, a physical backup creates an exact replica of the database during backup process. You can convert the replica to a read-write database, though if you do so, you will no longer be able to dump to the replica from the original database.

To perform a physical backup, use the following syntax:

gbak [-d] [options] *database target*

Notice that the physical backup uses the -d switch rather than the -b switch that is specified in the logical backup.

An **incremental backup** copies all of the changes that have been committed to the database since the last full backup. An incremental backup is a physical backup and uses the -d switch. The first time you use the gbak -d switch, InterBase performs a full physical backup (an online dump). After the initial full dump, each subsequent backup using the -d switch performs an incremental backup, saving and copying all of the transactions committed since the last full backup.

If you choose the **Incremental Backup** option using IBConsole, IBConsole performs an initial full online dump using the -d switch. All subsequent backups using the -d switch are incremental.

Important To add an additional level of database protection, use journal files and journal archiving. Journal files record each database transaction as it occurs, even those that occur when a backup is running. A journal archive stores current journal files. You can use a journal archive to recover data to a specific point in time. For more information about journaling and journal archives, see **"Journaling and Disaster Recovery."**

Database ownership

Although backing up a database can be performed only by the owner or SYSDBA, any user can restore a database as long as they are not restoring it over an existing database. A database file restored from a logical backup belongs to the user ID of the person who performed the restore. This means that backing up and restoring a database is a mechanism for changing the ownership of a database. It also means that an unauthorized user can steal a database by restoring a backup file to a machine where he knows the SYSDBA password. It is important to ensure that your backup files are secured from unauthorized access.

To restore a database over an existing database, you must be SYSDBA or the owner of the existing database.

Restoring the ODS

InterBase automatically restores the database to the latest on-disk structure (ODS). If the database uses an earlier ODS, errors may occur. To avoid this, keep your databases updated by using the latest version of the ODS.

To upgrade existing databases to a new ODS, perform the following steps:

1. Before installing a new version of InterBase, back up databases using the old version.
2. Install the InterBase server.
3. Once the new server is installed, restore the databases.

For more information about migrating to a later version of InterBase, see Appendix A, **“Migrating to InterBase.”**

Performing backups and restores using the gbak command

You can use the **gbak** command-line tool to perform database backups and restores, using different options to specify different outcomes. For instruction on how to use IBConsole for backups and restores, see “Performing backups and restores using IBConsole,” later in this chapter.

Note For information on how to use the gbak command to encrypt backup files and to restore encrypted backup files, see Chapter 14, “Encrypting Your Data,” in the *Data Definition Guide*.

General guidelines for using gbak

When backing up a database, keep the following information in mind:

- Unless the **-service** option is specified, **gbak** writes the backup files to the current directory of the machine on which it is running, not on the server where the database resides. If you specify a location for the backup file, it is relative to the machine where **gbak** is executing. You can write the backup files only to this local machine or to drives that are mapped to it. Note that the **-service** switch changes this behavior. (See **“Using gbak with InterBase Service Manager”** .)
- When you are backing up a multifile database, specify *only* the first file in the backup command. You must not name the subsequent database files: they will be interpreted as backup file names.
- The default unit for backup files is bytes. You can choose to specify kilobytes, megabytes, or gigabytes (**k**, **m**, or **g**) instead. Restored database files can be specified only in database pages.
- Use the **-transportable** switch if you operate in a multiplatform environment. This switch permits the database to be restored to a platform other than the one on which it originally resided. Using this option routinely is a good idea when you are operating in a multiplatform environment.

- Use the **-service** switch if you are backing up to the same server that holds the original database. This option invokes the InterBase Service Manager on the server host and saves both time and network traffic.

Tip It is good security practice to change your backup files to read-only at the system level after creating them. This prevents them from being accidentally overwritten. In addition, you can protect your databases from being "kidnapped" on UNIX and Windows systems by placing the backup files in directories with restricted access.

Initiating multi- and single-file backups

When backing up a multifile database, specify only the first file name of the database.

Syntax **For backing up to a single file**

```
gbak [-b] [options] database target
```

For backing up to multiple files

```
gbak [-b] [options] database target1 size1[k|m|g] target2
[size2[k|m|g] target3
```

Table 8.1
Table 8.2 Arguments for gbak

Argument	Description
database	• Name of a database to back up • For a multifile database, the name of the first database file
target	Name of a storage device or backup file to which to back up • On UNIX, can also be *stdout*, in which case **gbak** writes its output to the standard output (usually a pipe) • No size need be specified when restoring to a single file, since the database always expands as needed to fill all available space
size	Length of a backup file or restored database file • The only permissible unit for a restored database file is database pages; minimum value is 200 • Default unit for a backup file is bytes • Size of backup files can also be specified in kilobytes, megabytes, or gigabytes • Do not specify a size for the final backup file or database file; the last file always expands as needed to fill all available space

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.

Table 8.3 Backup Options for gbak

Option	Description
-b[ackup_database]	Backs up database to file or device
-co[nvert]	Converts external files as internal tables
-d[ump]	The first time you use the -d switch, it executes a full physical backup. Subsequent uses execute an incremental backup.
-e[xpand]	Creates a noncompressed back up
-fa[ctor] ***n***	Uses blocking factor *n* for tape device
-g[arbage_collect]	This option instructs the server not to perform garbage collection on every record it visits. This enables the server to retrieve records faster, and to send them to the gbak client for archiving.
-ig[nore]	Ignores checksums during backup; *Note*: InterBase supports true checksums only for ODS 8 and earlier
-l[imbo]	Ignores limbo transactions during backup
-m[etadata]	Backs up metadata only, no data
-nt	Creates the backup in nontransportable format
-ol[d_descriptions]	Backs up metadata in old-style format
-pas[sword] ***text***	Checks for password *text* before accessing a database
-role ***name***	Connects as role *name*

Table 8.3 Backup Options for gbak (*continued*)

Option	Description
-se[rvice] ***servicename***	• Creates the backup files on the host where the original database files are located, using InterBase's Service Manager • *servicename* invokes the Service Manager on the server host; syntax varies with the network protocol in use: TCP/IP `hostname:service_mgr` Named pipes `\\hostname\service_mgr` Local `service mgr` **IMPORTANT NOTE**: If you are providing file path names with embedded spaces in them and using the InterBase service manager (-service switch in GBAK), you will need to multi-quote the file names: <double_quote><single_quote>filepath<single_quote><double_quote> The above is required because the command shell strips away the external double_quotes and only leaves the internal single_quotes for InterBase to know that it is a single string value. For example: # gbak –service service_mgr –r "'/path/with space/foo.ibk'" "'/path/with space/foo.ib'" –user sysdba –password masterkey
-t[ransportable]	Creates a transportable backup [default]
-user ***name***	Checks for user *name* before accessing remote database
-v[erbose]	Shows 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

Creating incremental backups

An incremental backup copies all of the changes to the database that have occurred since the last full or incremental backup. The first time you use the gbak -d switch, InterBase performs a full physical backup (an online dump). After the initial full dump, the -d switch performs an incremental backup, saving and copying all of the transactions committed since the last full backup.

Incremental backup guidelines

When specifying an incremental backup, keep the following information in mind:

- Performing an incremental online dump still requires a full scan of the source database.
- The performance improvement accrues from limiting the number of page writes to the online dump files, especially if those files are located on a remote file server.
- Multiple online dumps of the same or distinct databases can be run concurrently though this would not be recommended for performance reasons.
- An active online dump can be cancelled by the InterBase Performance Monitor or killing the GBAK process.
- External tables are not backed up by an online dump.
- External tables may not be accessible if the online dump is attached as a read-only database. If the external file pathnames can't be accessed from the online dump's location, there is no way to modify the dump's metadata without making the dump a read-write database. If it is made a read-write database, it can no longer be a target for online dump again.
- Since an online dump is a physical backup, the online dump files are not transportable to other hardware platforms. To make the backup transportable, use gbak's traditional logical backup using the -t switch.
- When a CREATE JOURNAL ARCHIVE statement is executed, InterBase uses the online dump feature to copy the database to a journal archive directory. For more information about journal files and journal archiving, see **"Journaling and Disaster Recovery."**

Executing an incremental backup

To execute an incremental backup, use the following syntax:

```
  GBAK {-D} dbname   file [size] add_file1 [size1] add_file2
[size2] ...
```

The first dump file in the list is similar to the first database file in a multi-file database. It is the file that is used as a reference to an existing online dump. If there are additional dump files listed on the GBAK command line, those files are added to the set of files in the online dump.

Example The following example can assist you in creating an initial incremental online dump.

```
  [E:/EMPLOYEE] gbak -d EMPLOYEE.gdb EMPLOYEE.gdmp
EMPLOYEE.gdmp.1
  gbak: WARNING: Dumped 46270 pages of a total 46270
database pages
  gbak: WARNING:     Dumped 1 pages to page appendix file
```

```
  [E:/EMPLOYEE] gbak -d EMPLOYEE.gdb EMPLOYEE.gdmp
EMPLOYEE.gdmp.1
  gbak: ERROR: I/O error for file "E:\EMPLOYEE\
EMPLOYEE.GDMP.1"
  gbak: ERROR:     Error while trying to create file
  gbak: ERROR:     The file exists.

  gbak: Exiting before completion due to errors

  [E:/EMPLOYEE] gbak -d EMPLOYEE.gdb EMPLOYEE.gdmp
EMPLOYEE.gdmp.2
  gbak: WARNING: Dumped 2 pages of a total 46270 database
pages
  gbak: WARNING:     Dumped 0 pages to page appendix file
```

In the example above, EMPLOYEE.gdmp.1 was added in the course of a full database dump.

Re-executing the command gives an error because it tries to add EMPLOYEE.gdmp.1 again causing a file creation error. The last command adds a new file EMPLOYEE.gdmp.2 successfully.

The online dump files can be on either a local or a remote file system that is writable by the InterBase server. An online dump is a server-side operation only. While the online dump files can be located on any mounted file system, the page appendix file is always on the local file system. This file is written to by concurrent server threads handling client requests when it is necessary to preserve the state of page's image for the online dump. This is analogous to InterBase's multigenerational architecture (MGA) where a previous version of a row is stored when updating a row to preserve a transaction's snapshot. The page appendix file helps to maintain the physical page snapshot of the online dump. It is a temporary file and is deleted when the online dump completes.

The [size] parameter is optional and denotes the file's size in units of pages, using the database's page size. If the [size] parameter is not provided then that dump file's size will be determined by its file-sequenced counterpart in the database. If the dump file's sequence is higher than the sequence of any database file then it takes the size of its predecessor dump file.

If you run GBAK -D against an existing online dump, an incremental dump will be created.

```
  [E:/EMPLOYEE] gbak -d EMPLOYEE.gdb EMPLOYEE.gdmp
  gbak: WARNING: Dumped 46270 pages of a total 46270
database pages
  gbak: WARNING:     Dumped 23 pages to page appendix file

  [E:/EMPLOYEE] gbak -d EMPLOYEE.gdb EMPLOYEE.gdmp
  gbak: WARNING: Dumped 2 pages of a total 46270 database
pages
```

```
  gbak: WARNING:     Dumped 0 pages to page appendix file
```

This updates the online dump with only those pages that have changed since the last dump. An incremental dump can always be retried if it fails. If a full online dump fails, InterBase will delete the online dump files that were written prior to the failure. If InterBase cannot access those files because of the failure, those online dump files will have to be deleted manually.

Over-writing Incremental backups

The -OV overwrite switch causes the current set of online dump files to be deleted, and initiates a full database dump.

```
  [E:/EMPLOYEE] gbak -d EMPLOYEE.gdb EMPLOYEE.gdmp
  gbak: WARNING: Dumped 2 pages of a total 46270 database
pages
  gbak: WARNING:     Dumped 1 pages to page appendix file

  [E:/EMPLOYEE] gbak -d -ov EMPLOYEE.gdb EMPLOYEE.gdmp
  gbak: WARNING: Dumped 46270 pages of a total 46270
database pages
  gbak: WARNING:     Dumped 7 pages to page appendix file
```

The online dump files are marked as a read-only InterBase database. This means that it can be accessed by read-only database applications. It is undefined how such database applications will behave if they access the online dump “database” while the dump files are being incrementally updated. If an online dump is converted to read-write, it ceases to be an online dump and becomes a standalone database. Attempting to perform an online dump against it will fail.

```
  [E:/EMPLOYEE] gfix EMPLOYEE.gdmp -mode read_write

  [E:/EMPLOYEE] gbak -d EMPLOYEE.gdb EMPLOYEE.gdmp
  gbak: ERROR: online dump failure: dump file has no dump
timestamp
  gbak: Exiting before completion due to errors

  [E:/EMPLOYEE] gfix EMPLOYEE.gdmp -mode read_only

  [E:/EMPLOYEE] gbak -d EMPLOYEE.gdb EMPLOYEE.gdmp
  gbak: ERROR: online dump failure: dump file has no dump
timestamp
  gbak: Exiting before completion due to errors
```

The online dump can be converted to a read-write database by using the gfix _mode read_write command and used in place. If the current location is not convenient for database processing, then online dump can be run against these

dump files to copy them somewhere else local or remote. This provides a general copy mechanism that allows multifile databases to be copied and have their internal secondary file name links automatically updated for the copy destination.

Database validation (GFIX -V) can be run against an online dump because it is a database. An additional validation check is performed against an online dump, which checks that no database page has a write timestamp greater than that of the online dump timestamp. The online dump timestamp represents that last time a full or incremental dump succeeded.

```
  [E:/EMPLOYEE] gfix -v -n EMPLOYEE.gdmp
  Summary of validation errors

  Number of database page errors  : 1
  and in the InterBase log file:.
  IBSMP (Server)    Sat Jun 24 14:41:36 2006
  Database: E:\EMPLOYEE\EMPLOYEE.GDMP
  Page 155 has timestamp 1151170444 greater than dump
timestamp 1151170438
```

Timestamp Changes

GSTAT -H has been modified to list the online dump timestamp after the database creation date entry. Note that the database creation date is that of the source database and not the online dump.

```
  [E:/EMPLOYEE] gstat -h EMPLOYEE.gdmp
  Database "EMPLOYEE.gdmp"
  Database header page information:
        Flags                   0
        Checksum                12345
        Write timestamp         Jun 28, 2006 19:57:41
        Page size               4096
        ODS version             12.0
        Oldest transaction      72
        Oldest active           73
        Oldest snapshot         73
        Next transaction        74
        Sequence number         0
        Next attachment ID      0
        Implementation ID       16
        Shadow count            0
        Page buffers            0
        Next header page        0
        Clumplet End            102
        Database dialect        3
        Creation date           Jun 25, 2006 13:22:10
        Online dump timestamp   Jun 28, 2006 19:59:16
        Attributes              read only
```

Variable header data:

```
Dump file length: 20000
*END*
```

You can request an online dump by passing a string of database parameter blocks to the isc_attach_database() API.

Database parameter blocks used by an incremental backup

Incremental backup support has been added to the gbak utility using database parameter blocks (DPB). All general requirements and restrictions for DPB construction are documented in the *API Guide* in the Working with Databases chapter. Table 4.2 refers to DPB Parameters Groups by Purpose. Table 4.3 covers the Alphabetical List of DBP Parameters.

A successful online dump returns a warning status vector to pass back dump information status:

```
status [0]  = isc_arg_gds
status [1]  = isc_arg_success
status [2]  = isc_arg_warning
status [3]  = isc_old_dump_stats
status [4]  = isc_arg_number
status [5]  = <no. of dumped pages>
status [6]  = isc_arg_number
status [7]  = <total no. of DB pages>
status [8]  = isc_arg_gds
status [9]  = isc_old_appendix_stats
status [10] = isc_arg_number
status [11] = <no. pages written to appendix>
status [12] = isc_arg_end
```

Page Appendix File

When an online dump is running, client worker threads never write to the online dump files. Thus, their performance is not degraded by writing over the network to a remote file system. However, to maintain physical and time consistency of the dump, client worker threads may write pages to a local temporary file with a prefix of “ib_dump_”. Any database page is guaranteed to be written at most one time to this temporary file. This temporary file is known as the dump or page appendix file.

For very large databases with intensive update activity, the page appendix file could also grow to a very large size. There must be adequate space in the temp directories to handle this storage demand or the online dump will fail. The dump information returned to GBAK about the number of pages written to the appendix file can aid configuration of the temp file space.

Preallocating database space with gbak

GBAK backs up and restores database preallocation information. This preallocation information will be silently ignored by earlier versions of the product that are not aware of the feature. A switch has been added to -gbak to alter the stored preallocation in a database or backup file.

Example

```
  /D/testbed>isql
  Use CONNECT or CREATE DATABASE to specify a database
  SQL> create database 'pr.ib' preallocate 500;
  SQL> commit;
  SQL> quit;
  /D/testbed>ls -l pr.ib
  -rwxrwxrwx 1 Administrators None 2048000 Jul 2 18:09
  pr.ib /* It is 2MB size because each of the 500 database
pages is
  4KB in size */
  /D/testbed>isql -a pr.ib
  SET SQL DIALECT 3;
  /* CREATE DATABASE 'pr.ib' PREALLOCATE 500
  PAGE_SIZE 4096
  */
  /* Grant permissions for this database */
  /D/testbed>isql -x pr.ib
  SET SQL DIALECT 3;
  /* CREATE DATABASE 'pr.ib' PREALLOCATE 500 PAGE_SIZE 4096
  */
  /* Grant permissions for this database */
  /D/testbed>
```

Using the switch -PR(EALLOCATE) argument

The switch -PR(EALLOCATE) uses an integer argument, which consists of the number of preallocation pages. This switch is legal for both backup and restore command-line options. For backup, the preallocation switch stores its argument in the backup file instead of the value specified in the database that is being backed up. For restore, the preallocation switch argument is used at the preallocation value in the restore database, instead of the value stored in the backup file.

A GBAK preallocate switch value of 0 (zero) effectively disables database preallocation in the backup file or restored database. In GBAK verbose mode, database preallocation is logged to the console. The example below shows a sample database backup in verbose mode. A similar message is logged for database restore.

Example

```
gbak -v foo.gdb foo.gbk -pr 5000
    ...
    gbak: readied database foo.gdb for backup
```

```
gbak: creating file foo.gbk
gbak: starting transaction
gbak: database foo.gdb has a page size of 4096 bytes.
gbak: database preallocation 5000 pages
```

If a database restore reduces the page size, the number of pages for database preallocation is automatically scaled upward to maintain a constant database preallocation size in bytes. If the restored page size is increased, database preallocation is reduced accordingly with a similar “Reducing” message written to the console. If the GBAK -PREALLOCATE switch was given, then the automatic scaling of the database preallocation does not occur with changing page size. In other words, the -PREALLOCATE switch takes precedence.

Example

```
gbak -v foo.gdb foo.gbk -page_size 2048
...
      Reducing the database page size from 4096 bytes to
2048 bytes
      Increasing database preallocation from 5000 pages to
10000 pages
      created database foo1.gdb, page_size 2048 bytes
      database preallocation 10000 pages
```

Restoring a database with gbak

Use the following syntax to restore a database:

Syntax **For restoring:**

```
gbak {-c|-r} [options] source dbfile
```

For restoring to multiple files:

```
gbak {-c|-r} [options] source dbfile1 size1 dbfile2 [size2
dbfile3 …]
```

For restoring from multiple files:

```
gbak {-c|-r} [options] source1 source2 [source3 …] dbfile
```

By extension, you can restore *from* multiple files *to* multiple files using the following syntax:

```
gbak {-c|-r} [options] source1 source2 [source3 …] dbfile1
size1
  dbfile2 [size2 dbfile3 …]
```

For restoring from Embedded User Authentication backup files:

The USER and PASSWORD server credentials are required to create a database on the server and execute the restore service. The **-eua_u[ser] name** and **-eua_p[assword] text** database credentials are required to ensure that only the database owner can restore an EUA database.

Table 8.4 Restoring a Database with gbak

Argument	Description
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).
dbfile	The name of a restored database file
size	Length of a backup file or restored database file • The only permissible unit for a restored database file is database pages; minimum value is 200 • Default unit for a backup file is bytes • Size of backup files can also be specified in kilobytes, megabytes, or gigabytes • Do not specify a size for the final backup file or database file; the last file always expands as needed to fill all available space

Table 8.5 lists **gbak** options that are available when restoring databases.

Table 8.5 gbal Restore Options

Option	Description
-c[reate_database]	Restores database to a new file
-bu[ffers]	Sets cache size for restored database
-i[nactive]	Makes indexes inactive upon restore
-eua_u[ser] name	Checks for user name before accessing EUA (embedded user authentication) database. The -user and -password options still need to be provided, in addition to your -eua equivalents when restoring an EUA database. Either EUA switch can be omitted if it has an identical value to its counterpart. However it is recommended that the PASSWORD and EUA_PASSWORD should not be the same.

Table 8.5 gbal Restore Options (*continued*)

Option	Description
-eua_p[assword] text	Checks for password text before accessing EUA (embedded user authentication) database". The -user and -password options still need to be provided, in addition to your -eua equivalents when restoring an EUA database. Either EUA switch can be omitted if it has an identical value to its counterpart. However it is recommended that the PASSWORD and EUA_PASSWORD should not be the same.
-k[ill]	Does not create any shadows that were previously defined
-mo[de] {read_write | read_only}	Specifies whether the restored database is writable • Possible values are **read_only** and **read_write** • Default is **read_write**
-n[o_validity]	Deletes validity constraints from restored metadata; allows restoration of data that would otherwise not meet validity constraints
-o[ne_at_a_time]	Restores one table at a time; useful for partial recovery if database contains corrupt data
-p[age_size] n	Resets page size to *n* bytes (1024, 2048, 4096, 8192, or 16384); default is 4096
-pas[sword] text	Checks for password *text* before accessing a database
-r[eplace_database]	Restores database to new file or replaces existing file
-se[rvice] servicename	• Creates the restored database on the host where the backup files are located, using InterBase's Service Manager • *servicename* invokes the Service Manager on the server host; syntax ies with the network protocol in use: TCP/IP `hostname:service_mgr` NetBEUI `\\hostname\service_mgr` Local `service_mgr`
-user name	Checks for user *name* before accessing database

Table 8.5 gbal Restore Options (*continued*)

Option	Description
-use_[all_space]	Restores database with 100% fill ratio on every data page. By default, space is reserved for each row on a data page to accommodate a back version of an UPDATE or DELETE. Depending on the size of the compressed rows, that could translate e to any percentage.
-v[erbose]	Shows what **gbak** is doing
-va[lidate]	Use to validate the database when restoring it.
-w[rite] {async | sync | direct}	Overrides a database write mode. The “write” mode is preserved during a backup/restore lifecycle. **sync** enables forced writes; **async** enables buffered writes; **direct** enables direct I/O.
-y [file | suppress_output]	If used with **-v**, directs status messages to *file*; if used without **-v** and *file* is omitted, suppresses output messages
-z	Show version of **gbak** and of InterBase engine

When restoring a database, keep the following information in mind:

- Anyone can restore a database. However, only the database owner or SYSDBA can restore a database over an existing database.
- Do not restore a backup over a database that is currently in use; it is likely to corrupt the database.
- When restoring from a multifile backup, name all the backup files, in any order.
- Do not provide a file size for the last (or only) file of the restored database. InterBase does not return an error, but it always “grows” the last file as needed until all available space is used. This dynamic sizing is a feature of InterBase.
- You specify the size of a restored database in database pages. The default size for database files is 200 pages. The default database page size is 4K, so if the page size has not been changed, the default database size is 800K. This is sufficient for only a very small database. To change the size of the database pages, use the **-p[age_size]** option when restoring.

Tip Use the **-service** switch if you are restoring to the same server that holds the backup file. This option invokes the InterBase Service Manager on the server host and saves both time and network traffic.

Note If you specify several target database files but have only a small amount of data, the target files are quite small (around 800K for the first one and 4K for subsequent ones) when they are first created. They grow in sequence to the specified sizes as you populate the database.

Using gbak with InterBase Service Manager

When you run **gbak** with the **-service** switch, **gbak** invokes the backup and restore functions of InterBase's Service Manager on the server where the database resides. When run without the **-service** switch, **gbak** executes on the machine where it is invoked—typically a client—and writes the backup file on (or relative to) that machine. Using the **-service** switch to invoke the Service Manager saves a significant amount of time and network traffic when you want to create the backup on the same host on which the database resides. When the Service API is used, both the database and the backup file must be accessible to the server.

When you use the **-service** switch, you specify the host name followed by the string "service_mgr". The syntax you use for this ies with the network protocol you are using. Together, these components are referred to as "host_service" in the syntax statements that follow in this section.

Table 8.6 Host_service Syntax for Calling the Service Manager with **gbak**

Network protocol	Syntax
TCP/IP	*hostname*:service_mgr
NetBEUI	*hostname*\service_mgr
Local	service_mgr
OTW (Over-the-wire)	hostname/port number?ssl=true?...<other OTW parameters>...??:service_mgr

The syntax in the right column appears in the **gbak** syntax below as "host_service."

The local case is trivial on NT. If you are backing up a local database, the results in terms of time and network traffic are the same whether you use the **-service** switch or not, even though the actual implementation would be slightly different. On UNIX systems, the local case is equivalent to specifying (for TCP/IP) **localhost:service_mgr** and saves both time and network traffic.

Syntax **Backing up with Service Manager**

```
gbak -b [options] -se[rvice] host_service database filename
```

Syntax **Restoring with Service Manager**

```
gbak {-c|-r} [options] -se[rvice] host_service filename
database
```

You can back up to multiple files and restore from multiple files using Service Manager.

Important On UNIX systems, in order to restore a database that has been backed up using the Service Manager, you must either use the Service Manager for the restore or you must be logged onto the system as the user that InterBase was running as when the backup was created (either *root* or *InterBase*). This is because the InterBase user (*root* or *InterBase*) is the owner of the backup file at the system level when the Service Manager is invoked, and the backup file is readable to only that user. When **gbak** is used to back up a database without the **-service** option, the owner of the backup file at the system level is the login of the person who ran **gbak**. On Windows platforms, the system-level constraints do not apply.

The user name and password

When InterBase checks to see whether the user running **gbak** is authorized to do so, it determines the user according to the following hierarchy:

- The -**user** that is specified, with a correct password, as part of the **gbak** command
- The user and password specified in the ISC_USER and ISC_PASSWORD environment variables, provided they also exist in the InterBase security database. (Setting these environment variables is strongly *not* recommended, since it is extremely insecure.)
- UNIX only: If no user is specified at any of the previous levels, InterBase uses the UNIX login if the user is running on the server or on a trusted host.

Some backup and restore examples

Note The following examples use forward slashes exclusively. InterBase accepts either forward or backward slashes for paths on Wintel platforms.

Database backup examples

The following example backs up *foo.ib*, which resides on the server *jupiter* and writes the backup file to the current directory of the client machine where **gbak** is running. *foo.ib* can be either a single-file database or the name of the first file in a multifile database. Using this syntax (without the **-se** switch) copies a lot of data over the net.

```
gbak -b -user joe -password blurf@ jupiter:/foo.ib foo.ibk
```

The next example backs up *foo.ib*, which resides on the server *jupiter* and writes the backup file to the *C:/archive* directory on the client machine where **gbak** is running. As before, *foo.ib* can be a single file database or the name of the first file in a multifile database. This syntax causes the same amount of network traffic as the first example.

```
gbak -b -user joe -password blurf@ jupiter:/foo.ib C:\
archive\foo.ibk
```

The next example backs up the same database on *jupiter*, but uses the **-se[rvice]** switch to invoke the Service Manager on *jupiter*, which writes the backup to the *\backup* directory on *jupiter*. This command causes very little network traffic and is therefore faster than performing the same task without the **-se** (**-service**) switch. Note that the syntax (jupiter:service_mgr) indicates a TCP/IP connection.

```
gbak -b -user joe -password blurf@ -se jupiter:service_mgr /
foo.ib /backup/foo.ibk
```

The next example again backs up *foo1.ib* on server *jupiter* to multiple files in the */backup* directory on *jupiter* using the Service Manager. This syntax backs up a single file or multifile database and uses a minimum of time and network traffic. It converts external files as internal tables and creates a backup in a transportable format that can be restored on any InterBase-supported platform. To back up a multifile database, name only the first file in the backup command. In this example, the first two backup files are limited to 500K. The last one expands as necessary.

```
gbak -b -user joe -pass blurf@ -co -t -se
jupiter:service_mgr
  /foo1.ib/backup/backup1.ibk 500k /backup/backup2.ibk 500k
  /backup/lastBackup.ibk
```

Database restore examples

The first example restores a database that resides in the */archive* directory on the machine where **gbak** is running and restores it to *jupiter*, overwriting an existing (but inactive) database.

```
gbak -r -user joe -pass blurf@ C:\archive\foo.ibk jupiter:/
foo.ib
```

The next example restores a multifile database from the */backup* directory of jupiter to the */companydb* directory of *jupiter*. This command runs on the server by invoking Service Manager, thus saving time and network traffic. In this example, the first two files of the restored database are 500 pages long and the last file grows as needed.

```
gbak -r user -joe -pass blurf@ -se jupiter:service_mgr /
backup/foo1.ibk
  /backup/foo2.ibk /backup/fooLast.ibk /companydb/foo1.ib
500
  /companydb/foo2.ib 500 /companydb/fooLast.ib
```

The next example executes on server Jupiter using Service Manager and restores a backup that is on Jupiter to another server called Pluto.

```
gbak -r user -joe -pass blurf@ -se jupiter:service_mgr
  /backup/foo.ibk pluto:/companydb/foo.ib
```

gbak error messages

Table 8.7 gbak Backup and Restore 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 contains corruption or metadata violates integrity constraints • Try restoring tables using **-one_at_a_time** option, or delete validity constraints using **-no_validity** option
Cannot commit index <string>	• Data might 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
Datatype n not understood	An illegal datatype is being specified

Table 8.7 gbak Backup and Restore Error Messages (*continued*)

Error Message	Causes and Suggested Actions to Take
Database format *n* is too old to restore to	• The **gbak** version used is incompatible with the InterBase version of the database • Try backing up the database using the **-expand** or **-old** options and then restoring it
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 could 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	Try redefining the problem array
Expected array version number *n* but instead found m	Try redefining the problem array

Table 8.7 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 might 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 8.7 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 not assigned to any 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, 8192, or 16384, whichever is closest.
Page size specified (n) greater than limit	Specify a page size of 1024, 2048, 8192, or 16384. The default is 4096 bytes.
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

Table 8.7 gbak Backup and Restore Error Messages (*continued*)

Error Message	Causes and Suggested Actions to Take
Skipped *n* bytes after reading a bad attribute n	
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 might 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 might 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 • Try deleting 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	

Performing backups and restores using IBConsole

This section provides instruction on how to use IBConsole to backup and restore a database. You can use IBConsole to specify both full and incremental backups. For an overview of each type, see the **"About InterBase backup and restore options"** section at the beginning of this chapter.

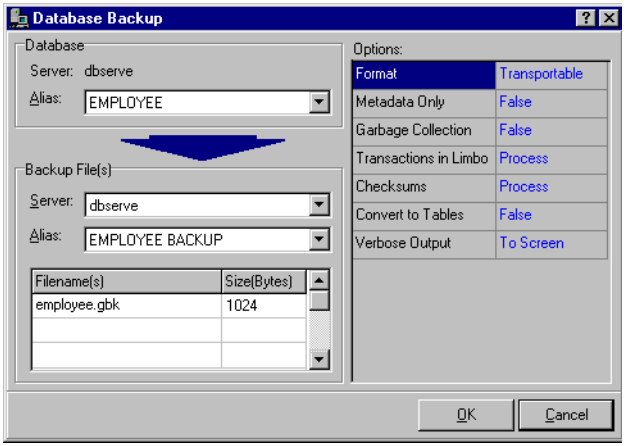
Performing a full backup using IBConsole

To initiate a full, logical backup using IBConsole, take the following steps:

1. Right-click on a database in the tree pane, and select **Backup/Restore** from the context menu.

2 When the context menu expands to display backup and restore options, select **Backup**. The **Database Backup** dialog appears, as shown in Figure 8.1.

Figure 8.1 Database Backup Dialog



3 Check the database server to make sure the server indicated is correct. If it is not, cancel this dialog and re-initiate the Database Backup dialog under the correct server.

4 If you accessed the Database Backup dialog from a database alias, the Alias field is automatically assigned. If you accessed the Database Backup dialog from the Databases menu, then you must select an alias from the list of database aliases.

The database alias references the associated database file name, so you need only specify the alias name, not the actual database filename, when indicating the database to back up. If the database spans multiple files, the server uses the header page of each file to locate additional files, so the entire database can be backed up based on the alias filename.

5 Select a destination server from a list of registered servers in the Backup Files Server drop-down list.

6 Once a destination server has been selected, a list of backup file aliases is available from the Backup Files Alias drop-down list. If you want to overwrite an existing backup file, select the appropriate file from the drop-down list. If you want to create a new backup file, you can type a new alias name in the Backup File(s) Alias field.

7 Indicate where the backup is to be stored by entering one or more filenames, specifying a size for each file, in the Backup File(s) table. To insert a new row into the Backup File(s) table, move to the last row and column in the table and type w-z.

When entering a filename, make sure to include the file path unless you wish to write the file to the current working directory.

If you select an existing backup alias, the table displays all the filenames and file sizes of that alias. You can edit any information within this table. To add another file to the backup file list, enter a new filename at the end of the table. To remove a file from the backup file list, delete the values in the table.

8. You can specify backup options by entering a valid value, by clicking the option value and choosing a new value from a drop-down list of values, or by double-clicking the option value to rotate its value to the next in the list of values. See **"About IBConsole backup options"** below for descriptions of these options.
9. Click OK to start the backup.

Note 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. Because files that have the *gdb* extension automatically get backed up whenever they are touched in some versions of Windows XP, InterBase now recommends using an *ib* extension for database files and *ibk* for backup files.

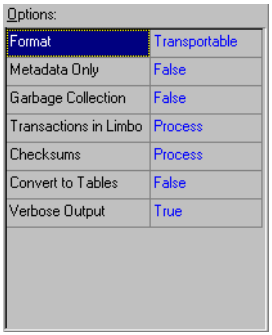
A backup file typically occupies less space than the source database because it includes only the current version of data and incurs less overhead for data storage. A backup file also contains only the index definition, not the index data structures.

If you specify a backup file that already exists, IBConsole overwrites it. To avoid overwriting, specify a unique name for the backup file.

About IBConsole backup options

The backup options are shown on the right side of the Database Backup dialog. You can specify options by entering a value, by clicking the option value and choosing a new value from a drop-down list of values, or by double-clicking the option value to rotate its value to the next in the list of values.

Figure 8.2 Database Backup Options



Format

Option values are Transportable and Non-transportable.

To move a database to a machine with an operating system different from the one under which the backup was performed, make sure the Format option is set to Transportable. This option writes data in a generic format, enabling you to move to any machine that supports InterBase.

Important Never copy a database from one location to another. Back it up and then restore it to the new location.

Metadata Only

Option values are True and False.

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 True for the Metadata Only option.

Tip You can also extract a database's metadata using **isql**. **isql** produces a SQL data definition text file that contains the SQL code used to create it. IBConsole backup Metadata Only creates a binary backup file containing only metadata.

This function corresponds to the **-metadata** option of **gbak.**

Garbage collection

Option values are True and False.

By default, IBConsole performs garbage collection during backup. To prevent garbage collection during a backup, set the Garbage Collection option value to False.

Garbage collection marks space used by old versions of data records as free for reuse. Generally, you want IBConsole to perform garbage collection during backup.

Tip Disabling garbage collection during the backup process improves backup performance. If garbage collection is necessary, you can run a separate sweep operation using gfix. In addition, 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**.

Transactions in limbo

Option values are Process and Ignore.

To ignore limbo transactions during backup, set the Transactions in Limbo option value to Ignore.

When IBConsole 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 Database|Maintenance|Transaction Recovery in the Database Maintenance window. Refer to **"Recovering Transactions"** for more information.

This function corresponds to the **-limbo** option of **gbak**.

Checksums

Note For performance reasons, InterBase supports true checksums only for ODS 8 and earlier. For ODS 9 and later, InterBase always generates the string "12345" as the checksum. This maintains compatibility with older versions.

Option values are Process and Ignore.

To ignore checksums during backup, set the Checksums option value to Ignore.

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**.

Convert to Tables

To convert external files to internal tables, set the Convert to Tables option value to True.

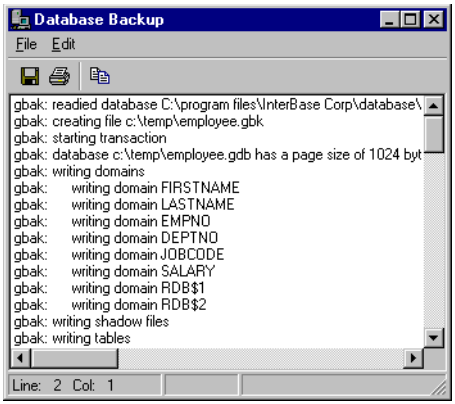
This function corresponds to the **-convert** option of **gbak**.

Verbose Output

Option values are None, To Screen and To File.

To monitor the backup process as it runs, set the Verbose Output option value to To Screen. This option opens a standard text display window to display status messages during the backup. For example:

Figure 8.3 Database Backup Verbose Output



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 **"Standard Text Display Window" .**

This function corresponds to the **-verbose** option of **gbak.**

Transferring databases to servers running different operating systems

1. Set the Format option to Transportable 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 a removable medium, such as a tape. Then load the contents of the medium onto another machine, or copy it across a network to another 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.

Performing an incremental backup using IBConsole

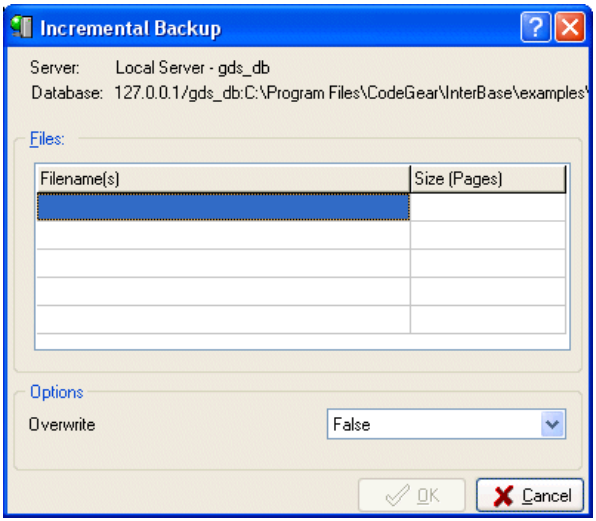
An **incremental backup** copies all of the changes that have been committed to the database since the last full backup. The first time you choose Incremental Backup from IBConsole, InterBase performs a full physical backup (an online dump). After the initial full dump, each subsequent incremental backup saves and copies all of the transactions committed since the last full backup.

To initiate an incremental backup using IBConsole, take the following steps:

1. In the tree pane, right-click the database on which to perform an incremental backup, and select Backup/Restore from the context menu.

2 When the context menu expands to display backup and restore options, select **Incremental Backup**. The Incremental Backup dialog appears, as shown in Figure 8.4.

Figure 8.4 Specifying an Incremental Backup



3 On **Incremental Backup**, specify the following:

- In the **Filename(s)** table, indicate where the backup is to be stored by entering one or more filenames. To insert a new row into the Filename(s) table, move to the last row and column in the table and type w-z. When entering a filename, make sure to include the file path unless you wish to write the file to the current working directory.
- The **Size (Pages)** field is optional and denotes the file's size in units of pages, using the database's page size. If the [size] parameter is not provided then that dump file's size will be determined by its file-sequenced counterpart in the database. If the dump file's sequence is higher than the sequence of any database file then it takes the size of its predecessor dump file.

4 To overwrite the previous incremental backup, change the **Overwrite** value to True. For detailed information about what occurs when you overwrite an incremental backup, see “Over-writing Incremental backups.”

5 Choose **OK** to start the backup.

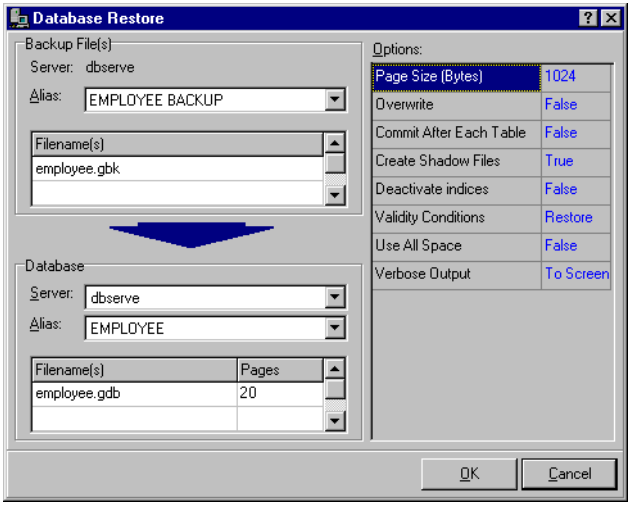
Restoring a database using IBConsole

Use the Database Restore dialog to restore databases. To access this dialog, select a server from the list of available servers displayed in the Tree pane and continue with one of these possible methods:

- Select anything under the databases hierarchy and choose Database|| Maintenance|Backup/Restore|Restore.
- Double-click any backup alias name under the Backup hierarchy.
- Right-click Backup or any backup alias name under the Backup hierarchy and choose Restore from the context menu.
- Select any backup alias name under Backup and click Restore in the Work pane.

The Database Restore dialog appears:

Figure 8.5 Database Restore Dialog



Important When restoring a database, do not replace a database that is currently in use.

To restore a database

1. Check the source Backup File(s) Server to make sure the server indicated is correct. If it is not, cancel this dialog and re-initiate the Database Restore dialog under the correct server.
2. If you accessed the Database Restore dialog from a backup alias, then the Alias field is automatically assigned. If you accessed the Database Restore dialog from Backup, then you must select an alias from the list of backup aliases.

   Note The backup alias references the associated backup file names, so you need only specify the alias name, not the actual backup file name, when indicating the backup to restore. If the backup spans multiple files, the server uses header page of each file to locate additional files, so the entire backup can be restored based on the alias filename.

3. If you choose a backup file alias, the Backup File(s) table displays the associated backup files. If you do not specify a backup file alias, then you can either enter the backup filenames manually, or browse for the file by selecting File from the Alias drop-down list. If you enter the file name manually, include the complete path. It is important that you include all filenames associated with the restore.

   To insert a new row into the Backup File(s) table, move to the last row and column in the table and type w-z.

4. Select a destination server from a list of registered servers in the Database Server drop-down list.

5. If you want to restore to an existing database, select its alias from the Database Alias drop-down list. If you want to restore to a new database, type a new alias name in the Database Alias field.

6. In the Filename(s) / Pages table, enter one or more filenames for the restored database and specify the number of pages required for each file. Include the complete path unless you want to place the files in the current working directory. To insert a new row into the Database table, move to the last row and column in the table and type w-z.

   You might want to restore a database to multiple files to distribute it among different disks, which provides more flexibility in allocating system resources.

   If you selected an existing database alias, the Database table displays all the associated filenames and number of pages. You can edit any information within this table. You can add another file to the database file list by entering a new filename at the end of the table. You can remove a file from the list by deleting the values in the table.

   Note You cannot restore a database to a network file system (mapped drive).

7. You can specify options for the restore by entering a valid value, by clicking the option value and choosing a new value from a drop-down list of values or by double-clicking the option value to rotate its value to the next in the list of values. See **“Restore options”** below for a description of these options.

8. Click OK to start the restore.

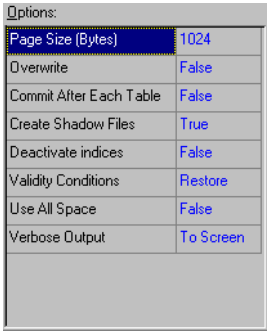
   Typically, a restored database occupies less disk space than it did before being backed up, but disk space requirements could change if the ODS version changes. For information about the ODS, see **“Restoring the ODS”** .

Note The InterBase restore utility 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 necessary to build an index. If this occurs, the database is restored and available, but indexes are inactive. After the restore completes, use ALTER INDEX to make the indexes active.

Restore options

The restore options are shown on the right side of the **Database Restore** dialog. You can specify options by entering a value, by clicking the option value and choosing a new value from a drop-down list of values, or by double-clicking the option value to rotate its value to the next in the list of values.

Figure 8.6 Database Restore Options



Page Size

InterBase supports database page sizes of 1024, 2048, 4096, 8192, and 16384 bytes. The default is 4096 bytes. To change the page size, back up the database and then restore it, modifying the Page Size option 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 4KB, 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 Database|Maintenance|Database Statistics to display index statistics, and consider increasing the page size if index depth is greater than three 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**.

Overwrite

Option values are True and False.

IBConsole cannot overwrite an existing database file unless the Overwrite option value is set to True. If you attempt to restore to an existing database name and this option is set to False, the restore does not proceed.

To restore a database over an existing database, you must be the owner of the existing database or SYSDBA.

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

Option values are True and False.

Normally, IBConsole restores all metadata before restoring any data. If you set the Commit After Each Table option value to True, IBConsole 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. This can happen if the data is corrupt or is 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**.

Create Shadow Files

Shadow files are identical, physical copies of database files in a database. To recreate shadow files that were saved during the backup process set the Create Shadow Files option to True. For further information on shadowing see **"Shadowing"** .

Deactivate Indexes

Option values are True and False.

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, set the Deactivate Indexes option to True. This makes indexes inactive and prevents them from rebuilding. Then eliminate the duplicate index values, and re-activate indexes through ALTER INDEX in **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**.

Validity Conditions

Option values are Restore and Ignore.

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 **isql**.

To restore a database that contains invalid data, set the Validity Conditions option to Ignore. 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**.

Use All Space

Option values are True and False.

To restore a database with 100% fill ratio on every data page, set the Use All Space option to True. By default, space is reserved for each row on a data page to accommodate a back version of an UPDATE or DELETE. Depending on the size of the compressed rows, that could translate to any percentage.

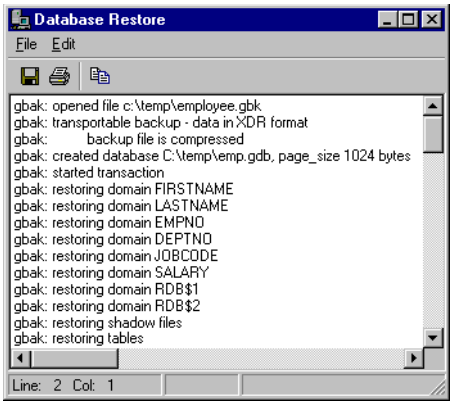
This function corresponds to the **-use_all_space** option of **gbak**.

Verbose Output

Option values are None, To Screen, and To File.

To monitor the restore process as it runs, set the Verbose Output option to To Screen. This option opens a standard text display window to display status messages during the restore. For example:

Figure 8.7 Database Restore Verbose Output



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 **“Standard Text Display Window” .**

This function corresponds to the **-verbose** option of **gbak**.

Chapter

9

Journaling and Disaster Recovery

Journaling combines the protection of forced writes (also known as synchronous writes) with better performance, and also supports the improved disaster recovery provided by journal archiving.

When enabled, journal archiving allows databases to recover from a complete loss of an InterBase server machine to within a few minutes of when the disaster occurred, or to a specific point in time.

This chapter defines key terms such as journal, journal file, and journal archive, and explains how to implement and use them with your database.

Note Journaling is available in the Server Edition of InterBase, starting from version 2007. It is not available in the Desktop Edition.

About Journals, Journal Files, and Journal Archives

A **journal** consists of one or more journal files. A **journal file** records each database transaction as it occurs. This makes a journal an always-up-to-date record of database transactions. **Journaling** is sometimes referred to as **write-ahead logging (WAL)**.

A **journal archive** is a directory that contains a full database dump *and* all the journal files completed since that dump. You can use the journal archive to recover to the last transaction recorded in the most recently archived journal file. You can also use an archived journal file to perform point-in-time recovery. Point-in-time recovery uses the internal timestamp that is recorded on each transaction in the file, which allows you to, if desired, recover to a specific date and time.

How Journaling Works

Journaling turns off forced writes to the hard drive, and synchronously writes the updates to the journal file instead. Because journal file writes are sequential rather than random, there is no disk head movement, which improves database performance.

To save changed pages in the database cache to the hard disk, you set up **journaling checkpoints** to occur automatically. A checkpoint specifies the time at which InterBase must save all the changed pages in the database cache to the database file. After the checkpoint has been reached, the data in the journal file is no longer needed, so the file can be reused. For best performance, place the journal files on a dedicated hard drive. The journal files must be on the database server machine.

Journaling guarantees that all changes are on disk before a transaction is marked committed as long as O/S and hardware caching are disabled.

You do not need to use journal archiving to use journaling and journal files. However, journal archiving lets you recover from a disaster that completely destroys the database server.

When a database is configured for direct I/O, adding journaling does not automatically convert the database to asynchronous buffered I/O as it does when the database is configured for synchronous buffer I/O. This is to avoid buffered I/O at all cost when the database is set to direct I/O.

When a database is configured for direct I/O, adding journaling does not automatically convert the database to asynchronous buffered I/O as it does when the database is configured for synchronous buffer I/O. This is to avoid buffered I/O at all costs when the database is set to direct I/O.

Important InterBase XE Update 2 enables the "direct I/O" functionality on Windows OS. This was implemented to circumvent the issue observed by many on Windows 2008 R2 and Windows 7 64-bit OS editions where System File Cache uses up too much physical memory leading to sluggish system performance.

- InterBase XE3 Update 4 Hotfix 1 (version >= 11.0.4.816) resolves the above issue and restores the original performance for databases in sync/async write mode.

InterBase uses buffered file I/O on all platforms to perform I/O on database pages for the file on disk. The pages are delivered via the System File Cache, which acts as a duplicate store of the pages on RAM. Subsequent loads of the same page(s) are quickly served by the OS kernel if the page exists in the System File Cache. On systems where there is high contention with other files for the System File Cache (a shared pool used by all processes for buffered file I/O) the performance of InterBase may not be optimal. If available System File Cache is limited due to

RAM resource limitations, the kernel must spend time cleaning up unused blocks of memory from other processes as well as provide for servicing a new block I/O request.

The performance problem is alleviated by using "direct I/O" (also known as non-buffered I/O) so blocks of pages are directly read from the disk into the process space and do not need to use the System File Cache.

This is supported on **Windows OS only.** This setting is **not supported on non-Windows** platform databases; you will see the following error.

feature is not supported
-direct I/O operation

If a database enabled with "direct" I/O is then copied to an older version of InterBase, the setting will not be used by the older InterBase server. The older server will employ the "sync" write mode in this case.

The fixes are as follows:

The **gfix** command line tool has been modified to allow setting a database to be in "direct" I/O write mode.

gfix [-write {async, sync, direct}] . . .

For example:
#gfix -write direct foo.ib -user sysdba -password masterkey

The **gbak** command line tool now has a new restore option (optional) setting to override a database write mode. The "write" mode will be preserved during a backup/restore lifecycle.

gbak [-write {async, sync, direct}] . . .
For example:
gbak -write direct -r foo.ibk foo.ib -user sysdba -password masterkey

Services API support for the new and updated *gfix* and *gbak* options.

You can find the various new agruments and respective values in ibase.h

API Guide Table 12.5: Service API database restore arguments

Argument: isc_spb_res_write_mode
Purpose: Set the write mode of the database. The next byte must be one of:
isc_spb_res_wm_async
isc_spb_res_wm_sync
isc_spb_res_wm_direct
Corresponds to gbak -write
Argument Length: 1 byte
Argument Value: byte

API Guide Table 12.6: Service API database properties arguments

Add isc_spb_prp_wm_direct to the following argument:
isc_spb_prp_write_mode

gstat command line tool will exhibit the following setting,direct, in its "Attributes" header line output.

```
# gstat -h foo.ib -user sysdba -password masterkey
. . .

Database header page information:
Flags 0
Checksum 12345
Write timestamp Mar 3, 2011 13:36:31
Page size 8192
ODS version 15.0
. . .
Creation date Feb 23, 2011 14:58:27
Attributes force write, direct, no reserve
Variable header data:
Sweep interval: 20000
*END*
```

. . .

It is important to note that a database needs to be set with "gfix -write direct" option and reloaded by the database engine for this to take effect.

Also, since the System File Cache will not be used when "direct" I/O is set, it is recommended that the database cache setting and database linger interval be set suitably. This allows the most frequently used pages to be in memory, the InterBase database cache, when new connections are serviced.

This "direct" I/O setting on a database is only possible if the database page size is an exact multiple of the file's underlying disk sector size. The standard for so many decades has been 512 *bytes per sector* on hard disks. Newer hard disks however are trying to adopt the more *Advanced Format* of 4096 *bytes per sector*. InterBase supports the following database page sizes: 1024, 2048, 4096, 8192 and 16384 bytes per page. Databases that have a page size of 1024 or 2048 bytes cannot be set to "direct" I/O on hard disks that only support the 4096 *bytes per sector* standard; you need to restore your database to a larger page size on such disks before enabling "direct" I/O on them.

If you try to enable "direct" I/O on an incompatible device, the following error message is returned stating the minimum required database page size. The following example shows an error message where the disk sector size is 4096 bytes.

```
Error: must backup and restore to DB page size

>= 4096 bytes to support direct I/O on this device.
```

How Journal Archiving Works

The purpose of journal archiving is to provide effective and efficient disaster recovery. As mentioned above, a journal archive is a directory that contains a full database dump and all of the completed journal files that have been archived since that dump. As such, a journal archive enables you to recover to the last committed transaction in the most recently archived and completed journal file.

Important For disaster recovery purposes, a journal archive should always be located on a different machine — ideally, in a remote location — than the one that houses the database server.

Only completed journal files are archived to the archive directory. This means that *up to the moment* recovery is possible when the hard drive that contains the current, active, unarchived journal file remains intact. However, if disaster wipes out the hard drive that contains the active, incomplete journal file, the data on that file will also be lost.

Note Before you can activate journal archiving, you must first enable journaling. For instructions on how to do so, see **“Enabling Journaling and Creating Journal Files”** . For instructions on how to activate journal archiving, see **“Using Journal Archiving”** .

Configuring your System to Use Journaling and Journal Archiving

Use the following criteria to determine your system’s optimal journaling configuration:

- The I/O speed of the device on which the journal files are created.
- The speed of concurrent creation of new journal files.
- Hardware requirements and ease of setup.

It is not necessary for InterBase to be installed and running on the machine used for journal archive storage.

Additional Considerations

- A journal archive is platform-specific. For example, an archive created with InterBase for Windows cannot be directly used to recover an InterBase database on another platform. Instead, an archived database dump could be logically backed up in transportable format and then logically restored on the other platform.
- Only full dumps are archived. You cannot archive incremental database dumps. The gbak -archive_database command initiates a full, physical backup. For more information about InterBase backup options, see **“Database Backup and Restore.”**

- The journal and journal archive are restricted to a single directory. The number of items allowed to be archived will be limited by the number of files that are allowed in a directory for a given file system.

Enabling Journaling and Creating Journal Files

To create a journal file and activate journaling, use the following DDL syntax:

Syntax

```
CREATE JOURNAL [<journal-file-specification>] [LENGTH
<number-of-pages> [PAGES]]
  [CHECKPOINT LENGTH <number-of-pages> [PAGES]]
  [CHECKPOINT INTERVAL <number-of-seconds> [SECONDS]]
  [PAGE SIZE <number-of-bytes> [BYTES]]
  [PAGE CACHE <number-of-buffers> [BUFFERS]]
  [[NO] TIMESTAMP NAME]
  [[NO] PREALLOCATE <number-of-pages> [PAGES]];
```

Note InterBase currently requires that all journal files be stored in the same directory.

All CREATE JOURNAL clauses are optional. Table 9.1 describes the function of each option and its default value.

Table 9.1 CREATE JOURNAL Options

Option	Description	Default values
<Journal_file_spec>	Specifies a quoted string containing the full path and base file name of the journal file. The base journal file name is used as a template for the actual journal file names as they are created.	The full database path and file name
LENGTH	This clause specifies the number of pages that can be written to the journal file before rolling over to a new journal file. For tips on calculating this number, see “Tips for Determining Journal Rollover Frequency” .	Maximum: 2GB or 4000 pages
CHECKPOINT LENGTH	This clause specifies the number of pages that can be written to the journal file before checkpoint occurs. For tips on calculating checkpoint length, see “Tips for Determining Journal Rollover Frequency” .	500

Table 9.1 CREATE JOURNAL Options

Option	Description	Default values
CHECKPOINT INTERVAL	Determines the number of seconds between database checkpoints. The checkpoint interval determines how long it will take to recover after a server crash. Note: If both `CHECKPOINT LENGTH` and `CHECKPOINT INTERVAL` are specified, whichever event occurs first will initiate a database checkpoint. For tips on calculating the checkpoint interval, see “Tips for Determining Checkpoint Intervals” .	0
PAGE SIZE	Determines the size of a journal page in bytes. A journal page size must be at least twice the size of a database page size. If a journal page size of less is specified, it will be rounded up to twice the database page size and a warning will be returned. The journal page size need not be a power of 2.	Twice the database page size
PAGE CACHE	Determines the number of journal pages that are cached to memory. This number must be large enough to provide buffers for worker threads to write to when the cache writer is writing other buffers. If the number is too small, the worker threads wait and performance suffers.	100 buffers
[NO] TIMESTAMP NAME	Determines whether or not to append the file creation timestamp to the base journal file name. If used, the base journal file name will be appended with a timestamp in the following format: `YYYY_MM_DDTHH_MM_SSZ.sequence_number.journal`	Enabled
[NO] PREALLOCATE	Specifies the amount of disk space preallocated for journal files. For more information about using the preallocate clause, see “About Preallocating Journal Space” .	No default value.

The `CREATE JOURNAL` statement causes all subsequent write operations on a database to be done asynchronously. The journal file I/O is always synchronous and cannot be altered. All transaction changes are safely recorded on durable storage before the transaction is committed. This guarantees the ACID properties of a transaction (the database industry standards for Atomicity, Consistency, Isolation, and Durability).

Using asynchronous I/O for database writes allows the operating system to optimize file I/O, such as by writing consecutive pages together, or by using scatter/gather techniques that write consecutive pages in discontiguous page buffers. Journal file I/O is performed using InterBase's careful write strategy. This implies that database pages can be written back to the database in any order after their changes have been journaled.

During a database checkpoint, any database page writes that were buffered asynchronously are flushed to disc before checkpoint completion is signaled. You can re-enable synchronous writes for the database, which removes the requirement for a flush operation before a database checkpoint can be considered complete. Doing so, however, can degrade performance.

About Preallocating Journal Space

As suggested earlier, for best performance journal files should be placed on a dedicated hard drive. If they are not, access to other files on the drive causes the disk heads to move away from the journal files, which forces a seek-back to the journal file before a new page is written. In addition, each time the journal file size increases, even when the journal files are on a dedicated drive, the disk heads seek away from the journal file so the operating system can update the directory information. To allow the disk heads to remain on the current journal file at all times, you can use the PREALLOCATE clause. The PREALLOCATE clause enables you to allocate space equal to the maximum number of journal files that will exist during normal operation, multiplied by the length of the journal files specified in the LENGTH clause of the CREATE JOURNAL statement.

If the journal is not on a dedicated drive, you can use the PREALLOCATE clause to allocate space equal to the size of the maximum number of journal files that might exist. This guarantees that other files cannot consume the space that may be needed for the journal. If journal archiving is enabled, and you are archiving to a remote machine, allocate enough space to accommodate the journal files that will accumulate if the connection to the remove machine is lost and the journal files cannot be archived for a period of time.

Tips for Determining Journal Rollover Frequency

Journal file rollover is a time-consuming operation. If rollover happens too frequently, performance will degrade. However, if you are using journal archiving, you want the journal file to rollover as often as possible so the completed journal file can be archived frequently, which minimizes the number of transactions that will be lost if disaster destroys the database server. Determining the most effective rollover frequency is a balancing act and the best answer will be different for each InterBase installation.

You can use the following equations to help you determine the most efficacious rollover frequency for your journal files. You can enter the resulting number in the LENGTH clause of the CREATE JOURNAL statement, which specifies when the end of a

journal file is reached. When the end of the file is reached, journaling resumes on a new file. When a journal file is complete (i.e. its end has been reached), it can be saved to the archive directory.

To determine frequency in bytes, use the following equation:

```
  (journal file length * journal page size) = # of bytes
before rollover occurs
```

To determine a rollover interval, you can use either of the following equations:

```
  (journal file length * journal page size) / (database page
size * writes per minute) = # of minutes between rollovers
```

The equation above lets you see how often a rollover will occur for a given journal file length. The equation below calculates the journal length that will give the rollover interval you specify.

```
  (rollover interval * database page size * writes per
minute) / journal page size = journal file length
```

Tips for Determining Checkpoint Intervals

InterBase uses the checkpoint interval to determine how long it takes InterBase to recover after a server crash. If the server crashes, all of the changes in the cache will be lost. This is okay because the changes were written synchronously to the journal, which stores them on disk, but not in the database. When the server restarts, it must read all of the journal files and copy the changes in the journal to the database to bring the database file up-to-date. The more journal files there are, the longer this will take. Performing frequent checkpoints means that the changes in the cache are written to the database frequently, so fewer changes are kept in the journal. This in turn means that fewer journal files are required and recovery will take less time.

You can use the following equations to help you determine the most effective checkpoint interval for your system:

```
  (checkpoint length * journal page size) / (database page
size * writes/minutes) = # of minutes between checkpoints
```

To help determine the time your system needs to recover, use this equation:

```
  (checkpoint length * journal page size) / 1,048,576 =
maximum time to recover after a crash in seconds
```

Note This equation assumes that the journal file is processed at a rate of one megabyte per second during crash recovery. Typically, a journal file is processed at one to two megabytes per second.

To determine checkpoint length for a given recovery time, use this equation:

```
  (recovery time in seconds * 1,048,576) / journal page size
= checkpoint length
```

Displaying Journal Information

To display journaling information for a database, use the following command:

```
gstat <a_database> -l
```

The switch is a lower case L, rather than the numeral one.

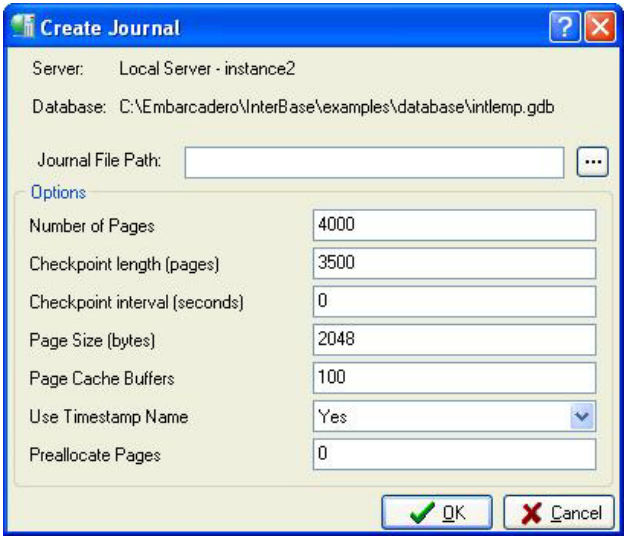
Using IBConsole to Initiate Journaling

IBConsole offers the same journaling options in a dialog box as those described above in Table 9.1. You cannot use IBConsole to create journal archives.

To initiate journaling from IBConsole, take the following steps:

1. In the tree pane, right-click the database for which to initiate journaling, and select **Backup/Restore** from the context menu.
2. When the Backup/Restore menu options appear, select **Create Journal**. The Create Journal dialog appears, as shown in Figure 9.1.

Figure 9.1 The Create Journal Dialog



3. On Create Journal, specify the options to use, then choose **OK** to begin journaling. For descriptions of each option, see Table 9.1.

Disabling Journal Files

The `DROP JOURNAL` statement discontinues the use of write-ahead logging and deletes all journal files. This operation does not delete any journal files in the journal archive but does discontinue maintenance of the journal archive. Dropping journal files requires exclusive access to the database.

Syntax To disable journaling, use the following syntax:

```
DROP JOURNAL
```

Using Journal Archiving

As mentioned above, a **journal archive** is a directory that contains a full database dump, and the completed journal files that have been archived since that dump. InterBase requires that you create and update archive content (the dump and journal files) in two steps:

1. Archive the database
2. Archive the journal files

The commands used to perform each task are explained below.

Note Creating a journal archive does not require exclusive database access.

The command that Activates Journal Archiving

The CREATE JOURNAL ARCHIVE command activates journal archiving in an InterBase database. A journal archive file (IB_JOURNAL_ARCHIVE) is placed in the journal archive directory.

Note To perform an initial (and subsequent) dumps, gbak -archive_database must be performed.

The following command is used to **activate** journaling:

Syntax

```
CREATE JOURNAL ARCHIVE <journal archive directory>
```

where <journal archive directory> is the location in which InterBase stores the journal archive. If the directory does not exist or is not accessible, InterBase returns an error message. The directory path can be a local drive, a mapped drive, or an UNC path (as long as the underlying file APIs can open the file using that specification). If you do not specify a journal archive directory in the CREATE JOURNAL ARCHIVE statement, InterBase uses the journal directory created with the CREATE JOURNAL statement.

When you do not activate journal archiving, the current journal files are reused after a checkpoint writes the journal file's records to the hard drive.

Important You only use the CREATE JOURNAL ARCHIVE command to initiate journal archiving on a database. Once you initiate archiving and InterBase performs the first dump, you use the gbak -archive_database command, discussed below, to perform subsequent dumps. If you disable journal archiving and want to resume it, use CREATE JOURNAL ARCHIVE.

The Command that Archives Journal Files

The `gbak -archive_journals` command instructs InterBase to copy the completed journal files to the journal archive directory. To ensure that the archive always contains the most recently completed journal files, you should issue this command on a regular basis.

To copy completed journal files to the archive directory, use the following syntax:

Syntax

```
gbak -archive_journals <dbname>
```

where <dbname> specifies the database that is being archived. The journal archive will grow in storage size as the most recently completed journal files are continually archived. For instructions on how to manage archive size, see **"Managing Archive Size"** .

The Command that Performs Subsequent Archive Dumps

After the initial dump, performed by the `CREATE JOURNAL ARCHIVE` command, you use the following syntax to perform subsequent dumps:

Syntax

gbak -archive_database <dbname>

This command performs a full, physical dump to the archive directory, which helps to reduce the number of journal files that must be stored. The older a dump is, the more journal files InterBase needs to keep the archive up-to-date.

How Often Should you Archive Journal Files?

Use the following statements and questions to help determine how often to archive journal files:

- How much data can you afford to lose if the IB server is destroyed?
- What is the journal rollover frequency? There is no reason to archive journal files more often that the journal rollover interval.
- Frequent journal rollover + frequent journal archiving means minimum data loss. However, too frequent journal rollover + too frequent journal archiving means poor performance. What is the best balance for your system?

Disabling a Journal Archive

The DROP JOURNAL ARCHIVE statement disables journal archiving for the database. It causes all journal files and database file dumps to be deleted in all journal archive directories. The file system directories themselves are not deleted.

Disabling journal archiving does not disable database journaling (the creation of journal files). The database will continue to use the write-ahead protocol to commit database changes to the journals. If the intent is to also disable journaling, then you must execute a separate DROP JOURNAL statement, shown in **"Disabling Journal Files"** .

To disable journal archiving, use the following syntax:

Syntax `DROP JOURNAL ARCHIVE`

Using a Journal Archive to Recover a Database

To recover a database from a journal archive, use the following syntax:

Syntax
```
gbak -archive_recover [-until <timestamp>]
<archive_dbname> <local_dbname>
```

If you do not use the -UNTIL switch, InterBase recovers the database to the last committed transaction in the most recently archived journal file *or* to the last committed transaction in the current, active journal file if the current, active journal file is accessible. The -until <timestamp> instructs InterBase to recover transactions until the date and time you specify.

It is recommended that you start building a new archive soon after a successful recovery event. You can create a new archive by issuing the gbak -archive_database and the `gbak -archive_journals` commands.

Managing Archive Size

As the number of archived journal files grows, recovery time increases. To keep the archive from becoming too large, you can use any of the following options:

- Run the gbak -archive_database command to create a new dump, thereby reducing the number of journal files InterBase needs to keep the archive up-to-date.
- Run the gfix command to set a maximum number of dumps to allow in the archive:

  ```
  gfix -archive_dumps <number> db_name
  ```

  When the number of database dumps in the archive exceeds the <number> given, older dumps and journals will be deleted.
- Run the `gfix -archive_sweep` command to manually control archive size (described below).

About Archive Sequence Numbers and Archive Sweeping

All archived items are denoted by an archive sequence number that corresponds to the order in which the items were created in the archive.

To remove all files in an archive with a sequence number lower than a number you specify, use the following syntax:

Syntax `gfix -archive_sweep <archive_sequence_no> [-force] db_name`

If an archived item cannot be swept (garbage-collected), the sweep will stop and return an error status. Sometimes all lower sequenced items cannot be deleted. For example, a database dump may depend on a lower sequenced journal file with which to start recovery. In that case, InterBase will automatically adjust the given sequence number to a lower number, so that this dependency is not lost.

Tracking the Archive State

To track the state of the archive, use the RDB$JOURNAL_ARCHIVES system table. The gbak and gfix commands use this system table to decide which archive items are targets for the command.

Table 9.2 describes column and data type information for RDB$JOURNAL_ARCHIVES.

Table 9.2 RDB$JOURNAL_ARCHIVES Table

Column Name	Data Type	Length	Description
RDB$ARCHIVE_NAME	CHAR	1024	The name of the archived item.
RDB$ARCHIVE_TYPE	CHAR	1	The type of the archived item. 'D' indicates a database dump. 'S' indicates a secondary database file of a database dump. 'J' indicates a journal file.
RDB$ARCHIVE_LENGTH	INT64	8	Length of the archived item as stored in bytes.
RDB$ARCHIVE_SEQUENCE	INTEGER	4	Sequence number of archived item.
RDB$ARCHIVE_TIMESTAMP	TIMESTAMP	8	Timestamp when item was stored in the archive.
RDB$DEPENDED_ON_SEQUENCE	INTEGER	4	Sequence of archived item that this item depends on. For 'S' archive types, it would be the sequence no. of the 'D' primary database dump file. For 'D' archive types, it is the sequence no. of the starting journal file for recovering from the archive.
RDB$DEPENDED_ON_TIMESTAMP	TIMESTAMP	8	As above, but the archive timestamp for the archived item that this item depends on.

Journaling Tips and Best Practices

The following example uses the EMPLOYEE sample database that is shipped with InterBase, and is intended as a “best practice” for creating and managing journal files and archives. Its settings are designed to minimize journal file rollover and to reduce the probability of journal buffer wait states. The default property values for the sample journal subsystem are for a low-end machine with minimal configuration. This is very similar to InterBase's default page buffer cache of 2048.

Designing a Minimal Configuration

To begin, set the following parameters:

```
CREATE JOURNAL 'e:\database\test'
LENGTH 65000
CHECKPOINT LENGTH 10000
PAGE CACHE 2500;
```

Given a database that has an 8KB page size, the journal PAGE SIZE will default to 16KB (2 x 8KB). Therefore, the LENGTH parameter (65000) will cause rollover to a new journal file every 1GB (65000 x 16KB). If you instead set the LENGTH at 500, the system would roll over to a new journal file every 8MB, which is extremely frequent. A performance drop may occur during this process. Using a larger LENGTH value will make this occur (65000/500 or 130 times) less often.

The CHECKPOINT LENGTH parameter of 10000 means the database checkpoint will occur every 160MB (10000 x 16KB). Assume the built-in CHECKPOINT LENGTH is 500, which means your system will checkpoint the database every 8MB (500 x 16KB). CHECKPOINT LENGTH is a matter of individual taste. It represents the maximum number of bytes that will have to be applied to a database from the journal files after a system crash. You can expect to average between 1MB to 2MB/sec. applying the journal files during the recovery process. So the 160MB checkpoint length suggested here would take a maximum of about 2 minutes to recover depending on your machine. If your organization can tolerate a longer recovery time in return for minimizing the online frequency of database checkpoints, then raise the CHECKPOINT LENGTH accordingly.

The PAGE CACHE parameter can be raised to reduce the probability of incurring journal buffer wait states. At any moment, the journal cache writer thread will be syncing some number of journal buffers to the journal file on disk. During this period, we want to insure that the worker threads have enough spare journal buffers to write to when a database page's journal changes need to be moved to a journal buffer.

For example, imagine that the journal cache writer is syncing 500 journal buffers to disk. The 2500 journal buffer configuration will leave 2000 spare buffers for the worker threads to dump their journal changes. At the built-in PAGE CACHE default of 100, the worker threads can stall due to a high rate of journal buffer wait states.

Lastly, the use of a SAN mirrored cache will always make InterBase's journaling sub-system result in lower performance than a non-journaled InterBase database. This is because twice the amount of data is being written with the journaling subsystem: once to the journal files and once to the database files, plus the additional CPU cost of journal cache management in the InterBase server.

Even for direct-attached storage, it is necessary to pay attention to on-disk write cache enablement. New computers sometimes arrive with on-disk write cache enabled. This means that synchronous writes to a database or journal are not really synchronized to disk oxide. Unless the write cache (SAN or direct) has been disabled or has battery backup, it can't offer durability for database commits.

InterBase journaling should only result in a performance gain when disk I/O is write-through, where every database write goes to disk oxide and not an on-disk cache.

Hopefully, the CREATE JOURNAL statement above will minimize this cost. Remember that the end goal is to provide point-in-time disaster recovery using the CREATE JOURNAL ARCHIVE statement to archive time-consistent database dumps and journal files.

Creating a Sample Journal Archive

To get started, issue:

```
CREATE JOURNAL ARCHIVE <journal archive directory>
```

This activates journal archiving and performs the initial database dump.

Then copy the completed journal files to the archive directory, using the following syntax:

```
gbak -archive_journals <dbname>
```

Now, in the archive directory listing below, the database dump, EMPLOYEE.2006-08-21T15-48-17Z.1.DATABASE, has no database changes made after 2006-08-21 15:48:17. It doesn't care what updates are going to the main database while it is being dumped or after it is finished dumping. This includes the checkpoint process.

```
24 Aug 21 15:45 IB_JOURNAL
24 Aug 21 15:45 IB_JOURNAL_ARCHIVE
130399832 Aug 21 16:00 EMPLOYEE.2006-08-21T15-45-11Z.1.JOURNAL
979562496 Aug 21 16:00 EMPLOYEE.2006-08-21T15-48-17Z.1.DATABASE
130397262 Aug 21 16:00 EMPLOYEE.2006-08-21T15-51-51Z.2.JOURNAL
130399932 Aug 22 18:13 EMPLOYEE.2006-08-21T15-57-03Z.3.JOURNAL
130398336 Aug 22 18:13 EMPLOYEE.2006-08-22T18-06-19Z.4.JOURNAL
130397418 Aug 22 18:14 EMPLOYEE.2006-08-22T18-10-52Z.5.JOURNAL
35392721 Aug 23 00:27 EMPLOYEE.2006-08-22T18-14-47Z.6.JOURNAL
```

Use the gstat -L EMPLOYEE.2006-08-21T15-48-17Z.1.DATABASE command to generate the following summary:

Database log page information:
Creation date Aug 21, 2006 15:45:11
Log flags: 1
Recovery required
Next log page: 0
Variable log data:
Control Point 1:
File name: E:\EMPLOYEE_JOURNALS_AND_ARCHIVES\ EMPLOYEE.2006-08-21T15-45-11Z.1.JOURNAL
Partition offset: 0 Seqno: 1 Offset: 5694

This is what the main database's log page looked like at precisely 2006-08-21 15:48:17. If you attempt to recover using this database dump, it will start with journal file, EMPLOYEE.2006-08-21T15-45-11Z.1.JOURNAL, at offset 5694 and continue through the last journal file or whatever timestamp was specified with an optional -UNTIL clause:

```
GBAK -ARCHIVE_R E:\EMPLOYEE_JOURNALS_AND_ARCHIVES\ EMPLOYEE.2006-08-21T15-48-17Z.1.DATABASE E:\EMPLOYEE_RECOVER\EMPLOYEE.GDB -UNTIL "2006-08-21 18:08:15"
```

and in the InterBase.LOG:

```
IBSMP (Server) Tue Aug 22 22:49:08 2006
Database: E:\EMPLOYEE_RECOVER\EMPLOYEE.GDB
Long term recovery until "2006-08-21 18:08:15" begin

IBSMP (Server) Tue Aug 22 22:49:09 2006
Database: E:\EMPLOYEE_RECOVER\EMPLOYEE.GDB
Applying journal file: E:\EMPLOYEE_JOURNALS_AND_ARCHIVES\EMPLOYEE.2006-08-21T15-45-11Z.1.JOURNAL

IBSMP (Server) Tue Aug 22 22:51:38 2006
Database: E:\EMPLOYEE_RECOVER\EMPLOYEE.GDB
Applying journal file: E:\EMPLOYEE_JOURNALS_AND_ARCHIVES\EMPLOYEE.2006-08-21T15-51-51Z.2.JOURNAL

IBSMP (Server) Tue Aug 22 22:53:24 2006
Database: E:\EMPLOYEE_RECOVER\EMPLOYEE.GDB
Applying journal file: E:\EMPLOYEE_JOURNALS_AND_ARCHIVES\EMPLOYEE.2006-08-21T15-57-03Z.3.JOURNAL

IBSMP (Server) Tue Aug 22 22:55:44 2006
Database: E:\EMPLOYEE_RECOVER\EMPLOYEE.GDB
Applying journal file:
E:\EMPLOYEE_JOURNALS_AND_ARCHIVES\EMPLOYEE.2006-08-22T18-06-19Z.4.JOURNAL

IBSMP (Server) Tue Aug 22 22:55:57 2006
```

```
Database: E:\EMPLOYEE_RECOVER\EMPLOYEE.GDB
Long term recovery end
```

GBAK -A (creating archive db dump) never locks anything. The only archive management restriction is that archive operations are serialized. You can't do multiple GBAK/GFIX operations against it at the same time. The important point here is that the main database is fully accessible at all times.

GBAK -ARCHIVE_J(OURNALS) <my_database> causes non-archived journal files to be copied to the archive (or marked as archived as above) when you don't want to dump the whole database. Again, a row is entered into RDB$JOURNAL_ARCHIVES for each archived journal file.

GFIX-ARCHIVE_S(WEEP) <sequence no.> <my_database> deletes all files in RDB$JOURNAL_ARCHIVES with RDB$ARCHIVE_SEQUENCE less than the requested sequence.

GFIX -ARCHIVE_DU(MPS) <number> <my_database> configures the maximum number of database dumps allowed in the archive. After issuing GBAK -ARCHIVE_DATABASE, archive management will automatically delete the oldest archive database dump and all earlier journal files if the dump limit has been exceeded by the addition of the new database dump.

GBAK -ARCHIVE_R(ECOVER) <archive_directory/archive_database> <new_database> [-UNTIL <timestamp>], will recover a database from the archived journal files. Remember that <archive_directory> has to be mounted for read access on the machine performing the recovery.

Archive directories can be located on InterBase servers or passive file servers and appliances. The archived files are opened directly by clients and not through an InterBase server. Archive database dumps are sealed so you can simultaneously run database validation (usually requires exclusive), logical GBAK, and have multiple, same-platform machines on the network attach the database for read-only queries, which implies high levels of page I/O over the network.

If the most current, non-archived journal files are accessible from the machine where the recover is being executed, then the recovery process will “jump” to those journal files to recover the most recently committed transactions, notwithstanding the optional -UNTIL clause. The recovered database is divorced of any journal or journal archive so it is necessary to define them again if desired.

However, it is more useful to leave the recovered database in a perpetual state of long term recovery. That is, every time after the first GBAK -ARCHIVE_RECOVER, subsequent GBAK -ARCHIVE_RECOVER statements apply the incremental journal changes. This provides perfect symmetry with the online dump feature:

```
GBAK -DUMP <main_database> <dump_database> -- Full online dump
GBAK -DUMP <main_database> <dump_database> -- Incremental dump
GBAK -DUMP <main_database> <dump_database> -- Incremental dump
...
GFIX -MODE READ_WRITE <dump_database> -- Divorce from main DB

GBAK -ARCHIVE_R <main_database> <recv_database> -- Full recover dump
```

```
GBAK -ARCHIVE_R <main_database> <recv_database> -- Incremental recover
GBAK -ARCHIVE_R <main_database> <recv_database> -- Incremental recover
...
GFIX -MODE READ_WRITE <recv_database> -- Divorce from main DB
```

This functional modification is much more efficient. Full, archival recovery can take hours depending on the volume of journal changes.

If you divorce from the database, you save 1 second in not having to type GFIX -MODE READ_WRITE at the cost of having to create another full recovery if you want a more recent copy (hour(s)). Now you have to run GFIX -MODE READ_WRITE to divorce, but you gain hours of efficiency by being able to get the incremental journal changes since the last GBAK -ARCHIVE_RECOVER. This also means that the recovered database can be deployed more quickly if the main database is lost. It also can function as a more up-to-date READ_ONLY database for queries and reporting purposes.

Lastly, the journal archive is never implicitly dropped as a side-effect of DROP DATABASE or DROP JOURNAL. It is necessary to explicitly issue a DROP JOURNAL ARCHIVE statement before DROP DATABASE. The journal archive potentially represents the last known source of the dropped database's contents so it is intentionally difficult to delete.

Chapter

10

Database Statistics and Connection Monitoring

InterBase provides a number of ways to view statistics about database behavior and to exert control over that behavior. This chapter provides a description of the following InterBase facilities:

- Monitoring with System Temporary Tables
- Viewing Statistics using IBConsole
- The gstat Command-line Tool
- Viewing Lock Statistics
- Retrieving Statistics with isc_database_info()

Monitoring with System Temporary Tables

The InterBase Server has always kept a lot of statistics about what was going on, but it has not been easy, or in some cases possible, to surface that information. Now, InterBase captures that information and makes it available in a set of global system temporary tables. These tables describe the runtime behavior of a database. They also provide a level of control.

Although it has always been possible to see a list of users who were currently attached to a database, you can now find out much more. For example, you can see how long each user has been connected, what application each user is running, or the total amount of data I/O used by each attachment. A glance at the temporary table metadata listed in the ***Language Reference Guide*** will suggest the vast possibilities that are available here.

It is also possible to exercise a certain amount of control over the state of a database by performing updates to these tables. See **“Updating System Temporary Tables”** .

These system temporary tables are specific to each database attachment and are visible only to the sysdba user and the database owner. There is therefore no need for unique names and no danger of collisions by separate attachments. Each table is populated only at the point when a client queries it.

The following system temporary tables are available. Their structure is documented in the ***Language Reference Guide***.

Table 10.1 InterBase Temporary System Tables

Table name	Description
TMP$ATTACHMENTS	One row for each connection to a database
TMP$DATABASE	One row for each database you are attached to
TMP$POOL_BLOCKS	One row for each block of memory in each pool
TMP$POOLS	One row for each current memory pool
TMP$PROCEDURES	One row for each procedure executed since the current connection began
TMP$RELATIONS	One row for each relation referenced since the current connection began
TMP$STATEMENTS	One row for each statement currently executing for any current connection
TMP$TRANSACTIONS	One row for each transaction that is active or in limbo

Querying System Temporary Tables

Clients can query these tables using SELECT statements, just as they would query any other table. By querying these tables, a rich collection of data about server performance and user behavior is available.

You cannot create or redefine temporary tables yourself.

Tip For frequent monitoring, the best transaction control is to start the transaction as READ_COMMITTED, READ_ONLY. Then commit it with COMMIT_RETAINING. This has the least impact on the system.

Refreshing the Temporary Tables

To refresh the rows in the temporary tables, commit your transaction and perform the SELECT from the temporary tables again. InterBase automatically deletes the rows stored in temporary tables on a commit.

Listing the Temporary Tables

To display a list of these temporary tables, issue the following command in **isql**:

SHOW SYSTEM

The temporary tables are listed at the end of the system tables. To see the metadata for a particular table, issue:

SHOW TABLE *tablename*

Note The SHOW SYSTEM command is available only in command-line **isql**, not in InterBase Windows **isql**.

Security

Unlike system tables, which have a default access privilege of SELECT for PUBLIC users, the temporary tables have no default access by PUBLIC. The display and manipulation of this runtime information is restricted to SYSDBA and the database owner. These two users have the option of using the GRANT statement to allow access to other users. The statement can grant only SELECT privileges.

Examples

To illustrate the richness of the possibilities afforded by these temporary tables, here are some examples how you might query them.

Top ten SQL statements by execution

```
SELECT a.tmp$user, s.tmp$timestamp, s.tmp$sql, s.tmp$quantum
  FROM TMP$STATEMENTS s, TMP$ATTACHMENTS a
  WHERE a.TMP$ATTACHMENT_ID = s.TMP$ATTACHMENT_ID
  ORDER BY s.TMP$QUANTUM DESC ROWS 10;
```

Top ten oldest transaction snapshots

```
SELECT a.TMP$USER, t.TMP$TIMESTAMP, t.TMP$TRANSACTION_ID,
t.TMP$SNAPSHOT
  FROM TMP$ATTACHMENTS a, TMP$TRANSACTIONS t
  WHERE a.TMP$ATTACHMENT_ID = t.TMP$ATTACHMENT_ID
  ORDER BY t.TMP$SNAPSHOT ROWS 10;
```

Top ten tables with the most garbage to clean up

```
SELECT TMP$RELATION_NAME, TMP$GARBAGE_COLLECT_PAGES
  FROM TMP$RELATIONS
  ORDER BY TMP$GARBAGE_COLLECT_PAGES DESC ROWS 10;
```

Top ten most executed stored procedures

```
SELECT TMP$PROCEDURE_NAME, TMP$INVOCATIONS
  FROM TMP$PROCEDURES
  ORDER BY TMP$INVOCATIONS DESC ROWS 10;
```

Is database sweep active and what's its progress?

```
SELECT TMP$SWEEP_RELATION, TMP$SWEEP_RECORDS
  FROM TMP$DATABASE
  WHERE TMP$SWEEP_ACTIVE = 'Y';
```

Pool memory allocations grouped by pool type

```
SELECT TMP$TYPE, SUM(TMP$POOL_MEMORY) TMP$TOTAL_MEMORY,
  SUM(TMP$FREE_MEMORY) TMP$TOTAL_FREE
  FROM TMP$POOLS
  GROUP BY TMP$TYPE
  ORDER BY 2 DESC;
```

Updating System Temporary Tables

There are cases where, having acquired information about the state of the database, you need to take appropriate action. You might, for example, detect a transaction that had unexpectedly been open for many hours, or one that was consuming resources that were needed by others. By updating the TMP$STATE column of certain temporary tables, you can perform the following updates:

- Roll back an active or limbo transaction
- Commit a limbo transaction
- Cancel an attachment's executing operation
- Shut down the current attachment
- Make an executing statement stop running

To roll back an active transaction

```
UPDATE TMP$TRANSACTIONS SET TMP$STATE = 'ROLLBACK' WHERE
TMP$TRANSACTION_ID=123;
```

To roll back a limbo transaction

```
UPDATE TMP$TRANSACTIONS SET TMP$STATE = 'ROLLBACK' WHERE
TMP$TRANSACTION_ID=123;
```

To commit a limbo transaction

```
UPDATE TMP$TRANSACTIONS SET TMP$STATE = 'COMMIT' WHERE
TMP$TRANSACTION_ID=123;
```

To cancel the attachment's currently executing operation

```
UPDATE TMP$ATTACHMENTS SET TMP$STATE = 'CANCEL' WHERE
TMP$ATTACHMENT_ID=123;
```

To shut down the current attachment

```
UPDATE TMP$ATTACHMENTS SET TMP$STATE = 'SHUTDOWN' WHERE
TMP$ATTACHMENT_ID=123;
```

Shutting down an attachment detaches the user from the database and terminates the local or network attachment to the server.

To make an executing statement stop running

```
UPDATE TMP$STATEMENTS SET TMP$STATE = 'CANCEL' WHERE
TMP$STATEMENT_ID=123;
```

Making Global Changes

The above examples operate on a single attachment or transaction. You can make more global changes. For example:

To roll back all active transactions

```
UPDATE TMP$TRANSACTIONS SET TMP$STATE = 'ROLLBACK' WHERE
TMP$STATE ='ACTIVE';
```

To roll back all limbo transactions

```
UPDATE TMP$TRANSACTIONS SET TMP$STATE = 'ROLLBACK' WHERE
TMP$STATE ='LIMBO';
```

To commit all limbo transactions

```
UPDATE TMP$TRANSACTIONS SET TMP$STATE = 'COMMIT' WHERE
TMP$STATE ='LIMBO';
```

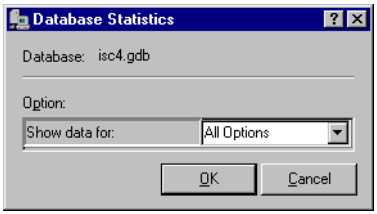
Viewing Statistics using IBConsole

To view database statistics, use one of the following methods to access the Database Statistics dialog:

- Select a connected database in the Tree pane and choose Database|Maintenance|Database Statistics.
- Select a connected database in the Tree pane and double-click Database Statistics in the Work pane.
- Right-click a connected database in the Tree pane and choose Maintenance|Database Statistics from the context menu.

A Database Statistics dialog appears where you can select which statistics you want to display.

Figure 10.1 Database Statistics Options



To view database statistics

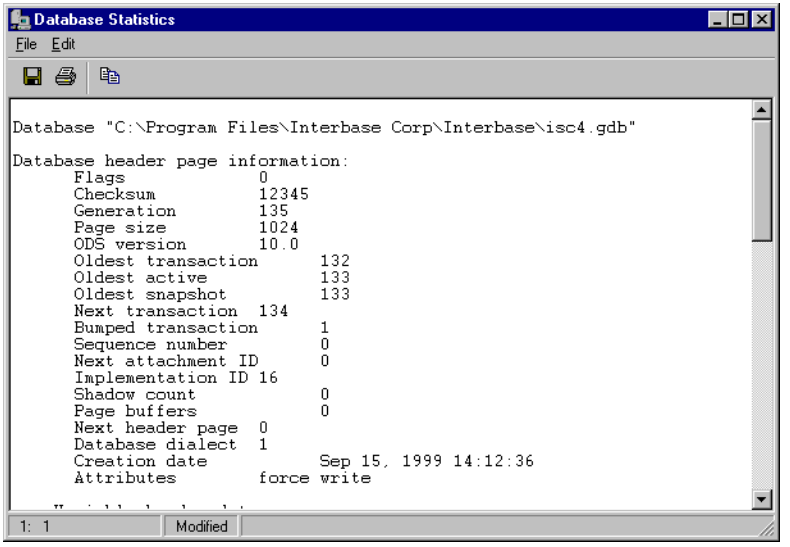
1. Select the statistical data you wish to generate from the Options list.

You can specify options by entering a value, by clicking the option value and choosing a new value from a drop-down list of values or by double-clicking the option value to rotate its value to the next in the list of values.

2 Click OK to generate database statistics.

Note In some cases, it can take a long time to display the statistics for large databases because, depending on what information has been selected to report, generating these statistics may analyze all the tables and indexes in a database.

Figure 10.2 Database Statistics Dialog



The Database Statistics report dialog is a standard text display window that exhibits database summary and database analysis information statistics. For an explanation of how to use the standard text display window, see **“Standard Text Display Window” .**

Database Statistics Options

When you request a statistic option, InterBase generates and displays information for that database statistic. Possible statistic option values include: All Options, Data Pages, Database Log, Header Pages, Index Pages, and System Relations.

Note In addition to the selected statistic, header page information is displayed, regardless which statistic has been selected to report. If Header Pages is the selected option value, then only header page information will be displayed.

All Options

Displays statistic information for all options including Data Pages, Database Log, Header Pages, Index Pages, and System Relations.

This function corresponds to the **-all** option of **gstat**.

Data Pages

Displays data page information in the database summary. Below is an example of data page information, followed by an explanation of each item.

```
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
```

The first line displays a database table name while the remaining lines contain item information pertaining to the table. These items include:

Table 10.2 Data page Information

Item	Description
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.

Database Log

Displays the database log in the database summary. Below is an example of database log information.

This function corresponds to the **-log** option of **gstat**.

```
Database log page information:
  Creation date Dec 20, 1998 11:38:19
  Log flags:2
    No write ahead log
  Next log page:0
  Variable log data:
  Control Point 1:
    File name:
    Partition offset: 0 Seqno: 0 Offset: 0
```

```
  Control Point 2:
    File name:
    Partition offset: 0 Seqno: 0 Offset: 0
  Current File:
    File name:
    Partition offset: 0 Seqno: 0 Offset: 0
```

Header Pages

Displays header page information in the database summary. Below is an example of database summary header page information, followed by an explanation of each item.

This function corresponds to the **-header** option of **gstat**.

```
Database "C:\Embarcadero\InterBase\examples\Database\
employee.ib"

Database header page information:
    Flags           0
    Checksum        12345
    Generation      41
    Page size       4096
    ODS version     12.0
    Oldest transaction29
    Oldest active   30
    Oldest snapshot 30
    Next transaction34
    Bumped transaction1
    Sequence number 0
    Next attachment ID0
    Implementation ID16
    Shadow count    0
    Page buffers    0
    Next header page0
    Database dialect1
    Creation date   Aug 26, 2006 17:05:03

Variable header data:
    Sweep interval: 20000
    *END*

Service ended at 9/3/2006 4:59:05 PM
```

The first line displays the name and location of the primary database file while the remaining lines contain information on the database header page. These items include:

Table 10.3 Header Page Information

Item	Description
Checksum	InterBase supports true checksums only for ODS 8 and earlier.For ODS 9 and later, the checksum value is always “12345”.
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 active, in limbo, or rolled back, but not committed).
Oldest active	The transaction ID number of the oldest active transaction.
Next transaction	The transaction ID number that InterBase assigns to the next transaction. 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”** .
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.

Table 10.3 Header Page Information (*continued*)

Item	Description
Implementation ID	The architecture of the system on which the database was created. These ID definitions are platform-dependent #define directives for a macro class named CLASS: • 1 HP Apollo Domain OS • 2 Sun Solaris SPARC, HP9000 s300, Xenix, Motorola IMP UNIX, UnixWare, NCR UNIX, NeXT, Data General DG-UX Intel • 3 Sun Solaris x86 • 4 VMS • 5 VAX Ultrix • 6 MIPS Ultrix • 7 HP9000 s700/s800 • 8 Novell NetWare • 9 Apple Macintosh 680x0 • 10 IBM AIX POWER series, IBM AIX PowerPC • 11 Data General DG-UX 88K • 12 HP MPE/xl • 13 SGI IRIX • 14 Cray • 15 SF/1 • 16 Microsoft Windows 7 (32-bit and 64-bit) • 17 DARWIN on Intel • 18 DARWIN on PowerPC • 19 DARWIN on iOS ARM architecture • 20 Android on x86 architecture (emulator) • 21 Android on ARM architecture (device)
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.
Database dialect	The SQL dialect of the database
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.
Variable header data	• sweep interval • secondary file information

Index Pages

Displays index information in the database summary. Below is an example of index page information, followed by an explanation of each item.

```
Index CUSTNAMEX (2)
  Depth: 2, leaf buckets: 2, nodes: 27
  Average data length: 45.00, total dup: 0, max dup: 0
  Fill distribution:
       0 - 19% = 0
      20 - 39% = 0
      40 - 59% = 1
      60 - 79% = 0
    80 - 99% = 1
```

Table 10.4 Index Pages Information

Item	Description
Index	The name of the index.
Depth	The number of levels in the index page tree. 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.
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.

System Relations

Displays information for system tables in the database.

```
RDB$CHECK_CONSTRAINTS (24)
    Primary pointer page: 54, Index root page: 55
    Data pages: 5, data page slots: 5, average fill: 59%
    Fill distribution:
     0 - 19% = 0
    20 - 39% = 1
    40 - 59% = 0
    60 - 79% = 4
    80 - 99% = 0
```

```
Index RDB$INDEX_14 (0)
Depth: 1, leaf buckets: 1, nodes: 68
Average data length: 0.00, total dup: 14, max dup: 1
Fill distribution:
     0 - 19% = 0
    20 - 39% = 0
    40 - 59% = 1
    60 - 79% = 0
  80 - 99% = 0
```

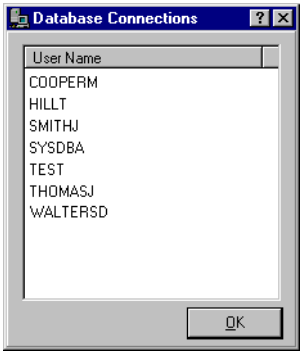
The statistics contained here are similar to that of data pages and index pages. For information on the items see **"Data Pages"** and **"Index Pages"** above.

Monitoring Client Connections with IBConsole

You can view a list of users currently connected to a particular database in IBConsole using the Database Connections dialog. You can access this dialog by one of the following methods:

- Select a database (or any branch under the database hierarchy) in the Tree pane and choose Database|Connected Users.
- Select a database in the Tree pane and double-click **Connected Users** in the Actions column of the Work pane.
- Right-click a database in the Tree pane and choose **Connected Users** from the context menu.

Figure 10.3 Database Connections Dialog



Note InterBase's temporary system tables provide resources for more extensive monitoring of database activity. See **"Monitoring with System Temporary Tables"**.

The 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 IBConsole earlier in this chapter.

You must be SYSDBA or the owner of a database to run **gstat**. On UNIX platforms, there is a further constraint on **gstat**: in order to run **gstat**, you must have system-level read access to the database files. You can gain this by logging in as the same account that the InterBase server is running as (InterBase or root) or by setting the system-level permissions on the database file to include read permission for your Group. These restrictions exist on UNIX platforms because **gstat** accesses the database file at the system level rather than through the InterBase server.

Note You can run **gstat** only against local databases: run **gstat** on the server host.

Options Table 10.5 lists the valid options for gstat.

Table 10.5 gstat Options

Option	Description
-all	Equivalent to supplying **-index** and **-data**; this is the default if you supply none of **-index**, **-data**, or **-all**
-data	Retrieves and displays statistics on data tables in the database
-header	Stops reporting statistics after reporting the information on the header page
-index	Retrieves and displays statistics on indexes in the database
-log	Stops reporting statistics after reporting the information on the log pages
-pa[ssword] ***text***	Checks for password *text* before accessing a database
-r[ecord]	Adds lines for average record length and average version length to the table statistics
-system	Retrieves statistics on system tables and indexes in addition to user tables and indexes
-t[able]	Outputs index and fill information for the requested table, in addition to database header, file, and log statistics; table name is case sensitive
-user ***name***	Checks for user *name* before accessing database
-z	Prints product version of **gstat**

Example The following command requests table statistics, including record and version length for the JOB table in *employee.ib*:

gstat -user SYSDBA -pa masterkey employee.ib -t JOB -r

The command produces the following output:

```
Database "employee.ib"

Database header page information:
        Flags                   0
        Checksum                12345
        Write timestamp         Jul 9, 2010 19:58:59
        Generation              26
        Page size               4096
        ODS version             15.0
        Oldest transaction      19
        Oldest active           20
        Oldest snapshot         20
        Next transaction        21
        Sequence number         0
        Next attachment ID      0
        Implementation ID       16
        Shadow count            0
        Page buffers            0
        Next header page        0
        Database dialect        1
        Creation date           Jul 9, 2010 19:58:59
        Attributes              force write

    variable header data:
        Sweep interval:         20000
        *END*


Database file sequence:
File employee.ib is the only file

Database log page information:
        Creation date
        Log flags:      2
                No write ahead log

        Next log page:  0

    variable log data:
        Control Point 1:
                File name:
                Partition offset: 0     Seqno: 0
Offset: 0
        Control Point 2:
                File name:
```

```
                Partition offset: 0     Seqno: 0
Offset: 0
        Current File:
                File name:
                Partition offset: 0     Seqno: 0
Offset: 0
        *END*

Analyzing database pages ...

JOB (129)
    Primary pointer page: 178, Index root page: 179
    Average record length: 64.87, total records: 31, max
record length: 77
    Average version length: 0.00, total versions: 0, max
versions: 0
    Data pages: 3, data page slots: 3, average fill: 72%
    Fill distribution:
         0 - 19% = 0
        20 - 39% = 1
        40 - 59% = 0
        60 - 79% = 0
        80 - 99% = 2

    Blob pointer page: 253
    Average blob length: 535.27, total blobs: 11, max blob
length: 4598
    Average segment length: 33.83, total segments: 175, max
segment       length:85
    Blob pages: 1, blob page slots: 1, average fill: 41%
    Fill distribution:
       0 - 19% = 0
      20 - 39% = 0
      40 - 59% = 1
      60 - 79% = 0
      80 - 99% = 0

    Index MAXSALX (2)
        Depth: 1, leaf buckets: 1, nodes: 31
        Average data length: 4.00, total dup: 5, max dup: 1
        Fill distribution:
             0 - 19% = 1
            20 - 39% = 0
            40 - 59% = 0
            60 - 79% = 0
            80 - 99% = 0

    Index MINSALX (1)
        Depth: 1, leaf buckets: 1, nodes: 31
```

```
            Average data length: 4.00, total dup: 7, max dup: 2
            Fill distribution:
                 0 - 19% = 1
                20 - 39% = 0
                40 - 59% = 0
                60 - 79% = 0
                80 - 99% = 0

        Index RDB$FOREIGN3 (3)
            Depth: 1, leaf buckets: 1, nodes: 31
            Average data length: 1.00, total dup: 24, max dup:
20
            Fill distribution:
                 0 - 19% = 1
                20 - 39% = 0
                40 - 59% = 0
                60 - 79% = 0
                80 - 99% = 0

        Index RDB$PRIMARY2 (0)
            Depth: 1, leaf buckets: 1, nodes: 31
            Average data length: 10.00, total dup: 0, max dup: 0
            Fill distribution:
                 0 - 19% = 1
                20 - 39% = 0
                40 - 59% = 0
                60 - 79% = 0
                80 - 99% = 0
```

Viewing Lock Statistics

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 such as whether the lock has been granted or is waiting. This information is useful when trying to correct deadlock situations.

Syntax iblockpr [a,o,w] (Windows) or gds_lock_print [a,o,w] (UNIX)

iblockpr [-i{a,o,w}] [*t n*]

Description **iblockpr** monitors performance by checking lock requests.

The first form of syntax given above retrieves a report of lock statistics at one instant in time. The second form monitors performance by collecting samples at fixed intervals.

The options display interactive information on current activity in the lock table. The utility prints out the events per second for each sampling and gives the average of the values in each column at the end.

Table 10.6 iblockpr/gds_lock_print 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
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** separately
-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
t	Specifies the time in seconds between samplings
n	Specifies the number of samples to be taken

Example The following statement prints "acquire" statistics (access to lock table: acquire/s, acqwait/s, %acqwait, acqrtry/s, and rtrysuc/s) every three seconds until ten samples have been taken:

```
gds_lock_print -ia 3 10
```

Retrieving Statistics with isc_database_info()

InterBase 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:

Table 10.7 Database I/O Statistics Information 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

Table 10.7 Database I/O Statistics Information Items (*continued*)

Request Buffer Item	Result Buffer Contents
isc_info_read_idx_count	Number of reads done via an index • Reported per table • Calculated since the current database attachment started
isc_info_read_seq_count	Number of sequential database reads, that is, 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

See the ***API Guide*** for information on request buffers, and details of using this API call.

Chapter

11

Interactive Query

This chapter documents the IBConsole interactive SQL (isql) and command-line **isql** utilities for InterBase. These tools provide an interface to InterBase's Dynamic SQL interpreter. You can use these query tools to perform data definition, prototype queries before implementing them in your application, or to perform ad hoc examination of data in your database.

Topics covered in this chapter include:

- The IBConsole isql Window
- Executing SQL Statements
- Committing and Rolling Back Transactions
- Saving isql Input and Output
- Changing isql Settings
- Extracting Metadata
- Command-line isql Tool
- Using batch updates to submit multiple statements

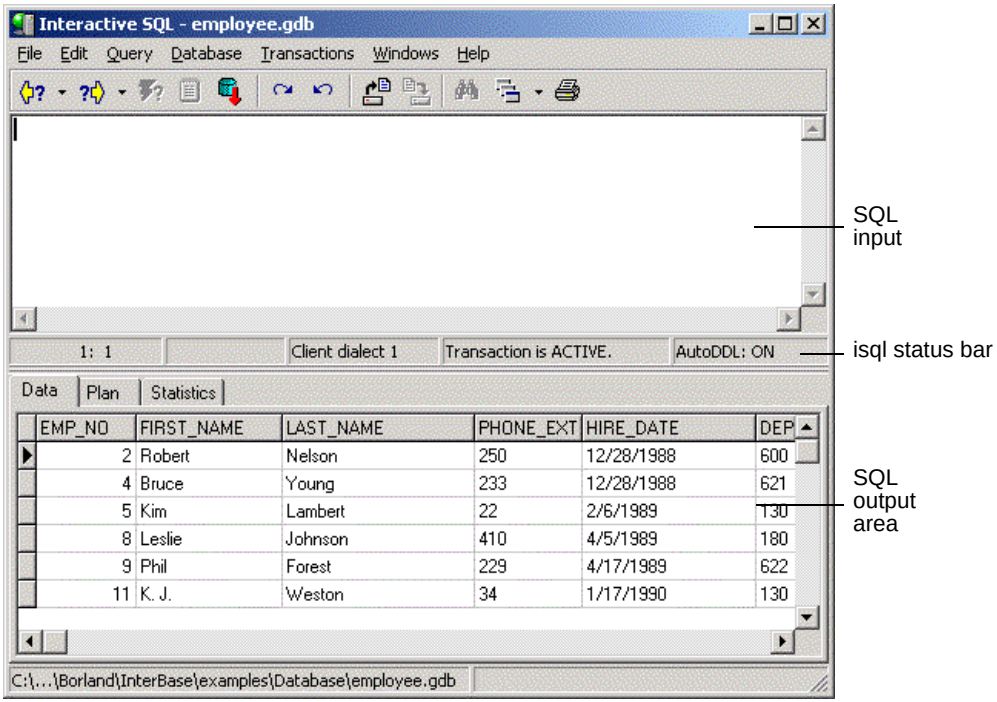
The IBConsole isql Window

The IBConsole isql Window permits you to execute DDL and DML commands to the InterBase server as well as to load, save, print, cut, paste, and copy SQL scripts and results.

To display the isql window:

- Click the Interactive SQL toolbar button .
- Choose Tools|Interactive SQL.

Figure 11.1 The Interactive SQL Editor in IBConsole



SQL Input Area

The SQL input area is where you can type SQL statements or scripts to be executed. It scrolls vertically.

SQL Output Area

The SQL output area is where the results of the SQL statements or scripts are displayed. It scrolls both horizontally and vertically. The SQL output area contains two tabs:

- The Data tab displays any data returned by the SQL output in a grid format.
- The Plan tab displays the plan for the SQL statement or script.
- The Statistics tab displays statistics for the SQL output, including: execution time, prepare time, starting memory, delta memory, current memory, number of buffers, number of reads, number of writes, the plan, and the number of records fetched.

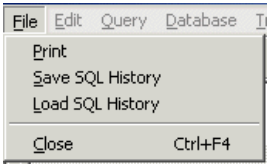
Status Bar

The status bar at the bottom of the SQL input area displays information relevant to the SQL input areas such as cursor position, input status, client dialect, and transaction status. You can change the client dialect by right clicking on the status bar.

isql Menus

The IBConsole Interactive SQL window—also called the SQL Editor—are the File, Edit, Query, Database, Transactions, and Windows menus.

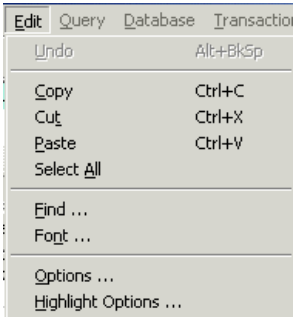
File Menu



The File menu contains the following entries:

- **Print** prints the current contents of the SQL input area.
- **Save** saves the stack of SQL commands from the current session to a file.
- **Load** loads a saved SQL history file.
- **Close** closes the SQL Editor.

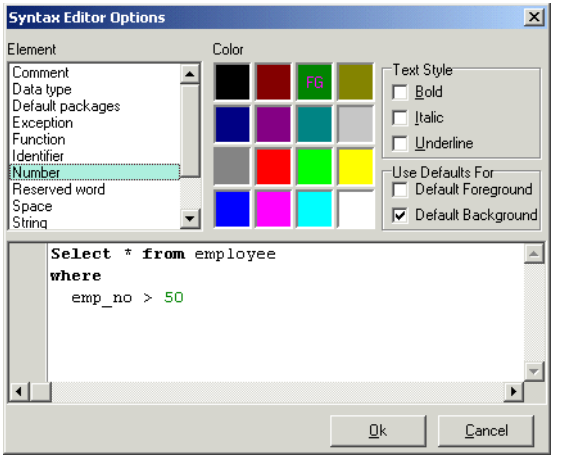
Edit Menu



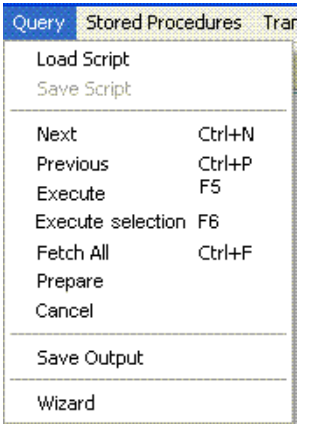
The Edit menu contains the following entries:

- **Undo**, **Cut**, and **Paste** are for use in the SQL input area only. “Undo” in the Edit menu does not undo database changes. Use Transactions|Rollback to undo database changes.
- **Copy**, **Select All**, and **Find** are for use in both the input and output area.

- **Font** specifies the font to be used in the input area
- **Options** are described in **Table 11.2**. They include settings for query plan, auto commit, Blobs, and terminators.
- **Highlight Options** allows you to set the appearance of text in the input area



Query Menu

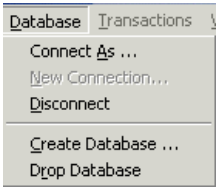


The Query menu contains the following entries:

- **Load Script** loads a SQL script file
- **Save Script** saves a SQL script file
- **Next** steps you forward to the next SQL statement after you have used the Previous command to display earlier statements in the stack.

- **Previous** displays the previous statement that you entered in the input area. This command can be used repeatedly to step through the stack of statements entered during the current interactive SQL session.
- **Execute** executes the SQL statements currently displayed in the input area.
- **Execute Selection** executes the selected portion of the SQL statement displayed.
- **Fetch All** forces the IBConsole to fetch all the rows of a subquery rather than a subset. Rarely needed, because IBConsole fetches the results as you scroll down in the output area.
- **Prepare** displays the optimization plan in the SQL output area.
- **Cancel** exits the optimization plan.
- **Save Output** saves the contents of the output area to a text file.
- **Wizard** opens the Query Wizard where you can create query scripts.

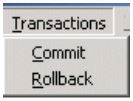
Database Menu



The Database menu contains the following entries:

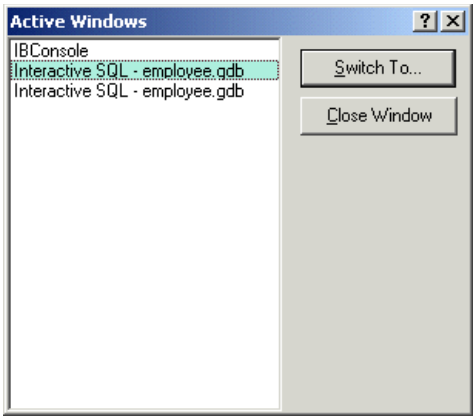
- **Connect As** allows you to connect to the database as a different user.
- **New Connection** is active only when you are not connected to a database. It displays the Database Connect dialog.
- **Create Database** and **Drop Database** are exactly the same as the equivalent commands in the main IBConsole **Database** menu. They are provided in the SQL Editor **Database** menu for convenience.

Transactions Menu



The Transactions menu allows you to commit or roll back transactions.

Windows Menu



Clicking the Windows menu displays a list of current open windows. You can display a window by highlighting it in the list and clicking Switch to or by double-clicking the entry.

isql Toolbar

The IBConsole SQL Editor toolbar contains the following buttons:

Table 11.1 Toolbar Buttons for Executing SQL Statements

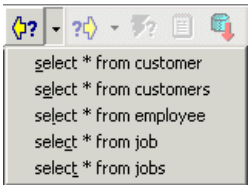
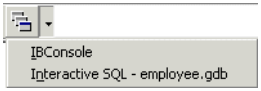
Button	Description
	Displays the previous statement that you entered in the input area. This button can be used repeatedly to step through the stack of statements entered during the current interactive SQL session. The accelerator key is w-P. Click the down arrow next to the button to see a list of available statements. Click a statement in the list to display it in the input area:  (Same as Query|Previous.)
	Steps you forward to the next SQL statement after you have used the Previous command to display earlier statements in the stack. The accelerator key is w-N. Click the down arrow next to the button to see a list of available statements. Click a statement in the list to display it in the input area. (Same as Query|Next.)
	Executes the current statement or script in the SQL input area. The output is displayed in the SQL output area. The accelerator key is w-E. (Same as Query|Execute.)
	Displays the query plan for the current query in the SQL output area. (Same as Query|Prepare.)
	Forces IBConsole to fetch all the rows of a subquery rather than a subset. Rarely needed, because IBConsole fetches the results as you scroll down in the output area. (Same as Query|Fetch All.)
	Commits the transaction specified by the SQL statement to the database. (Same as Transactions|Commit.)
	Rolls back all database changes since the last commit. (Same as Transactions|Commit.)
	Loads a script for SQL execution into the SQL input area. (Same as Query|Load Script.)
	Saves SQL statements entered in the SQL input area to a file. (Same as Query|Save Script.)

Table 11.1 Toolbar Buttons for Executing SQL Statements

Button	Description
	Finds text in the SQL input area. (Same as Edit|Find.)
	Displays a list of current open windows. You can display a window by highlighting it in the list and clicking Switch To or by double-clicking the entry. You can also change windows by clicking the down arrow and clicking the window you want from the drop-down list: 
	Prints the contents of the SQL input area.

Managing isql Temporary Files

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, they are stored in the *Windows* directory.

To avoid cluttering the *Windows* directory with InterBase temporary files, specify a different directory for them by defining the TMP environment variable.

When you exit, isql deletes these temporary files. If isql terminates abnormally, then these files remain and may be freely deleted without any adverse effects. You should not delete any of these temporary files while isql is running, because they may be used in the current session.

Executing SQL Statements

Within isql, you can execute SQL statements in either of two ways:

- Interactively, one statement at a time
- From a script containing multiple statements

Executing SQL Interactively

To execute a SQL statement interactively:

1. Type a single SQL statement in the SQL input area. Make sure any other existing statements are *commented*. A statement is commented if it is preceded by “/*” and followed by “*/”.

   If the statement already exists in the SQL input area make sure all statements except the one you wish to execute are commented. Commented statements in the SQL input area are ignored during execution.

2. Choose Query|Execute, enter w+E, or click the Execute toolbar button.

   If more than one statement is uncommented, Execute executes each statement, one after the other.

Tip You can copy text from other Windows applications such as the Notepad and Wordpad text editors and paste it into the SQL input area. You can also copy statements from the isql output area and paste them into the SQL input area. This cut-and-paste method is also a convenient way to use the online SQL tutorial provided in the online Help.

When SQL statements are executed, whether successfully or not, they become part of the isql *command history*, a sequential list of SQL statements entered in the current session.

Preparing SQL Statements

Use the Prepare toolbar button, or select Query|Prepare, to prepare SQL statements for execution and to view the query plan. Prepare compiles the query plan on the server, and displays it in the Plan tab of the SQL output area. Use Prepare to determine if your SQL script is well-constructed, without having to wait for the SQL script to execute.

Valid SQL 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 in the *Language Reference* as **isql** statements; all these have functionally equivalent menu items in isql.

  For example, the SET NAMES statement cannot be executed from the SQL input area. To change the active character set, choose Edit|Options and select the desired character set option value in the SQL Options dialog.

- SQL script files can include statements that are not valid to enter interactively. For example, you can use the SET statements such as SET LIST in scripts.

- Transaction names may not be used with SET TRANSACTION statement.

- The SQL input area accepts multiple statements, although only one can be executed at a time. Each statement entered in the SQL input area must be terminated by a semicolon (;). The SQL input area accepts multiple statements, although only one can be executed at a time. An uncommented statement that holds the mouse cursor is called the *current statement*.

Executing a SQL Script File

To execute a SQL script file containing SQL statements:

1. Choose Query|Load Script or click the Load Script toolbar button.
2. Locate the desired script file in the Open dialog, and click Open to display the statements of the script file in the SQL input area.
3. Ensure that you are connected to the desired database.
4. If you are connected to the database, comment out any CONNECT or CREATE DATABASE statements.
5. Choose Query|Execute or click Execute on the toolbar to begin executing the entire script statement by statement.

Note Statements executed from a loaded script file do not become part of the command history.

Using Batch Updates to Submit Multiple Statements

Batch updates allow you to send a group of SQL statements to a server in a single unit. Grouping SQL statements into batches reduces the amount of network traffic between the client and the database server. This results in improved performance, especially in LAN and WAN environments.

Note Batch updates only work using the InterBase 2007 client library and an InterClient JDBC driver.

You can send multiple INSERT, UPDATE, and DELETE statements to the server using batch updates. In response, the server returns an array of ULONG values that reflect the number of affected rows per statement.

SQL statements such as SELECT and CREATE DATABASE are not supported in batch updates. SQL DDL is supported.

Figure 11.2 shows the flow of communication between client and server when completing a number of INSERT statements using traditional InterBase client APIs. Note the flow of communication shown in the figure also applies to UPDATE and DELETE statements.

Figure 11.2INSERT Without Batch Updates

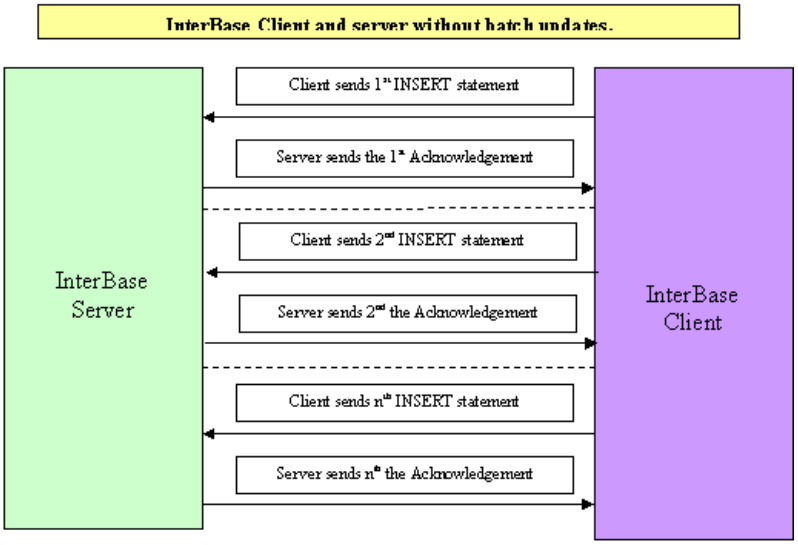
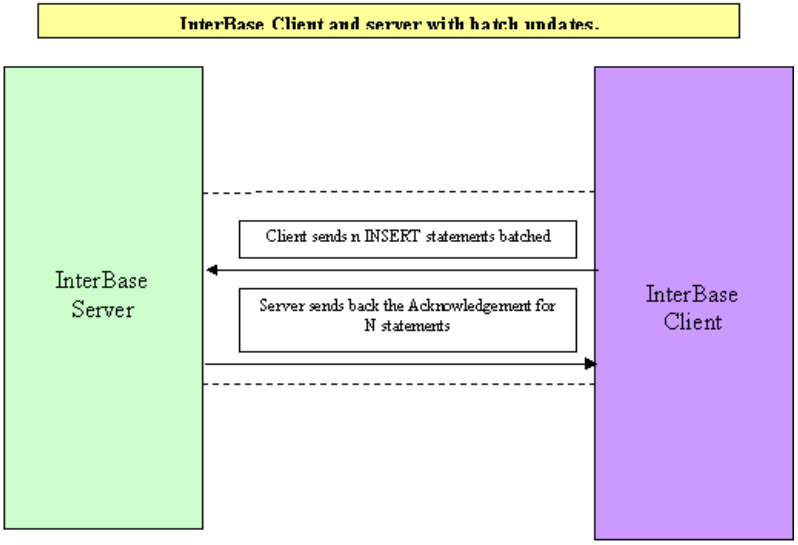


Figure 11.3 shows the flow of communication when using batch updates. Note the reduction in network traffic, resulting in better performance.

Figure 11.3INSERT With Batch Updates



Using the Batch Functions in isql

In isql, SQL statements to be executed in batch mode must be surrounded by the new BATCH START and BATCH EXEXCUTE commands. For example:

```
BATCH START;
...
(allowed DDL/DML statements)
...
BATCH EXECUTE;
```

The BATCH EXECUTE command sends the statements between BATCH START and BATCH EXECUTE to the server. To begin another batch operation, you must issue another BATCH START command.

The following demonstrates a specific example of using batch mode with isql.

```
BATCH START;

INSERT INTO t1(f1, f2) VALUES (0,1);
UPDATE t1 SET f1=1 WHERE f2=1;

BATCH EXECUTE;
```

The first SQL statement in the example inserts a new row into table t1. The second statement updates the newly inserted row with a new value. Both of these statements are executed in one API call.

For details on how to use the batch_excute and batch_execute_immed functions, see Chapter 15 of the InterBase ***API Guide***.

Note The AUTOCOMMITDDL mode of isql must be turned off in order to use batch updates.

Committing and Rolling Back Transactions

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 Edit|Options and set the Auto Commit DDL option to false in the SQL Options dialog.

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 Transactions|Commit or by clicking Commit on the toolbar.

To undo all database changes from DML statements since the last commit, choose Transactions|Rollback or click Rollback on the toolbar.

Saving isql Input and Output

You can save the following to a file:

- SQL statements entered in the SQL input area of the current session.
- The output of the last SQL statement executed.

Saving SQL Input

To save the SQL statements entered in the SQL input area of the current session to a text file:

1. In the SQL Editor, choose Query|Save Script or click the Save Script toolbar button.
2. Enter a file name, including the location for the new file, in the Save As dialog and click Save.

   To include the location for the file, type the file path and file name in the Filename text area, or browse to the folder where you would like the file to reside and type only the file name.

Only the SQL statements entered in the current session, not the output, are saved to the specified file.

Saving SQL Output

To save the results of the last executed SQL statement to a file:

1. In the SQL Editor, choose Query|Save Output.
2. Enter a file name, including the location for the new file, in the Export To dialog and click Save.

   To include the location for the file, either type the file path and file name in the Filename text area, or browse to the folder where you would like the file to reside and type only the file name.

The output in the Data tab from the last successful statement is saved to the named text file.

If you run a SQL script, and then choose to save the output, 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.

Changing isql Settings

Use the SQL Options dialog to display and modify isql session settings, determine if the main IBConsole window will be updated based on the statements given in the isql window, and specify how open transactions are handled when the isql window is closed.

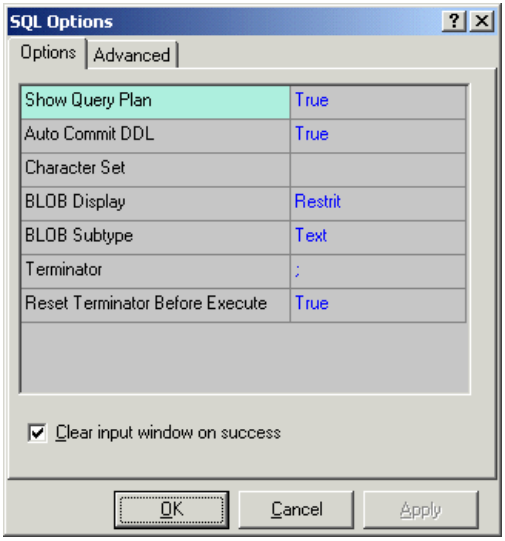
Select Edit|Options from the Interactive SQL window to display the SQL Options dialog.

The SQL Options dialog has two tabs: Options and Advanced.

Options Tab

Use the Options tab to display and modify the isql session settings. You can specify options by clicking the option value and choosing a new value from a drop-down list of values or by double-clicking the option value to rotate its value to the next in the list of values.

Figure 11.4 Options Tab of the SQL Options Dialog



The following table summarizes the **isql** session settings available on the Options tab.

Table 11.2 Options Tab of the SQL Options Dialog

Setting	Description
Show Query Plan	Values: true (default) or false If this setting is True, IBConsole displays the query plan chosen by the optimizer when a SELECT statement is entered. To modify the optimizer plan, use the PLAN option of the SQL SELECT statement. See **"SET PLAN"**.
Auto Commit DDL	Values: true (default) or false • If this setting is True, IBConsole automatically commits DDL (data definition) statements as each statement is entered. • If this setting is False, you must explicitly commit DDL statements (with Transactions | Commit) to make them permanent. See **"SET AUTODDL"**.
Character Set	Determines the active character set for strings for subsequent connections to the database; 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*. • Your 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 a 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"** for more information.
BLOB Display	• Values: Enabled (default), Disabled, Restrict Determines how IBConsole displays columns 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. • If this setting is set to Enabled, IBConsole displays the contents of Blob columns. • If this setting is set to Disabled, IBConsole does not display the contents of Blob columns. If this setting is set to Restrict, IBConsole displays the contents of only Blob columns of the specified BLOB Subtype.
Terminator	Identifies the end-of-statement symbol to be used for SQL queries

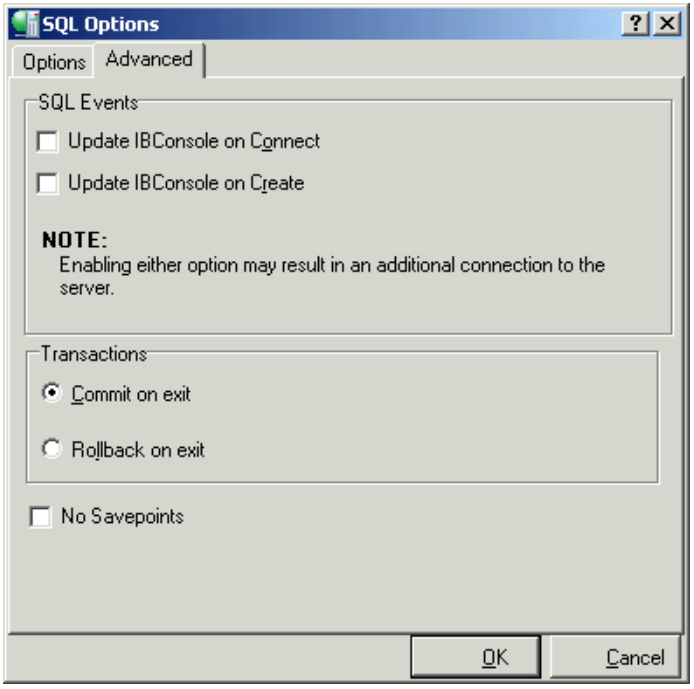
Table 11.2 Options Tab of the SQL Options Dialog (*continued*)

Setting	Description
Clear input window on success	Check this box to clear the SQL input window after a SQL statement is successfully executed

Advanced Tab

Use the Advanced tab to determine whether the main IBConsole window will be updated based on the statements given in the isql window, and to specify how open transactions are handled when the isql window is closed.

Figure 11.5 Advanced Tab of the SQL Options Dialog



The following table summarizes the settings available on the Advanced tab:

Table 11.3 Advanced Tab of the SQL Options Dialog

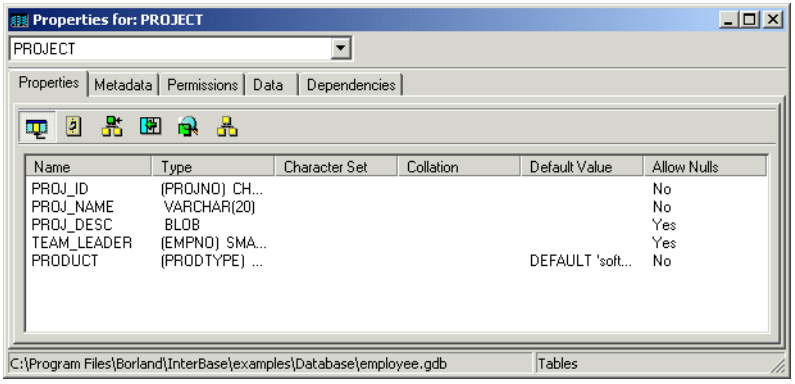
Setting	Description
Update IBConsole on Connect	Does not update the connected databases in the IBConsole window; however, it ensures that the IBConsole window is updated to reflect statements executed in the isql window.
Update IBConsole on Create	Updates the main window if the currently selected server is active. Automatically registers any database created in the isql window and creates an alias for it.
Commit on exit	Commits any active transactions when the isql window is closed.
Rollback on exit	Rolls back any active transactions when the isql window is closed.
No Savepoints	When this option is checked, transactions will not have Savepoints. This option is only for IB7.5 and higher.

Inspecting Database Objects

Use the object inspector to view properties, metadata, permissions, data, and dependencies for the entire database or for a specific table, view, function, procedure, or any other database attribute displayed in the Tree pane.

To open the object inspector, double-click a database object in the Work pane. The object inspector appears:

Figure 11.6 Object Inspector



Depending on the database object selected, the object inspector has some or all of the following tabs: Properties, Metadata, Permissions, Data, and Dependencies. These are discussed in the following sections.

Viewing Object Properties

The Properties tab is available when viewing Table and View database objects. Use the Properties tab of the object inspector to display properties for database objects, including columns, triggers, check constraints, indexes, unique constraints, and referential constraints. The Properties tab has five toolbar buttons for displaying the various object properties:

Table 11.4 Object Inspector Toolbar Buttons

Button	Description
	Show columns: displays the name, type, collation, character set, default value, and whether or not null values are acceptable for every row in the column. The accelerator key is w+y+C. For more information on columns, refer to “Defining columns” in the ***Data Definition Guide***.
	Show triggers: displays the name and type of each trigger, as well as whether or not it is active. In addition, it displays the SQL trigger statement. The accelerator key is w+y+T. For more information on triggers, refer to “Working with Triggers” in the ***Data Definition Guide***.
	Show check constraints: displays the names of the constraints, whether or not they can be deferred, and if they were initially deferred. In addition, it displays the SQL check constraint statements. The accelerator key is w+y+H. For more information, refer to “Defining a CHECK constraint” in the ***Data Definition Guide***.
	Show indexes: displays the name of the index keys, and whether or not they are unique, descending, or active. The accelerator key is w+y+R. For more information, refer to “Working with Indexes” in the ***Data Definition Guide***.
	Show unique constraints: displays the names of the constraints, whether or not they can be deferred, if they were initially deferred, and the index keys. The accelerator key is w+y+U.
	Show referential constraints: displays the names of the constraints, whether or not they can be deferred, if they were initially deferred, the update rule, the update rules, the delete rules, the index, and the reference table. The accelerator key is w+y+R.

Viewing Metadata

The metadata which the Metadata tab of the object inspector displays depends on the database that is selected in the Tree pane, or the item that is selected in the Work pane.

To view metadata for an entire database Select a connected database in the Tree pane, and then double-click View Metadata in the Work pane. The metadata is displayed in a text window.

To view metadata for a specific database object perform one of the following actions:

- Select a database element from the hierarchy displayed in the Tree pane, and then in the Work pane double-click an object to display its Properties dialog. Click the Metadata tab to see the object's metadata.
- Select a database element from the hierarchy displayed in the Tree pane, and then in the Work pane right-click a database object associated with that element and select Extract from the context menu.

For example, if you want metadata for domains only, expand the desired database hierarchy (if it is not already expanded), select Domains, double-click on a domain in the Work pane, and select the Metadata tab of the Properties dialog.

Use the drop-down list at the top of the dialog to select other objects associated with the database element.

The following table lists the items for which you can view metadata for associated objects with the object inspector.

Table 11.5 Metadata Information Items

Item	Displays
Blob Filters	Blob filters definition
Domains	Metadata script, dependencies, datatype, description, check constraints, and default values
Exceptions	Description, exception number, exception message, metadata script, and dependencies
External Functions	UDFs definition
Generators	Generator ID, current value, metadata script, and dependencies
Stored Procedures	Metadata script, procedure body, input parameters, output parameters, permissions, data, and dependencies
Roles	Role definition
Tables	Columns, datatypes, triggers, indexes, unique constraints, referential constraints, check constraints, metadata script, permissions, data, and dependencies
Views	Metadata script, permissions, data, and dependencies

Extracting Metadata

You can extract a metadata script to a file by displaying the desired metadata in the Metadata tab and clicking the Save Script toolbar button.

Extracting an entire database exports metadata in a specific order, to allow the resulting script to be used as input to recreate the database.

Table 11.6 Metadata Extraction Constraints

Metadata	Comments
1. Database	Extracts database with default character set and PAGE_SIZE
2. Domains	Must be before tables that reference domains
3. Tables	Must be after domains
4. Indexes	Must be after tables
5. FOREIGN KEY constraints	Must be added after tables to avoid tables being referenced before they have been 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

Items that are not extracted include:

- 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 Metadata

You can extract a metadata script to a file by displaying the desired metadata in the Metadata tab and clicking the Save Script toolbar button.

Extracting an entire database exports metadata in a specific order, to allow the resulting script to be used as input to recreate the database.

Table 11.7 Order of Metadata Extraction

Metadata	Comments
1. Database	Extracts database with default character set and PAGE_SIZE
2. Domains	Must be before tables that reference domains
3. Tables	Must be after domains
4. Indexes	Must be after tables
5. FOREIGN KEY constraints	Must be added after tables to avoid tables being referenced before they have been 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

Items that are not extracted include:

- 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.

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"**.

You can use **isql** in the following ways:

- Interactively to process SQL statements, by entering statements at the **isql** prompt
- Noninteractively to process SQL statements in a file

Invoking isql

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 noninteractively, depending on the options. For example, reading an input file and writing to an output file are noninteractive 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.

Table 11.8 isql Command-line Options

Option	Description
-a	Extracts all DDL for the named database
-c[ache]	Set number of cache buffers for this connection to the database; see **"Default Cache Size Per isql Connection"** .
-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
-e[cho]	Displays (echoes) each statement before executing it
-ex[tract]	Same as **-x**
-i[nput] ***file***	Reads commands from an input file such as a SQL script 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
-m[erge_stderr]	• Merges stderr output with stdout • Useful for capturing output and errors to a single file when running **isql** in a shell script or batch file
-n[oauto]	Turns off automatic commitment of DDL statements; by default, DDL statements are committed automatically in a separate transaction

Table 11.8 isql Command-line Options (*continued*)

Option	Description
-nowarnings	Displays warning messages if and only if an error occurs (be default, **isql** displays any message returned in a status vector, even if no error occurred)
-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
-q[uiet]	
-r[ole] ***rolename***	Grants privileges of role *rolename* to *user* on connection to the database
-s[qldialect] ***n***	Interprets subsequent commands as dialect *n* until end of session or until dialect is changed by a SET SQL DIALECT statement • For *n* = 1, commands are processed as in InterBase 5 or earlier • For *n* = 2, elements that have different interpretations in dialect 1 and 3 are all flagged with warnings or errors to assist in migrating databases to dialect 3 • For *n* = 3, all statements are parsed as current InterBaseSQL semantics: double quotes are delimited identifiers, DATE datatype is SQL DATE, and exact numerics with precision greater than 9 are stored as INT64
-t[erminator] ***x***	Changes the end-of-statement symbol from the default semicolon (**;**) to *x*, where *x* is a single character or any sequence of characters; deprecated in InterBase 7
-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**

Using Warnings

Warnings can be issued for the following conditions:

- SQL statements with no effect

- SQL expressions that produce different results in InterBase 5 versus InterBase 6 or later
- API calls that will be replaced in future versions of the product
- Pending database shutdown

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.ib'
  ```

- 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 either local or remote databases. The syntax is slightly different for the two:

To connect to a local database on a Windows platform, use the CONNECT command with the full path of the database as the argument. For example:

```
SQL> CONNECT 'C:/Embarcadero/InterBase/Database/examples/employee.ib' role
'staff';
```

To connect to a remote database on a 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.

Examples of connecting to remote databases
To connect to a database on a UNIX platform named *jupiter*:

```
SQL> CONNECT 'jupiter:/usr/InterBase/examples/employee.ib';
```

To connect to a database on a Windows platform named venus:

```
SQL> CONNECT 'venus:c:/Embarcadero/InterBase/examples/database/employee.ib';
```

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 can 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.

Setting isql Client Dialect

To use **isql** to create a database in a particular dialect, first set **isql** to the desired dialect and then create the database. You can set **isql** dialect the following ways:

- On the command line, start **isql** with option **-sql_dialect** *n*, where *n* is 1, 2, or 3:

  isql -sql_dialect *n*

- Within an **isql** session or in a SQL script, include the following statement:

  SET SQL DIALECT *n*;

isql dialect precedence is as follows:

- Lowest: Dialect of an attached version 6 or later database
- Next lowest: Dialect specified on the command line
- Next highest: Dialect specified during the session
- Highest: Dialect of an attached Version 5 database (=1)

In InterBase, **isql** has the following behavior with respect to dialects:

- If you start **isql** and attach to a database without specifying a dialect, **isql** takes on the dialect of the database.
- If you specify a dialect on the command line when you invoke **isql**, it retains that dialect after connection unless explicitly changed.
- When you change the dialect during a session using SET SQL DIALECT *n*, **isql** continues to operate in that dialect until it is explicitly changed.
- When you create a database using **isql**, the database is created with the dialect of the **isql** client; for example, if **isql** has been set to dialect 1, when you create a database, it is a dialect 1 database.
- If you create a database without first specifying a dialect for the **isql** client or attaching to a database, **isql** creates the database in dialect 3.

The statements above are true whether you are running **isql** as a command-line utility or accessing it through IBConsole.

Important Any InterBase **isql** client that attaches to a Version 5 database resets to dialect 1.

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.

Table 11.9 isql Extracting Metadata Arguments

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

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 Metadata"**.For example, the following statement extracts the system catalogs from the database *employee.ib* to a file called *employee.sql*:

```
isql -extract -output employee.sql employee.ib;
```

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.ib* and store in the file *employee.sql*, enter:

```
isql -a employee.ib -output employee.sql
```

The following example extracts the DDL statements from the database *dev.ib*:

```
isql -x dev.ib
```

This example combines the **-extract** and **-output** options to extract the DDL statements from the database *dev.ib* into a file called *dev.out*. The output database name must follow the **-output** flag.

```
isql -extract -output dev.out dev.ib
```

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”**

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 a 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:

```
SQL> QUIT;
```

To exit the **isql** utility and commit all work, enter:

```
SQL> 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:

Table 11.10 SQLCODE and Message Summary

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

For a detailed discussion of error handling, see the ***Embedded SQL Guide***. For a complete listing of SQLCODE and InterBase status array codes, see the *Language Reference* Guide.

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* Guide.

Command-line **isql** supports the following special commands:

Table 11.11 isql 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

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 (**:**) • First number is the ID of the table containing the BLOB column • 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 platforms, EDIT calls the text editor specified by the EDITOR environment variable. If this environment variable is not defined, then EDIT uses the Microsoft Notepad 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 noninteractive 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**

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:

Table 11.12 SET Statements

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 11.12 SET Statements (*continued*)

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 (deprecated in InterBase 7 and later)	;
SET TIME	Toggles display of time in DATE values	ON

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 IBConsole 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 IBConsole isql or **isql**. These statements are not exclusively **isql** statements, so they are not documented in this chapter. See the ***Language Reference* Guide** for details.
- SET DATABASE is exclusively an embedded SQL statement. See the ***Language Reference* Guide** and the ***Embedded SQL 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 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. No table is created. 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.

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 for Blob data of a text subtype [default] • Use other integer values for other subtypes
ALL	Displays Blob data of all subtypes
OFF	Turns 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 can 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 IBConsole 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 is 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.ib';
...
```

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 that 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_COUNTRYMIN_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_COUNTRYMIN_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 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.

This **isql** statement has no equivalent function in IBConsole isql.

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 JOB table. The SHOW TABLES command is used first to display a list of available tables.

```
SHOW TABLES;
  COUNTRY    CUSTOMER
  DEPARTMENTEMPLOYEE
  EMPLOYEE_PROJECTJOB
  PHONE_LISTPROJECT
  PROJ_DEPT_BUDGETSALARY_HISTORY
  SALES

SHOW CHECK JOB;
  CHECK (min_salary < max_salary)
```

See also **SHOW TABLES**

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.ib';
  Database: employee.ib

SHOW DB;
  Database: employee.ib
      Owner: SYSDBA
  PAGE_SIZE 4096
  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 that 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. To display information about a specific function in the current database, use SHOW FUNCTION *function_name*. 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 UNIX 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 JOB table, 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.ib*, 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 NameDependency      Type
  ================ ================== =======
  ADD_EMP_PROJ  EMPLOYEE_PROJECTTable
                UNKNOWN_EMP_ID  Exception
  DELETE_EMPLOYEE DEPARTMENT    Table
                EMPLOYEE        Table
                EMPLOYEE_PROJECTTable
                PROJECT         Table
                REASSIGN_SALES  Exception
                SALARY_HISTORY  Table
                SALES           Table
  DEPT_BUDGET   DEPARTMENT      Table
                DEPT_BUDGET     Procedure
...
```

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
DOONETHINGDOSOMETHING
```

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$RELATION_CONSTRAINTSRDB$RELATION_FIELDS
  RDB$ROLES          RDB$SECURITY_CLASSES
  RDB$TRANSACTIONS   RDB$TRIGGERS
  RDB$TRIGGER_MESSAGESRDB$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) CHAR(15) NOT NULL
  CURRENCY CHAR(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:

```
SQL> SHOW VERSION;
```

```
isql Version: WI-V7.0.0.168
InterBase/x86/Windows NT (access method), version "WI-
V10.0.0.247"
on disk structure version 11.0
```

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 are:

1. Create the script file using a text editor.
2. Run the file with **isql** or IBConsole.
3. View output and confirm database changes.

Creating an isql Script

You can use any text editor to create a 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) that specifies the database on which the script file is to operate. The CONNECT or CREATE statement must contain a complete database file name and directory path.

Note You cannot set dialect in a CREATE DATABASE statement. To create a dialect 3 database, specify **isql** option -r 3.

A 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 end with a terminator.

Note 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*.

Running a SQL Script

The following steps execute all the SQL statements in the specified script file. The contents of the script are not displayed in the SQL input area.

To Run a SQL Script Using IBConsole

1. If you are not already in the SQL window, click the Launch SQL toolbar button or choose Tools|Interactive SQL.
2. If you are not running the SQL script on the database to which you are currently connected, then check that the file begins with a valid, uncommented, CONNECT or CREATE DATABASE statement.
3. Choose Query|Load Script.
4. Enter or locate the desired script filename in the Open dialog, and click Open to load the script into the SQL input area.
5. Click the Execute toolbar button, or choose Query|Execute.

If IBConsole encounters an error, an information dialog appears indicating the error. Once IBConsole finishes executing the script, the script results are displayed in the SQL output window.

After a script executes, all **isql** session settings prior to executing the script are restored as well as the previous database connection, if any. In other words, any **isql** SET commands in the script affect only the **isql** session while the script is running.

To Run a SQL Script Using the Command-line isql Tool

You can run a script from any console prompt using the -input option to isql. Specify the full path and filename. In the following example, the script does not contain a CREATE DATABASE statement; it runs against an existing database:

isql *database_name* -input *filename*

The following example runs a script that creates a database:

isql -input *filename*

During an active **isql** session in which you are already connected to a database, you use the INPUT command to read and execute a SQL script against that database:

SQL> INPUT *filename*

See **"Invoking isql"** for more about running **isql**.

Committing Work in a 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, or set it in the Query Options dialog. See **"Using InterBase Manager to Start and Stop InterBase"** for more information.

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 a SQL statement or **isql** command and can be of any length, as long as it is preceded by "/*" and followed by "*/".

Chapter

12

Database and Server Performance

This chapter describes techniques for designing and operating an InterBase client/server system for best speed and efficiency.

The guidelines in this chapter are organized into the following categories:

- Hardware configuration
- Operating system configuration
- Network configuration
- Database properties
- Database design principles
- Database tuning tasks
- Application design techniques
- Application development tools

Introduction

One of the most important requirements for a database as part of your application is to store and retrieve data as quickly as possible. Like any software development technique, there is always more than one method to implement a given specified software solution, and it takes knowledge and experience to choose the design that results in the most efficient operation and the highest performance.

Each project offers unique challenges and requires specific solutions. The suggestions in this chapter augment your own software engineering discipline, which should include careful analysis, testing, and experimentation to implement the best design for your specific project.

Hardware Configuration

This section gives guidelines for platform hardware sizing. The suggestions focus on requirements for a server platform.

Choosing a Processor Speed

The performance of database systems tends by nature to be bound by I/O bandwidth or network bandwidth. An application often waits for I/O or network operations, instead of being computationally intensive. A fast CPU clock speed gives definite performance advantage, but a 10% increase in CPU clock speed is less important for server performance than some other hardware factors, such as RAM configuration, I/O system, or network hardware.

CPU clock speed is often more important on client platforms, because applications that use data might perform CPU-intensive computational analysis on data, or might render sophisticated visualization of data in a computationally costly manner.

It's not appropriate for this document to recommend a specific CPU clock speed for your server, because it is likely that such a recommendation would be obsolete as you read it. You should evaluate the benefit of spending more money on a faster CPU, because the price/performance curve becomes steep for the latest CPU hardware.

Sizing Memory

It is important to equip your server with a sufficient amount of physical memory to ensure good performance.

While InterBase can function in a low-profile hardware configuration, with as little as 32MB of RAM on most operating systems, it is recommended to have at least 64MB of RAM on a server system. Database servers that experience a high load can benefit from more RAM.

The base RAM requirement of the **ibserver** executable and for each connected user is low: approximately 1500KB, plus 28KB for each client connection. **ibserver** caches metadata and data for each database to which it connects. User operations such as sorting temporarily consume additional memory. A heavily loaded server with dozens of clients performing concurrent queries requires up to 256MB of RAM.

On Windows, you can use the Task Manager, Performance Monitor, and other tools to monitor the resource use of **ibserver**. UNIX and Linux servers have similar resource consumption reporting tools. Add RAM to a system that shows too many page faults.

Using High-performance I/O Subsystems

A multiuser database server's hard drives are no place to be thrifty, especially in today's market of inexpensive storage. Configuring a relatively high-end I/O system is a cost-effective way to increase performance.

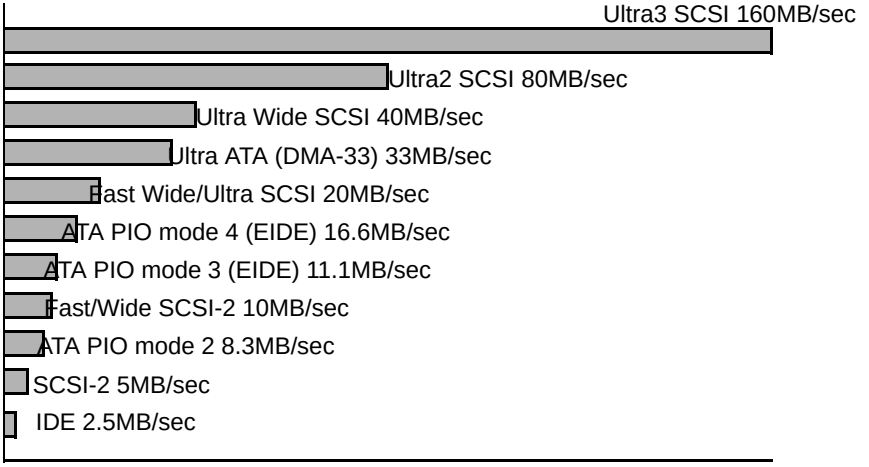
Slow disk subsystems are often the weak link in an otherwise high-performance server machine. The top-rated CPU and maximum memory helps. But if a cheap disk I/O interface limits the data transfer rate, then the money spent on the expensive components is wasted.

It's not appropriate for this document to recommend a particular configuration. The technology changes so quickly that any recommendation here would be outdated. When you specify the machine for a server platform, research the best hardware solution available.

Read the following guidelines for principles:

- Advanced SCSI technology offers superior I/O throughput. The following graph illustrates the relative maximum throughput of different disk interfaces.

Figure 12.1 Comparing external transfer rate of disk I/O interfaces



- The external interface capacity usually exceeds the internal or sustained transfer rate of any individual device. Only systems that use multiple disk devices make full use of a high-capacity I/O interface.

- Bus-mastering I/O controllers use less CPU resources. This is particularly important on I/O-intensive server machines. SCSI is generally bus-mastering, and newer PCI EIDE interfaces are bus-mastering. IDE is not.
- Use a disk controller with built in cache memory. The controller cache reduces the need for the operating system to use system RAM for disk cache.
- Don't assume all disks of a given size perform equally; research performance ratings made by independent testing labs.

Distributing I/O

Disk device I/O is orders of magnitude slower than physical memory accesses or CPU cycles. There is a delay while the disk device seeks the data requested. While an application is waiting for data it has requested from a disk device, it is advantageous for the application to spend the time executing other tasks. One appropriate way to do this is to spread multiple data requests over multiple devices. While one disk is preparing to return data, the application requests another disk to start seeking another set of data. This is called *distributed I/O* or *parallel I/O*.

This section describes ways you can persuade InterBase to distribute I/O over multiple disk devices.

Using RAID

You can achieve up to a ten times performance improvement by using RAID.

RAID (redundant array of inexpensive disks) is a hardware design that is intended to give benefits to performance and reliability by storing data on multiple physical disk devices. It is transparent for software applications to use RAID, because it is implemented in the operating system or at the hardware level. InterBase uses operating system I/O interfaces, so InterBase supports RAID as would any other application software.

Disk striping (included in RAID levels 0, 3, or 5) provides performance benefits by distributing I/O across multiple disks.

Hardware RAID is faster than software RAID or software disk mirroring. RAID implemented with software provides only protection from hard disk failure; it is actually slower than operating without RAID.

Using Multiple Disks for Database Files

Similarly to RAID, you can distribute files of a multifile InterBase database among multiple physical disk drives.

For example, if you have a server with four physical disks, *C:*, *D:*, *E:*, and *F:*, and a 10GB database, you can create your database to take advantage of parallel I/O with the following database creation statement:

```
CREATE DATABASE 'C:\data\bigdata1.ib' PAGE_SIZE 4096
```

```
FILE 'D:\data\bigdata2.ib' STARTING AT PAGE 1000000
FILE 'E:\data\bigdata3.ib' STARTING AT PAGE 2000000
FILE 'F:\data\bigdata4.ib' STARTING AT PAGE 3000000;
```

Using Multiple Disk Controllers

If you have so much disk activity on multiple disks that you saturate the I/O bus, you should equip the server with multiple disk controllers, and connect the multiple drivers to the controllers as evenly as possible.

For example, if you have sixteen disk devices hosting database files, you might benefit from using four disk controllers, and attaching four disks to each controller.

Making Drives Specialized

A database server makes heavy use of both the operating system's virtual memory page file and of temporary disk space. If possible, equip the server with multiple disks and configure the virtual memory file, temporary directory, and database files on separate physical disk devices. This can use parallel I/O to the fullest advantage.

For example, on Windows, you could locate the operating system files and *pagefile.sys* on *C:*, the temporary directory and infrequently-used files on *D:*, and database files on drives *E:* and higher.

Change the location of the virtual memory file with Control Panel|System|Performance|Virtual Memory.

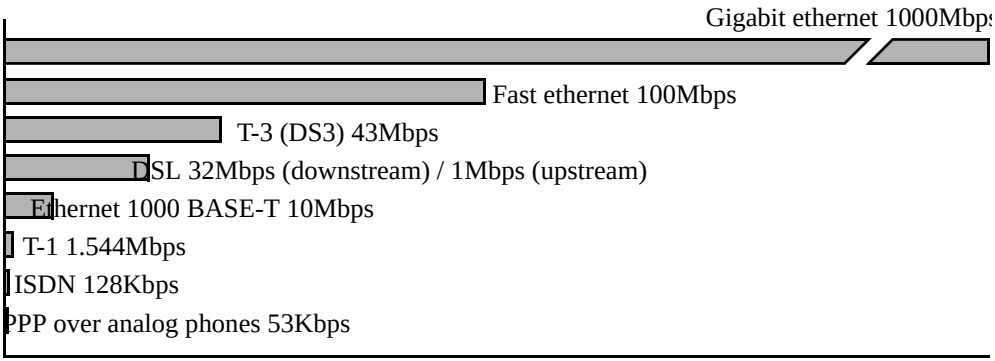
Change the location of the InterBase temporary directory by either specifying a system environment variable INTERBASE_TMP, or editing the *ibconfig* file and specifying the path of the appropriate directory as a value for the TMP_DIRECTORY entry (see **"Configuring Sort Files"**).

Using High-bandwidth Network Systems

For client/server systems, hardware that supports high network bandwidth is as important as I/O capacity. The speed of the network often becomes a bottleneck for performance when many users are making demands on the network simultaneously.

Inexpensive 1000 BASE-T ethernet equipment is common today, but this technology is bare minimum for LAN configuration. It is recommended to use at least 100 Base-T for a high-performance network. The following graph illustrates relative bandwidth rates for various network interface technology.

Figure 12.2 Comparing bandwidth of network interfaces



The maximum bandwidth of gigabit ethernet extends beyond the scale of the graph above.

At the time of this writing, most gigabit ethernet network interface cards (NICs) provide only 600 to 700Mbps bandwidth. Switches, routers, and repeaters also have constrained capacity. It is expected that the state of this technology will continue to improve.

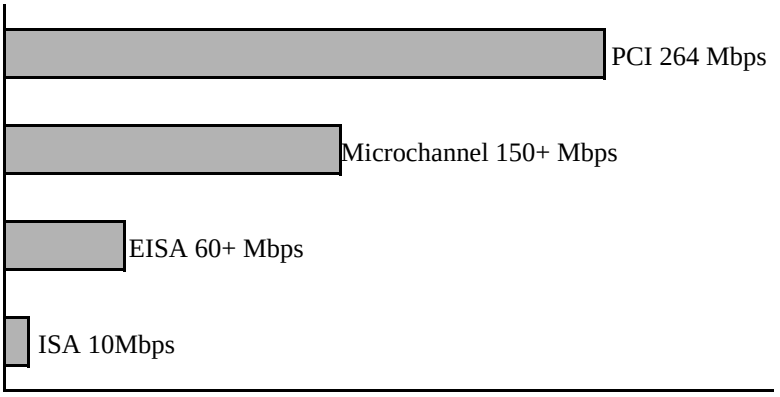
It is recommended that you research reviews and experiment to learn the true throughput of all network hardware in your environment. The slowest component ultimately determines the true throughput.

Tip Network cables develop flaws surprisingly frequently. The result can be sporadic lost packets, for which operating systems compensate by automatically resending packets. This translates into mysterious network performance degradation. You should test network cables regularly. Replacing flawed cables is a low-cost way to keep your network running at peak efficiency.

Using high-performance Bus

Bus is important for both I/O controllers and network interface hardware.

Figure 12.3 Comparing Throughput of Bus Technologies



While 32-bit full-duplex PCI bus is capable of up to 264Mbps, PCI cards actually range from 40Mbps to 130Mbps.

Tip Use controllers on an integrated local PCI bus, it's faster than peripheral cards that plug into the motherboard.

Useful Links

- The T10 Committee home page:
  http://www.t10.com
  This is a useful place to find information on various storage interface technology.
- PC Guide Hard disk interface & configuration:
  http://www.pcguide.com/ref/hdd/if/index.htm
- The SCSI Trade Association:
  http://www.scsita.org
  News and vendor information about the state of SCSI technology and products.
- The Gigabit Ethernet home page:
  http://www.gigabit-ethernet.org/
- The Fibre Channel home page.
  http://www.fibrechannel.com/
  Fibre Channel (FC-AL) is an emerging extended bus technology for network, storage, video transmission, and clustering.

Operating System Configuration

After you have equipped your server hardware appropriately, you should spend time tuning your operating system for server performance.

Disabling Screen Savers

Screen savers can have a serious impact on the performance of a server. Because servers are often set aside in a machine room, it's easy for the performance impact of a screen saver to be overlooked. Screen savers demand a surprising amount of CPU resources to run, and these programs run continuously, 24 hours a day.

Screen savers are evasive in their ability to disappear when a database administrator logs in to the console to diagnose a mysterious drop in performance. The server seems responsive to the database administrator as soon as she touches the server, but the speed degrades soon after she leaves the server.

Not all screen savers have the same performance cost. The Windows *OpenGL* screen savers perform continuous floating-point computations to draw three-dimensional shaded shapes in real time. They demand up to 90% of the system CPU, and cause InterBase and other services to slow to one-tenth their normal speed.

The Windows *Marquee* screen saver is one of the least demanding ones, especially when it is configured to pass text across the screen slowly. Some system administrators like to configure a Marquee on each screen in the machine room, to display the respective machine's hostname. This becomes a machine-name label, in raster form.

A screen saver can also be entertainment, but these should be reserved for workstations. A server in a machine room should be unattended, not used as a workstation.

If you must have phosphor burn protection for a monitor that you leave on, get an *Energy Star* approved monitor that has a power conservation mode. This mode blackens the screen after a configurable period of idleness. This not only protects against phosphor burn, but it conserves power. This is like a simple black screen saver, but it is handled by the electronics of the monitor, instead of by software.

The best option is to simply turn off the monitor when you aren't using it. This saves the phosphors, saves electricity, and decreases the amount of heat in the machine room.

Console Logins

Don't leave the console logged in on a Windows database server. Even if the desktop is idle, it might use as much as 30 percent of the machine's CPU resources just maintaining the interface. You should log out of the server's console when you aren't using it. IBConsole enables you to perform most InterBase maintenance and monitoring tasks from another workstation, without logging in at the server's console.

Sizing a Temporary Directory

When you configure a temporary directory (see **“Managing Temporary Files”**), choose a location that has plenty of free disk space. For some operations such as building an index, InterBase can use a great deal of space for sorting. InterBase can even use an amount of space up to twice the size of your database.

The effects of insufficient temporary space include rapid virtual memory page faults, called *thrashing*, which causes a dramatic performance penalty. Another possible effect is a series of “I/O error” related messages printed to the *InterBase.log* file on the server.

Use a Dedicated Server

Using a server for both workgroup file and print services and as a database server is like letting another user play a video game on your workstation. It detracts from the performance of the workstation, and it’s not the intended use for the machine.

Use a secondary server as the file and print server, and a new machine for the database server. Alternately, use the secondary server for InterBase, depending on the relative priority of these tasks—the database server benefits from having a dedicated machine, even if it is not the fastest model available. Whatever is the most important service should be given the best machine as dedicated hardware.

If performance is a high priority, you can spend money more effectively by buying a dedicated machine instead of trying to increase resources such as RAM on a machine that is providing another competing service. Compare the cost of the hardware with the cost of having less than maximum performance.

Similarly, it is best to put a database on a dedicated drive, so that the database I/O doesn’t compete with the operating system virtual memory paging file or other operating system I/O. See **“Making Drives Specialized”** .

Optimizing Windows for Network Applications

It is recommended to set the Windows server to optimize for network applications. Without this setting, you might see the CPU usage of InterBase peak for a few seconds every InterBase server is configured by default to give priority to filesharing services. You can change this configuration on the server: Control Panel|Network|Services|Server. In the Optimization panel, choose Optimize Throughput For Network Applications.

This change can result in a dramatic improvement of performance for InterBase, as well as other services.

Understanding Windows Server Pitfalls

Windows servers have a peculiar way of balancing processes on SMP machines. If a process is exercising one CPU and the other CPU is relatively idle, Windows NT tries to switch the context of the process to the less burdened CPU. On a dedicated database server, the **ibserver** process is likely to be the only significant user of CPU resources. Unfortunately, Windows still tries to reassign the context of the process to the other CPU in this case. Once Windows has moved the **ibserver** process to the idle CPU, the first CPU becomes less burdened. The Windows server detects this and tries to move **ibserver** back to the first CPU. The second CPU becomes less burdened. This continues many times per minute, and the overhead of switching the process context between the CPUs degrades performance.

There are several possible solutions:

- Run **ibserver** on an SMP server that has enough other duties to occupy the other CPU
- Run **ibserver** only on a single-CPU machine
- Assign CPU affinity to the **ibserver** process:

  a Launch the Task Manager

  b Highlight the **ibserver** process

  c Right-click to raise a window that includes CPU affinity settings

This technique works only if you run **ibserver** as an application, not as a service. If you run InterBase as a service, you must use the Windows API to programmatically set the CPU affinity of the **ibserver** process.

On some operating systems, using a RAM disk is a technique for forcing very heavily used files to be in memory, but still allow them to be opened and closed like any other file. If you consider using a RAM disk on Windows, be aware that the Microsoft RAM disk utility for Windows uses paged memory to allocate the RAM disk. The RAM disk itself can be paged out of RAM and stored on the physical disk in *pagefile.sys*. Therefore, it is futile to use a RAM disk on Windows to create a high-performance file system.

Network Configuration

This section describes performance considerations you should know when configuring a network configuration.

Choosing a Network Protocol

InterBase supports two protocols: TCP/IP when connecting to any server, and NetBEUI when connecting to a Windows server. See **“Network Protocols”** for more details.

NetBEUI

You can use NetBEUI on a network with fewer than 20 users without significant performance costs. Use TCP/IP if you have more active users on your network simultaneously.

NetBEUI is a network protocol designed for use on small local area networks. It is commonly used for filesharing services. It is a connectionless protocol, which means that it broadcasts packets to the entire network. This causes a growing amount of “noise” on a LAN. Noise, from the point of view of any given host, can be defined as network traffic that is not intended for that host. On a LAN with many hosts, enabling NetBEUI can overwhelm the network and reduce the available bandwidth for everyone to use. On most enterprise networks, IT experts discourage the use of NetBEUI.

TCP/IP

TCP/IP is a connection-based protocol, which means packets are routed to the intended recipient. This reduces the saturation of the network and the load on individual hosts. There is effectively more bandwidth available to all hosts, and a large number of hosts can share the same network with less performance penalty.

Configuring Hostname Lookups

Each host on a TCP/IP network has a designated IP address, and TCP/IP traffic is routed to hosts by address. TCP/IP requires a mechanism for clients to translate hostnames to their numeric addresses. Each client host can store the hostname/address associations in a file called *hosts*. You can alternately store this information on a central server, and the clients then retrieve the information on demand using a protocol called DNS. The client requests that the DNS server resolve a hostname, and the server returns the IP address. Then the client can use the IP address to communicate directly with the intended destination. In this configuration, the client must keep only one IP address locally: that of the DNS server host.

Depending on the load on the network and the DNS server itself, hostname resolution can take several seconds. This translates directly into delays when making a network connection. This is related to the message you might see in a web browser, “Looking up host name…” followed by, “Connecting to host name…” This indicates the delay while querying a DNS server to resolve a hostname.

You can speed up hostname resolution by adding the hostname/address mapping of the database server to the *hosts* file on the client computer. The client can resolve the hostname to its address much faster and more reliably by looking it up

in a local file than by querying a service running on another host over the network. This reduces the hostname resolution delay when initiating connections to hosts listed in the local *hosts* file.

Note If you use this technique and later change the address of your database server, you must manually update the hosts files on each client workstation. Depending on the number of workstations in your enterprise, this can be tedious and time consuming. That's why DNS was invented, to centralize TCP/IP address administration. The suggestion to keep the database server address in a local file is intended to provide improved connection performance, but you should be aware of the administrative workload that it requires.

Tip If you object to the general IP address administration tasks required by using TCP/IP (independently from the DNS issue), consider using DHCP to simplify the task of assigning and tracking IP addresses of each host on the network. InterBase works in a DHCP environment as long as the client host has some means to resolve the server's IP address correctly at the time a client application requests an InterBase connection.

Database Properties

Changing database properties can give an improvement in performance without changing anything in the design of your database. Applications require no change in their coding or design. Property changes are transparent to the client and database design.

Choosing a Database Page Size

InterBase pages are 4KB by default. A typical production InterBase database gains 25 to 30 percent performance benefit by using this page size, relative to smaller page sizes. This page size results in better performance for the following reasons:

- Fewer record fragments are split across pages

  It is common for records to be larger than a single page. This means that InterBase fragments records and stores them on multiple pages. Querying a given record requires multiple page reads from the database.

  By increasing the size of a page, InterBase can reduce the number of multiple page reads and can store record fragments more contiguously.

- Index B-trees are more shallow

  Indexes are B-trees of pointers to data pages containing instances of specific indexed values. If the index B-tree is larger than one page, InterBase allocates additional database pages for the index tree. If the index pages are larger,

InterBase needs fewer additional pages to store the pointers. It is easier for the database cache to store the entire B-tree in memory, and indexed lookups are much faster.

- I/O is more contiguous

  It is fairly likely for a query to request successive records in a table. For example, this is done during a table scan, or query that returns or aggregates all records in a table. InterBase stores records on the first page that is unused, rather than ensuring that they are stored near each other in the file. Doing a table scan can potentially require retrieval of data by seeking all over the database. Seeks take time just as reading data takes time.

  Any given page can store records from only one table. This indicates that a larger page is certain to contain more data from the same table, and therefore reading that page returns more relevant data.

- Default number of cache buffers is a larger amount of memory

  InterBase allocates the database cache in number of pages, rather than a fixed number of bytes. Therefore defining a larger page size increases the cache size. A larger cache is more likely to have a better hit rate than a smaller cache.

- Most operating systems perform low-level I/O in 4096 byte blocks

  InterBase performs a page read or write at the OS level by reading in 4096 byte increments regardless of the size of the database page. Therefore, by defining the database with a page size of 4096, the database I/O matches the low-level I/O and this results in greater efficiency when reading and writing pages.

Although 4KB seems to be the best page size for most databases, the optimal size depends on the structure of the specific metadata and the way in which applications access the data. For this reason, you should not consider the 4KB page size guideline to be a magic value. Instead, you should perform testing with your application and database under several different page sizes to analyze which configuration gives the best performance.

Setting the Database Page Fill Ratio

Data pages store multiple versions of data records, as applications update data. When a database is restored, the **gbak** utility fills pages with data only up to 80 percent of the capacity of each page, to leave space for new record version deltas to be stored, hopefully on the same page with the original record. But in a database that is used mostly for reading data rather than updating it, applications never benefit from this 80 percent fill ratio. In this case, it makes sense to restore data using the full capacity of each page. By storing 25 percent more data on each page, it reduces the amount of record fragmentation and increases the amount of data returned in each page read. You can specify the option to use all the space of every page for storing data during a database restore using the command:

```
gbak -c -use_all_space backup_file.ibk database_file.ib
```

Sizing Database Cache Buffers

InterBase maintains a cache in the server's RAM of database pages currently in use. If you have a highly active database, you can gain some performance benefit by raising the default cache from its default of 2048 database pages. As with any cache system, at some point you find diminishing returns. Some experimentation with your particular application and database reveals that point.

See **"Configuring the Database Cache"** for details about server cache settings.

The **ibserver** process running on an InterBase server maintains a cache in memory of recently used data and index pages. Like any cache, it depends on repeated use of data on a given page to help speed up subsequent access. In InterBase SuperServer implementations, the cache is shared by all clients connected to the database.

By default, InterBase allocates enough memory for 2048 pages per database. If the page size of the current database is 4KB, then **ibserver** uses 8MB of memory. If the page size is 8KB, then **ibserver** uses 16MB of RAM for cache. The InterBase API provides a method for any individual client to request that the size of the cache be higher. You can set a property on an individual database that establishes a different default cache size when any client connects to that database:

```
gfix -buffers 5000 database.ib
```

The default of 2048 assumes that the server has a sufficient memory configuration to allocate for 8MB of RAM per database. If memory is less plentiful on your server, or you have many databases that require simultaneous access, you might need to reduce the default number of cache buffers.

It is highly recommended to increase the cache size for a database if you have enough memory to accommodate it. Consider the following points:

- It is not useful to raise the cache size so high that the memory used by **ibserver** starts to page into virtual memory. That defeats the benefit of caching data from disk in memory.
- It is not useful to raise the cache size higher than the number of pages in the database (which you can view with View Database Statistics in IBConsole, or with the **gstat** command-line program). There's no benefit to this, since any given page from disk occupies only one page in the cache, and isn't duplicated.
- One block of memory is allocated for cache per database. If a client connects to two separate databases on one server, the **ibserver** process maintains two separate cache areas of memory. For example, if *database1.ib* has a default cache size of 8000 pages of 4KB each, and *database2.ib* has a default cache size of 10,000 pages of 2KB each, then while both databases have at least one connection, **ibserver** allocates a total of 32MB + 20MB of RAM.

You should experiment with larger cache sizes and analyze the performance improvements. At some point, you will observe diminishing returns. A typical application can achieve up to 30% performance increase from proper cache sizing.

See **"Various InterBase Limits"** for details about cache sizings.

Buffering Database Writes

InterBase on Windows platforms implements a *write-through* cache by default. Every write operation to a page in cache is immediately written out to the operating system's disk I/O, which itself might have a cache.

By contrast, a *write-back* cache defers flushing of the contents of a given cache page until a later time. InterBase performs multiple writes to a cache page in RAM before it writes the page out to disk. This results in better response time for the majority of write operations. Write-back cache consolidates I/O efficiently, and therefore it is much faster than write-through cache.

InterBase offers write-back cache as the default on UNIX and Linux, and as an option on Windows platforms. You can configure this at the database level using **gfix -write async** or by disabling forced writes for the database in IBConsole (Database Properties|General tab|Options).

The real benefit of using asynchronous writes (write-back cache) is about four times performance in the typical case. Some users have reported up to 20 times performance improvement from configuring asynchronous writes, in applications that make heavy use of write operations (INSERT, UPDATE, DELETE). The more writing an application does to the database—including write operations spawned by triggers—the more benefit the application gains.

The risk of asynchronous writes is that data in cache might be lost if the server has a power loss, or if **ibserver** exits abnormally for any reason. Write-through cache protects against data loss, at some performance cost. If you test your server host and client/server application thoroughly and they aren't susceptible to crashes, then it is highly recommended to use asynchronous writes.

Tip Use an uninterruptible power supply (UPS) to help protect your server against sudden power loss. A modest UPS is inexpensive relative to the cost of losing your data, and easy to install. This can allow you to gain the benefits of the asynchronous I/O mode in safety.

Database Design Principles

This section presents guidelines for database design techniques that benefit performance.

Defining Indexes

Proper use of indexes is an important factor in database performance. Effective policies for defining and maintaining indexes can be the key to a very high performance client/server system. The self-tuning nature of indexes in InterBase greatly benefits performance, but you can gain some additional benefit by periodic maintenance tasks.

What is an Index?

An index in InterBase is a Balanced-Tree data structure stored inside the database file that provides a quick lookup mechanism for the location of specific values in a table. Queries make use of appropriate indexes automatically by means of the cost-based optimizer, which analyzes the tables and columns used in a given query and chooses indexes that speed up the searching, sorting, or joining operations.

Defining indexes for some columns is part of designing a production database. Indexes dramatically improve performance of SELECT queries. The greater the number of rows in the table, the greater the benefit of using an index. Intelligently analyzing your database and defining indexes appropriately always improves performance.

Indexes incur a small cost to maintain the index B-tree data structure during INSERT and UPDATE operations. Because of this cost, it is not recommended to be overly liberal with index definitions. Don't create redundant indexes, and don't make an index on every column as a substitute for database usage analysis.

You shouldn't define an index for columns that have few distinct data values. For example, a column FISCAL_QUARTER might have only four distinct values over a potentially very large data set. An index doesn't provide much benefit for retrieval of data with this kind of distribution of values, and the work required to maintain the index tree might outweigh the benefits.

What Queries Use an Index?

InterBase uses indexes to speed up data fetching for the following query elements:

- Primary and foreign keys
- Join keys
- Sort keys, including DISTINCT and GROUP BY
- Search criteria (WHERE)

In general, you should define indexes on all columns that you use in JOIN criteria or as sorting keys in an ORDER BY clause. You don't have to define indexes on primary or foreign key columns, because these table constraints implicitly create indexes.

What Queries Don't Use Indexes?

InterBase doesn't employ an index in the following operations, even if an index exists for the specified columns:

- Search criteria for CONTAINING, LIKE, and <> inequality operations
- Columns used in aggregate functions, like COUNT()
- Other expressions, like UPPER()

Directional Indexes

Indexes are defined as either ASCENDING or DESCENDING. To sort in both directions, you need one index of each type. This is also very important if you are using a scrolling list in a Delphi form, or when using the *TTable.Last* method.

Normalizing Databases

Design your database with proper normalization of data. Records that have lots of repeating groups of fields are larger than they need to be. Large records can increase the cost of sorting, and also cause records to span more pages than is necessary, resulting in more page fragmentation and needlessly large databases.

Denormalized table design can be more convenient for some types of client applications. You can use InterBase *views* and *stored procedures* to in effect store a denormalized query on the server, for convenient access from client applications. Meanwhile, the physical storage of the data is kept in a more efficient, normalized form.

See the ***Data Definition Guide*** for details on views and stored procedures.

Choosing Blob Segment Size

A Blob is a datatype with an unbounded size. It can be many megabytes in size, much larger than any database interface can handle in a single I/O transfer. Therefore, Blobs are defined as a series of segments of uniform size, and the I/O interface transfers Blobs one segment at a time.

Blobs are a special case because there is a special Blob page type, on which other datatypes cannot be stored. The data page for a record containing a Blob stores a Blob ID, which indicates which Blob page the Blob is stored on. A Blob is stored on the same page as the primary record version, if it fits. If it does not fit on that page, special pages are allocated for the Blob--as many as are required--and an index is stored on the primary page. Blob pages are never shared; either a Blob is on a normal data page, or it has a page to itself.

It is advantageous to define a Blob with a segment size equal to the page size. If both the page size and the Blob segment size are 4096 bytes, queries of large Blobs can achieve a data transfer rate of up to 20MB per second. InterBase

ceases to be any kind of bottleneck in this situation; it is more likely that the hardware I/O bus, the network bandwidth, or the middleware are the limiting factors for throughput.

Database Tuning Tasks

This section describes ways you can perform periodic maintenance on your database to keep it running with the best performance.

Tuning Indexes

Periodic maintenance of indexes can improve their performance benefit. You can write SQL scripts to automate these tasks. See **“Using SQL Scripts”** .

Rebuilding Indexes

Periodically, a B-tree data structure might become imbalanced, or it might have some values in the tree that have been deleted from the database (this should not happen in InterBase versions later than 5, due to index garbage collection).

You should periodically rebuild indexes by turning them off and on:

```
ALTER INDEX name INACTIVE;
ALTER INDEX name ACTIVE;
```

Recalculating Index Selectivity

The selectivity of an index is an indicator of its uniqueness. The optimizer uses selectivity in its cost-based analysis algorithm when deciding whether to use a given index in a query execution plan. If the selectivity is out of date and doesn’t accurately represent the state of the index, the optimizer might use or discount the index inappropriately. This doesn’t usually have a great performance penalty unless the selectivity is highly out of date.

You should recalculate the index selectivity if a change to the table affects the average distribution of data values:

```
SET STATISTICS INDEX name;
```

Performing Regular Backups

There are several performance-related benefits to doing periodic backup and restore of an InterBase database. See **“About InterBase backup and restore options”** .

Increasing Backup Performance

- Disable garbage collection if you’re just going to replace the database immediately anyway; this can make the backup execute faster.

- Back up to a different disk drive.

Increasing Restore Performance

- Restore from a different disk drive.
- Disable indexes on restore; this makes the restore execute faster so you have a usable database quickly. You must then have to manually activate the indexes after the restore is complete.

Tip Create a SQL script with all the ALTER INDEX statements necessary to activate your indexes, and keep that handy. Use it like a batch file with **isql -i script.sql** to help automate this procedure. You can create this script with this query:

```
SELECT 'ALTER INDEX ' || RDB$INDEX_NAME || ' ACTIVE;'
  FROM RDB$INDICES
  WHERE RDB$SYSTEM_FLAG = 0 OR RDB$SYSTEM_FLAG IS NULL;
```

You can get the database up and restored more quickly, then activate indexes afterwards. The data is accessible even if the indexes are inactive, but it's slower to query the tables.

Facilitating Garbage Collection

By default, InterBase databases have a built-in function to automatically sweep old record versions when they become too numerous. However, sweeping is partially inhibited by outstanding active transactions. If the server cannot do complete garbage collection, it has to do extra work to maintain each client's snapshot of the database.

Design your client applications to explicitly start and COMMIT transactions promptly, to reduce the number of outstanding transactions.

See **"Overview of Sweeping"** for more details on sweeping, garbage collection, and the database snapshot.

Application Design Techniques

This section describes general application programming methods for InterBase, that help to create high-performance clients.

Using Transaction Isolation Modes

InterBase's multigenerational architecture requires that any query or other operation be associated with an active transaction. Without a transaction, an operation has no context with which to maintain its snapshot of the database. IBConsole tools do a certain amount of automatic transaction management, but it is helpful for performance to manually start and finish transactions.

In the InterBase server engine, a snapshot is generated by making a copy of the state of all other transactions in the database. This snapshot is static for the current transaction. This means that any data committed to the database after the snapshot is created is not visible to operations using that snapshot. This is the repeatable read transaction mode. Two identical queries made at different times are guaranteed to get the same result set, even if other clients are updating data in the database.

Starting a transaction and making a snapshot data structure for the new transaction incurs some amount of overhead. This overhead is magnified when using automatic transaction-handling, because the typical automatic transaction behavior is to start a new transaction and commit it for every statement executed against the database!

Using Correlated Subqueries

Subqueries are SELECT statements which are included as a clause or expression within another statement. They are typically used to generate a value or result set that are used in conditions of the superior query.

A correlated subquery is one in which the conditions of the subquery are different for each row in the parent query, because they depend on values that y from row to row. InterBase executes the subquery many times, once for each row in the parent query. Evaluating each row has a large cost in performance relative to a non-correlated subquery. InterBase optimizes non-correlated subqueries out of the loop, executes once, and uses the result as a fixed dataset.

Example as correlated subquery:

```
SELECT * FROM DEPARTMENT D
WHERE EXISTS (SELECT * FROM EMPLOYEE E
   WHERE E.EMP_NO = D.MNGR_NO AND E.JOB_COUNTRY = 'England')
```

Example as join:

```
SELECT D.*
FROM DEPARTMENT D JOIN EMPLOYEE E
   ON D.MNGR_NO = E.EMP_NO WHERE E.JOB_COUNTRY = 'England'
```

InterBase's optimizer executes a non-correlated subquery once, and uses the result set as many times as necessary in the parent query.

Sometimes a correlated subquery is necessary, given the semantics of the SQL language. However, these types of queries should be used with care and with the understanding that their performance is geometric in relation to the size of the dataset on which they operate.

Preparing Parameterized Queries

Any dynamic SQL (DSQL) statement must go through a cycle of parse, prepare, and execute. You can submit a DSQL statement to go through this process for each invocation, or you can separate the steps. If you have a situation where you execute the same statement multiple times, or the same form of statement with different parameters, you should explicitly prepare the statement once, then execute it as your looping action.

With *parameterized queries*, you can prepare a statement, but defer supplying the specific values for certain elements of the query.

InterBase supports parameterized queries in DSQL, for cases when a given statement is to be executed multiple times with different values. For example, loading a table with data might require a series of INSERT statements with values for each record inserted. Executing parameterized queries has a direct performance benefit, because the InterBase engine keeps the internal representation and optimization of the query after preparing it once.

Use parameterized DSQL queries in Delphi by following these steps:

1 Place a named parameter in the statement with the Delphi *:PARAMETER* syntax. in place of a constant value in a query. InterBase supports parameters in place constants. Tables and column names cannot be parameterized.

2 Prepare the statement. Use the TQuery method *Prepare*. Delphi automatically prepares a query if it is executed without first being prepared. After execution, Delphi unprepares the query. When a query will be executed a number of times, an application should always explicitly prepare the query to avoid multiple and unnecessary prepares and unprepares.

3 Specify parameters. For example, with the TQuery component, use the *ParamByName* method to supply values for each parameter in the query.

4 Execute the statement. SELECT statements should use the *Open* method of TQuery. INSERT, UPDATE, and DELETE statements should use the *ExecSQL* method. These methods prepares the statement in SQL property for execution if it has not already been prepared. To speed performance, an application should ordinarily call Prepare before calling *ExecSQL* for the first time.

5 Repeat steps 3 and 4 as needed.

6 Unprepare the query.

In some real-world cases involving repetitive operations, using parameterized queries has increased performance 100%.

Designing Query Optimization Plans

The *optimization plan* describes the way the optimizer has chosen to execute a query. For certain types of queries, the optimizer might not select the truly optimal plan. A human can analyze different alternate plans and specify a plan overriding

the optimizer's analysis. The result can be amazing improvements in performance for some types of queries. In some dramatic cases, this has been used to reduce a 15 minute query to three seconds.

The elements of plan selection are:

- Assigning indexes
- Combining indexes
- Determining join order
- Generating rivers
- Cost estimation
- Sort merges

InterBase supports syntax with the SELECT expression in embedded SQL and DSQL to allow the user to specify the PLAN for a query. The syntax also works with SELECT statements in the body of a view, a stored procedure, or a trigger.

It is beyond the scope of this chapter to describe in detail the syntax of the PLAN clause for specifying the execution plan, or techniques for analyzing queries manually. The section on SELECT in the *Language Reference* includes some examples of using PLAN.

Deferring Index Updates

Inserting and updating data requires indexes to be updated, which can cause performance to suffer during data INSERT or UPDATE. Some cost incurred while data is entered can result in a big performance win during later data queries.

To minimize the performance hit during INSERT, consider temporarily disabling indexes during high-volume INSERTs. This "turns off" the indexes, making them unavailable to help speed up queries, but also making them not be updated by data INSERTs. Then re-enable the indexes after INSERTing data. This updates and rebalances the indexes once for all the inserted data.

Application Development Tools

This section describes ways you can develop applications that are efficient, using various popular development environments and tools.

InterBase Express™ (IBX)

InterBase engineers have created a full-featured set of data-aware VCL components for use with the *TDataSet* architecture in Delphi. IBX can also be used with Embarcadero's C++ Builder. See the ***Developer's Guide*** for full documentation of InterBase Express.

IB Objects

Another set of VCL components is available for projects with Delphi. It is designed to provide very sophisticated data component technology that is optimized for use with InterBase. The demo product can be downloaded from http://www.ibobjects.com.

Visual Components

This section describes visual components that developers commonly use in Delphi and C++Builder to access data from InterBase. Follow the recommendations below for better client/server performance.

Understanding Fetch-all Operations

In a client/server configuration, a “fetch-all” is the nadir of performance, because it forces BDE to request that the database generate a dataset again and send it over the network.

InterBase and most relational databases do not keep datasets in cache on the server in case the client requests a refresh. InterBase must execute the SQL query again when the BDE requests a refresh. If the query involves a large quantity of data, or complex joining or sorting operations, it is likely to take a long time to generate the dataset.

It is also costly for the server to transfer a large dataset across a network interface. It is more costly by far than it is for a desktop database like Paradox to return a dataset, because a desktop database typically runs locally to the application

It is often the case that software developers choose to use a relational database like InterBase because they are managing a larger amount of data than a desktop database like Paradox can handle efficiently. Naturally, larger datasets take more time to generate and to send over a network.

The person using the client application perceives that it has better performance if the user doesn’t have to wait for refreshes. The less often the client application requests a refresh of the dataset, the better it is for the user.

Important A principle of client/server application design is therefore to reduce the number of costly refresh operations as much as possible.

TQuery

- *CachedUpdates* = False

Allows the server to handle updates, deletes, and conflicts.

- *RequestLive* = False

Setting *RequestLive* to False can prevent the VCL from keeping a client-side copy of rows; this has a benefit to performance because it reduces the network bandwidth requirement

- Below are some operations in which a TQuery perform a fetch-all. Avoid these as much as possible, or be aware of the cost of such operations.

Using the *Locate* method
You should use *Locate* only on local datasets.

Using the *RecordCount* property
It's convenient to get the information on how many records are in a dataset, but when using InterBase, calculation of the *RecordCount* itself forces a fetch-all. For this reason, referencing the *RecordCount* property takes as much time as fetching the entire result dataset of the query.

A common use of *RecordCount* is to determine if the result set of an opened TQuery contains any records, or if it contains zero records. If this is the case, you can determine this without performing a fetch-all by testing for both EOF and BOF states. If both end of file and beginning of file are true for the dataset, then no records are in the result set. These operations do not involve a fetch-all.

For example, for a given *TQuery* instance called *qryTest*:

```
qryTest.Open;
if qryTest.BOF and qryTest.EOF then begin
  // There are no result set records.
end
else begin
  // There are some result set records.
end;
```

Using the *Constraints* property
Let the server enforce the constraint.

Using the *Filter* property
For the TQuery to filter records, it must request a much larger dataset than that which it subsequently displays. The InterBase server can perform the filtering in a much more efficient manner before returning the filtered dataset. You should use a WHERE clause in your SQL query. Even if you use a WHERE clause, any use of the *TQuery.Filter* property still forces a fetchall.

TTable

The *TTable* component is designed for use on relatively small tables in a local database, accessed in core memory. *TTable* gathers information about the metadata of the table, and tries to maintain a cache of the dataset in memory. *TTable* refreshes its client-side copy of data when you issue the *TTable.post* method and when you use the *TDatabase.rollback* method. This incurs a huge network overhead for client/server databases, which tend to have larger datasets and are accessed over a network. You can observe the activity of *TTable* with the SQL Monitor tool. This reports all calls to the BDE and InterBase API.

Though *TTable* is very convenient for its RAD methods and its abstract data-aware model, you should use it sparingly with InterBase or any other client/server database. *TTable* was not designed to be used for client/server applications.

Appendix

A

Migrating to InterBase

InterBase is a mature product that was originally architected before current standards came into existence. As the standards evolved, it became clear that bringing InterBase into compliance with them would produce a somewhat challenging migration path.

With the advent of InterBase 6, InterBase 6 introduced an increased compliance with the SQL-92 standard, but migrating older (InterBase 5 and earlier) clients and databases might, in some cases, require considerable attention to detail.

The feature areas affected are: the use of double quotes, which are now reserved for delimited identifiers; the meaning of the DATE datatype; the behavior of exact numeric datatypes, and the existence of new keywords that might conflict with older metadata names.

This document describes how to plan and execute a smooth migration from earlier versions of InterBase to InterBase 6 or later.

The earlier pages of this guide discuss the issues involved in the migration. Near the end, you will find detailed, step-by-step instructions for both in-place migration and for migrating an old database to a new one. See **Migrating Servers and Databases**, **Migrating Databases to Dialect 3**, and **Migrating Clients.**

Migration Process

These are the steps you must take to migrate servers, databases, and clients. Each is discussed in detail in later sections:

Server and Database Migration

1 Backup all databases to be migrated

2. Install the latest InterBase server
3. Restore databases to be migrated using the most recent **gbak**; at this point, you have dialect 1 databases
4. Validate migrated databases
5. Migrate databases to SQL dialect 3 (**Migrating Databases to Dialect 3)**

Client Migration

1. Identify the clients that must be upgraded.
2. Identify areas in your application which may need upgrading.
3. Install the InterBase client to each machine that requires it.
4. Upgrade SQL applications to SQL dialect 3.

Migration Issues

Before migrating your databases, you need to learn about InterBase SQL dialects and understand their effect on servers, clients, and the use of certain features introduced in InterBase 6 and later.

InterBase SQL Dialects

InterBase recognizes different client and database dialects to allow users more mobility in how their legacy databases are used, accessed, and updated. Beginning with InterBase 6, each client and database has a SQL *dialect*: an indicator that instructs an InterBase 6 or later server how to interpret *transition features*: those features whose meanings have changed between InterBase versions. The following transition features have different meanings based on the dialect used by the client applications:

- Double quote (“): changed from a synonym for the single quote (‘) to the delimiter for an object name.
- DECIMAL and NUMERIC datatypes with precision greater than 9: now stored as INT64 datatypes instead of DOUBLE PRECISION.
- DATE, TIME, and TIMESTAMP datatypes: DATE has changed from a 64-bit quantity containing both date and time information to a 32-bit quantity containing only date information. TIME is a 32-bit quantity containing only time information, while TIMESTAMP is a 64-bit quantity containing both date and time information (the same as DATE in pre-Version 6 SQL).

Clients and Databases

Clients and databases each have dialects. Servers do not themselves have a dialect, but they interpret data structures and client requests based on the dialect of each. Applications using an older version of the InterBase client work with InterBase 6 and later servers and their databases with some restrictions:

- Version 5 clients cannot access dialect 3 columns that are stored as INT64, TIME, or DATE. (DECIMAL and NUMERIC columns with precision greater than 9 are stored as INT64.)
- Version 5 clients cannot display new datatypes in metadata using the SHOW command, or any equivalent.
- Version 5 clients interpret the DATE datatype as TIMESTAMP, since that was the definition of DATE prior to InterBase 6.
- Version 5 clients cannot access any object named with a delimited identifiers.

Keywords Used as Identifiers

Version 5 clients have one advantage over version 6 clients: If you migrate an older database that uses some version 6 keywords as identifiers to version 6 dialect 1, these older version 5 clients can still access those keyword objects. Version 6 dialect 1 cannot do so. Dialect 3 clients can access these keyword objects if the objects are delimited in double quotes.

If version 5 clients use any InterBase 6 or 7 keywords as object names, the InterBase 6 server permits this without error because it recognizes that these clients were created at a time when these were not keywords.

Example For example, the following statement uses the new keyword word TIME:

SELECT TIME FROM atable;

This statement, when executed via a pre-InterBase 6 client returns the information as it did in previous versions. If this same query is issued using a version 6 or 7 client, an error is returned since TIME is now a reserved word. See the **New InterBase Keywords**.

Understanding SQL Dialects

Below are explanations of server and client behavior with SQL dialects 1, 2, and 3.

Dialect 1 Clients and Databases

In dialect 1, the InterBase 6 and later servers interpret transition features exactly as an InterBase 5 server does:

- Double quoted text is interpreted as a string literal. Delimited identifiers are not available.

- The DATE datatype contains both time and date information and is interpreted as TIMESTAMP; the name has changed but the meaning has not. Dialect 1 clients expect the entire timestamp to be returned. In dialect 1, DATE and TIMESTAMP are identical.
- The TIME datatype is not available.
- Dialect 1 databases store DECIMAL and NUMERIC datatypes with precision greater than 9 as DOUBLE PRECISION, not INT64.
- Dialect 1 clients expect information stored DECIMAL and NUMERIC datatypes to be returned as double precision; such clients cannot create database fields to hold 64-bit integers.

InterBase 6 and later servers recognize all the other InterBase features in dialect 1 clients and databases.

Dialect 2 Clients

Dialect 2 is available only on the client side. It is intended for assessing possible problems in legacy metadata that is being migrated to dialect 3. To determine where the problem spots are when you migrate a database from dialect 1 to dialect 3, you extract the metadata from the database, set **isql** to dialect 2, and then run that metadata file through **isql**. **isql** issues warning whenever it encounters double quotes, DATE datatypes, or large exact numerics to alert you to places where you might need to change the metadata in order to make a successful migration to dialect 3.

To detect problem areas in the metadata of a database that you are migrating, extract the metadata and run it through a dialect 2 client, which will report all instances of transition features. For example:

```
isql -i v5metadata.sql
```

Do not assign dialect 2 to databases.

Dialect 3 Clients and Databases

In dialect 3, the InterBase server interprets transition features as InterBase 6 SQL 92-compliant:

- Double quoted strings are treated as delimited identifiers.
- Dialect 3 DATE datatype fields contain only date information. Dialect 3 clients expect only date information from a field of datatype DATE.
- The TIME datatype is available, and stores only time information.
- Dialect 3 databases store DECIMAL and NUMERIC datatypes with precision greater than 9 as INT64 *if and only if they are in columns that were created in dialect 3*.
- Dialect 3 clients expect DECIMAL and NUMERIC datatypes with precision greater than 9 to be returned as INT64.

To learn how to migrate older data to INT64 storage, see **Do you really need to migrate your NUMERIC and DECIMAL datatypes?** and **Migrating NUMERIC and DECIMAL datatypes**.

Setting SQL Dialects

You can set the SQL dialect for a server or client in a variety of ways. For example, the IBConsole user interface has menu options for specifying the SQL dialect. See **IBConsole: The InterBase Interface** for a complete explanation of using IBConsole. This section explores the command-line methods for setting a dialect.

Setting the isql Client Dialect

To use **isql** to create a database in a particular dialect, first set **isql** to the desired dialect and then use it to create the database.You can set **isql** dialect in the following ways:

- On the command line, start **isql** with option **-sql_dialect *n***, where *n* is 1, 2, or 3.

  isql -sql_dialect *n*

- Within an **isql** session or in a SQL script, you can issue this statement:

  SET SQL DIALECT *n*

The following table shows the precedence for setting **isql** dialect:

Table 13.1 isql Dialect Precedence

Ranking	How dialect is set
Lowest	Dialect of an attached Version 6 and later database
Next lowest	Dialect specified on the command line: `isql -sql_dialect n`
Next highest	Dialect specified during the session: `SET SQL DIALECT n;`
Highest	Dialect of an attached Version 5 database (=1)

In InterBase 6 and later, **isql** has the following behavior with respect to dialects:

- If you start **isql** and attach to a database without specifying a dialect, **isql** takes on the dialect of the database.
- If you specify a dialect on the command line when you invoke **isql**, it retains that dialect after connection unless explicitly changed.
- When you change the dialect during a session using SET SQL DIALECT *n*, **isql** continues to operate in that dialect until explicitly changed.

- When you create a database using **isql**, the database is created with the dialect of the **isql** client; for example, if **isql** has been set to dialect 1, when you create a database, it is a dialect 1 database.
- If you create a database without first specifying a dialect for the **isql** client or attaching to a database, **isql** creates the database in dialect 3.

The statements above are true whether you are running **isql** as a command-line utility or are accessing it through IBConsole, InterBase's new interface.

Important Any InterBase 6 and later **isql** client that attaches to a version 5 database resets to dialect 1.

Setting the gpre Dialect

In InterBase 6 and later, **gpre's** default behavior is to take on the dialect of the database to which it is connected. This enables **gpre** to parse pre-Version 6 source files without moderation.

There are two ways to change the dialect of **gpre**:

- Start **gpre** with option **-sql_dialect *n***. For example, this command sets **gpre** to dialect 3:

  gpre -sql_dialect 3

- Specify dialect within the source, for example:

```
EXEC SQL
  SET SQL DIALECT n
```

The dialect precedence for **gpre** is as follows:

Lowest	Dialect of an attached database
Middle	Command line specification: gpre -sql_dialect n

Highest	Dialect explicitly specified within the source, for example
	`EXEC SQL`
	`SET SQL DIALECT n`

Setting the Database Dialect

To set the dialect of an ODS 10 or later database, attach to the database as either the owner or SYSDBA. Use **gfix** with the command-line option **-sql_dialect *n***, where *n* is 1 or 3. For example, the following statement sets *mydb.ib* to dialect 3:

gfix -sql_dialect 3 mydb.ib

See **Migrating Databases to Dialect 3** for details about issues to consider before you issue the command.

Features and Dialects

Many of the features introduced in InterBase 6 and later operate without reference to dialect. Other features are dialect-specific. The dialect-specific features are discussed below:

Features Available in All Dialects

The following new features are available in both dialect1 and dialect 3:

IBConsole, InterBase's Graphical Interface

IBConsole, InterBase's graphical user interface, combines the functionality of the older Server Manager and InterBase Windows isql. You now create and maintain databases, configure and maintain servers, and execute interactive SQL from one integrated interface.

Read-only Databases

You can make InterBase 6 and later databases be read-only. This permits distribution on read-only media such as CDROMs and reduces the chance of accidental or malicious changes to databases.

Altering Column Definitions

The ALTER COLUMN clause of the ALTER TABLE statement can change a column's name, datatype, or position.

Altering Domain Definitions

ALTER DOMAIN now includes the ability to change the name or datatype of a domain definition.

The EXTRACT() Function

The new EXTRACT() function extracts information from the new DATE, TIMESTAMP, and TIME datatypes. In dialect 1, you can use it to extract information from the TIMESTAMP datatype.

Note "DATE" is new in the sense that it has a different meaning in dialect 3 databases than it did previously.

SQL Warnings

The InterBase API function set now returns warnings and informational messages along with error messages in the status vector.

The Services API, Install API, and Licensing API

InterBase now provides three new function libraries. The Services API, which is part of the InterBase client library, provides functions for database maintenance tasks, software activation, requesting information about the configuration of databases and server, and working with user entries in the security database.

New gbak Functionality

In InterBase 6 and later, **gbak's** functionality has been extended. gbak can now perform all of the following actions:

- Back up to multiple files and restore to multiple files
- Perform server-side backups and restores using the **-service** switch
- Set databases to read-only mode when restoring

InterBase Express™ (IBX)

IBX provides native Delphi components for InterBase data access, services, and installation. Borland C++ Builder also can access IBX components.

Features Available Only in Dialect 3 Databases

The following features are available only in dialect 3 clients and databases because they conflict with dialect 1 usage.

Delimited Identifiers

Identifiers can now be keywords, contain spaces, be case sensitive, and contain non-ASCII characters. Such identifiers must be delimited by double quotes. String constants must be delimited by single quotes.

INT64 Data Storage

In dialect 3 databases, data stored in DECIMAL and NUMERIC columns is stored as INT64 when the precision is greater than 9. This is true only for columns that are *created* in dialect 3. These same datatypes are stored as DOUBLE PRECISION in dialect 1 and in all older InterBase versions. This change in storage also requires different arithmetic algorithms.

DATE and TIME Datatypes

In dialect 3, the DATE datatype holds only date information. This is a change from earlier InterBase versions in which it stored the whole timestamp.

Dialect 3 allows the use of the TIME datatype, which hold only the time portion of the timestamp.

New InterBase Keywords

InterBase 6, 6.5, and 2007 introduced the following new keywords:

BOOLEAN	HOUR	TIMESTAMP
COLUMN	MINUTE	TRUE
CURRENT_DATE	MONTH	TYPE
CURRENT_TIME	PERCENT	UNKNOWN
CURRENT_TIMESTAMP	ROWS	WEEKDAY
DAY	SECOND	YEAR
EXTRACT	TIES	YEARDAY
FALSE	TIME	

InterBase 7.5 and 9 introduced the following new keywords:

CASE	NULLIF
COALESCE	PRESERVE
DECRYPT	TEMPORARY
ENCRYPT	
ENCRYPTION	
GLOBAL	

These keywords are reserved words in all dialects.

- Beginning with InterBase 6, you cannot create objects in a dialect 1 database that have any of these keywords as object names (identifiers).
- You can migrate a version 5 database that contains these keywords used as identifiers to version 6 or later dialect 1 without changing the object names: a column could be named “YEAR”, for instance.
  - Version 5 clients can access these keyword identifiers without error.
  - Version 6 and later clients *cannot* access keywords that are used as identifiers. In a dialect 1 database, you must change the names so that they are not keywords.
  - If you migrate directly to dialect 3, you can retain the names, but you must delimit them with double quotes. To retain accessibility for older clients, put the names in all upper case. Delimited identifiers are case sensitive.
- Although TIME is a reserved word in version 6 and later dialect 1, you cannot use it as a datatype because such databases guarantee datatype compatibility with version 5 clients.
- In dialect 3 databases and clients, any reserved word can be used as an identifier as long as it is delimited with double quotes.

Delimited Identifiers

To increase compliance with the SQL 92 standard, InterBase 6 and later introduces *delimited identifiers*. An identifier is the name of any database object; for instance a table, a column, or a trigger. A delimited identifier is an identifier that is enclosed in double quotes. Because the quotes delimit the boundaries of the name, the possibilities for object names are greatly expanded from previous versions of InterBase. Object names can now:

- mimic keywords
- include spaces (except trailing spaces)
- be case sensitive

How Double Quotes have Changed

Up to and including version 5, InterBase allowed the use of either single or double quotes around string constants. The concept of delimited identifiers did not exist. Beginning with InterBase 6 (dialect 3), anything in single quotes is interpreted as a string constant and anything in double quotes is interpreted as a delimited identifier. Here is the summary:

- In all versions of InterBase, anything inside single quotes is treated as a string constant.
- In InterBase version 5 and older, anything within double quotes is treated as a string constant, because those versions do not have the concept of a delimited identifier.
- Version 6 dialect 1 is a transition mode that behaves like older versions of InterBase with respect to quote marks: it interprets strings within either single or double quotes as string constants.
- Beginning with version 6 dialect 3, InterBase interprets anything inside double quotes as a delimited identifier. Anything inside single quotes is interpreted as a string constant.
- When InterBase servers version 6 or later detect that the client is dialect 1, they permit client DML (data manipulation) statements to contain double quotes and they correctly handle these as string constants. However, they do not permit double quotes in client DDL (data definition) statements because that metadata would not be allowed in dialect 3. Version 6 servers all insist that string constants be delimited with single quotes when clients create new metadata.

DATE, TIME, and TIMESTAMP Datatypes

InterBase 6 and later dialect 3 replaces the old InterBase DATE datatype, which contains both date and time information, with SQL-92 standard TIMESTAMP, DATE, and TIME datatypes. The primary migration problem exists in the source code of

application programs that use the InterBase 5 DATE datatype. In InterBase 6 and later, the DATE keyword represents a date-only datatype, while a Version 5 DATE represents a date-and-time datatype.

Columns and domains that are defined as DATE datatype in InterBase 5 DATE appear as TIMESTAMP columns when the database is restored in InterBase 6. However, a TIMESTAMP datatype has four decimal points of precision, while a Version 5 DATE datatype has only two decimal points of precision.

If you migrate your database to dialect 3 and you require only date or only time information from a TIMESTAMP column, you can use ALTER COLUMN to change the datatype to DATE or TIME. These columns each take only four bytes, whereas TIMESTAMP and the InterBase 5 DATE columns each take eight bytes. If your TIMESTAMP column holds both date and time information, you cannot change it to an InterBase 6 and later DATE or TIME column using ALTER COLUMN, because ALTER COLUMN does not permit data loss. Dialect use also enforces certain rules:

- In dialect 1, only TIMESTAMP is available. TIMESTAMP is the equivalent of the DATE datatype in previous versions. When you back up an older database and restore it in version 6 and later, all the DATE columns and domains are automatically restored as TIMESTAMP. DATE and TIMESTAMP datatypes are both available and both mean the same thing in dialect 1.
- In dialect 3, TIMESTAMP functions as in dialect 1, but two additional datatypes are available: DATE and TIME. These datatypes function as their names suggest: DATE holds only date information and TIME holds only time.
- In dialect 3, DATE and TIME columns require only four bytes of storage, while TIMESTAMP columns require eight bytes.

The following example shows the differences between dialect 1 and dialect 3 clients when date information is involved.

Example CREATE TABLE table1 (fld1 DATE, fld2 TIME);

INSERT INTO table1 VALUES (CURRENT_DATE, CURRENT_TIME);

Using dialect 1 clients

SELECT * FROM table1;

```
  Statement failed, SQLCODE = -804
  Dynamic SQL Error
  -SQL error code = -804
  -datatype unknown
  -Client SQL dialect 1 does not support reference to TIME
datatype
```

SELECT fld1 FROM table1;

```
  Statement failed, SQLCODE = -206
  Dynamic SQL Error
  -SQL error code = -206
  -Column unknown
```

```
-FLD1
-Client SQL dialect 1 does not support reference to DATE datatype
```

Using dialect 3 clients

SELECT * FROM table1;

```
FLD1            FLD2
=========== =============
1999-06-25  11:32:30.0000
```

SELECT fld1 FROM table1;

```
FLD1
===========
1999-06-25
```

Example CREATE TABLE table1 (fld1 TIMESTAMP);

```
INSERT INTO table1 (fld1) VALUES (CURRENT_TIMESTAMP);
```

SELECT * FROM table1;

In dialect 1

```
FLD1
===========
25-JUN-1999
```

In dialect 3

```
FLD1
=========================
1999-06-25 10:24:35.0000
```

Example SELECT CAST (fld1 AS CHAR(5)) FROM table1;

In dialect 1

```
======
25-JU
```

In dialect 3

```
Statement failed, SQLCODE = -802
arithmetic exception, numeric overflow, or string
truncation
```

Converting TIMESTAMP Columns to DATE or TIME

Once you have migrated a database to dialect 3, any columns that previously had the DATE datatype will have the TIMESTAMP datatype. If you want to store that data in a DATE or TIME column, follow these steps:

1. Use ALTER TABLE to create a new column of the desired type.
2. Insert the values from the original column into the new column:

   UPDATE *tablename* SET *new_field* = CAST (*old_field* AS *new_field*);
3. Use ALTER TABLE to drop the original column.
4. Use ALTER TABLE … ALTER COLUMN to rename the new column.

Casting Date/time Datatypes

InterBase 6 and later dialect 3 no longer allows the use of the CAST operator to remove the date portion of a timestamp by casting the timestamp value to a character value. When you cast a TIMESTAMP to a CHAR or CHAR in dialect 3, the destination type must be at least 24 characters in length or InterBase will report a string overflow exception. This is required by the SQL3 standard.

You can use the CAST() function in SELECT statements to translate between date/time datatypes and various character-based datatypes. The character datatype must be at least 24 characters in length. You can, however, cast a TIMESTAMP to a DATE and then cast the DATE to a CHAR of less than 24 characters. For example:

```
SELECT CAST (CAST (timestamp_col AS DATE) AS CHAR(10)) FROM table1;
```

It is not possible to cast a date/time datatype to or from BLOB, SMALLINT, INTEGER, FLOAT, DOUBLE PRECISION, NUMERIC, or DECIMAL datatypes.

For more information, refer to “Using CAST() to convert dates and times” in the ***Embedded SQL Guide***.

Table 13.2 outlines the results of casting *to* date/time datatypes:

Table 13.2 Results of Casting to Date/time Datatypes

	To		
Cast From	**TIMESTAMP**	**DATE**	**TIME**
CHAR(*n*) CHARACTER(*n*) CSTRING(*n*)	String must be in format YYYY-MM-DD HH:MM:SS.thousands	See below.	String must be in format HH:MM:SS.thousands
TIMESTAMP	Always succeeds	Date portion of timestamp	Time portion of timestamp
DATE	Always succeeds; time portion of timestamp set to 0:0:0.0000	Always succeeds	Error
TIME	Always succeeds; date portion of timestamp set to current date	Error	Always succeeds

Casting DATE to string results in YYYY-MM-DD where “MM” is a two-digit month. If the result does not fit in the string variable a string truncation exception is raised. In earlier versions, this case results in DD-Mon-YYYY HH:mm:SS.hundreds where “Mon” was a 3-letter English month abbreviation. Inability to fit in the string variable resulted in a silent truncation.

Casting a string to a date now permits strings of the form:

```
'yyyy-mm-dd'  'yyyy/mm/dd'  'yyyy mm dd'
'yyyy:mm:dd'  'yyyy.mm.dd'
```

In all of the forms above, you can substitute a month name or 3-letter abbreviation in English for the 2-digit numeric month. However, the order must always be 4-digit year, then month, then day.

In previous versions of InterBase, you could enter date strings in a number of forms, including ones that had only two digits for the year. Those forms are still available in InterBase 6 and later. If you enter a date with only two digits for the year, InterBase uses its “sliding window” algorithm to assign a century to the years.

The following forms were available in earlier versions of InterBase and are still permitted in InterBase 6 and later:

```
'mm-dd-yy''mm-dd-yyyy''mm/dd/yy''mm/dd/yyyy'
'mm dd yy''mm dd yyyy''mm:dd:yy''mm:dd:yyyy'
'dd.mm.yy' 'dd.mm.yyyy'
```

If you write out the month name in English or use a three-character English abbreviation, you can enter either the month or the day first. In the following examples, “xxx” stands for either a whole month name or a three-letter abbreviation. All of the following forms are acceptable:

```
'dd-xxx-yy''dd-xxx-yyyy''xxx-dd-yy''xxx-dd-yyyy'
'dd xxx yy''dd xxx yyyy''xxx dd yy''xxx dd yyyy'
'dd:xxx:yy' 'dd:xxx:yyyy'  'xxx:dd:yy'     'xxx:dd:yyyy'
```

For example, the following INSERT statements all insert the date “January 22, 1943”:

```
INSERT INTO t1 VALUES ('1943-01-22');
INSERT INTO t1 VALUES ('01/22/1943');
INSERT INTO t1 VALUES ('22.01.1943');
INSERT INTO t1 VALUES ('jan 22 1943');
```

The following statement would enter the date “January 22, 2043”:

```
INSERT INTO t1 VALUES ('01/22/43');
```

Table 13.3 outlines the results of casting *from* date/time datatypes:

Table 13.3 Results of Casting to Date/time Datatypes

Cast From	To CHAR(*n*), CHARACTER (*n*), or CSTRING(*n*)
TIMESTAMP	Succeeds if *n* is 24 or more. Resulting string is in format YYYY-MM-DD HH:MM:SS.thousands.
DATE	Succeeds if *n* is 10 or more. Resulting string is in the format YYYY-MM-DD.
TIME	Succeeds if *n* is 13 or more. Resulting string is the format HH:MM:SS.thousands.

Adding and Subtracting Datetime Datatypes

The following table shows the result of adding and subtracting DATE, TIME, TIMESTAMP, and numeric values. “Numeric value” refers to any value that can be cast as an exact numeric value by the database engine (for example, INTEGER, DECIMAL, or NUMERIC).

Table 13.4 Adding and Subtracting Date/time Datatypes

Operand1	Operator	Operand2	Result
DATE	+	DATE	Error
DATE	+	TIME	TIMESTAMP (concatenation)
DATE	+	TIMESTAMP	Error
DATE	+	Numeric value	DATE + number of days: fractional part ignored
TIME	+	DATE	TIMESTAMP (concatenation)
TIME	+	TIME	Error
TIME	+	TIMESTAMP	Error
TIME	+	Numeric value	TIME + number of seconds: 24-hour modulo arithmetic
TIMESTAMP	+	DATE	Error
TIMESTAMP	+	TIME	Error
TIMESTAMP	+	TIMESTAMP	Error
TIMESTAMP	+	Numeric value	TIMESTAMP: DATE + number of days; TIME + fraction of day converted to seconds
DATE	–	DATE	DECIMAL(9,0) representing the number of days
DATE	–	TIME	Error
DATE	–	TIMESTAMP	Error
DATE	–	Numeric value	DATE: number of days; fractional part ignored
TIME	–	DATE	Error
TIME	–	TIME	DECIMAL(9,4) representing number of seconds
TIME	–	TIMESTAMP	Error

Table 13.4 Adding and Subtracting Date/time Datatypes

Operand1	Operator	Operand2	Result
TIME	–	Numeric value	TIME: number of seconds; 24-hour modulo arithmetic
TIMESTAMP	–	DATE	Error
TIMESTAMP	–	TIME	Error
TIMESTAMP	–	TIMESTAMP	DECIMAL(18,9) representing days and fraction of day
TIMESTAMP	–	Numeric value	TIMESTAMP: DATE – number of days; TIME: fraction of day converted to seconds

Note Numeric value + DATE, TIME, or TIMESTAMP is symmetric to DATE, TIME, or TIMESTAMP + numeric value.

Using Date/time Datatypes with Aggregate Functions

You can use the date/time datatypes with the MIN(), MAX(), COUNT() functions, the DISTINCT argument to those functions, and the GROUP BY argument to the SELECT() function. An attempt to use SUM() or AVG() with date/time datatypes returns an error.

Default Clauses

CURRENT_DATE, CURRENT_TIME, and CURRENT_TIMESTAMP can be specified as the default clause for a domain or column definition.

Extracting Date and Time Information

The EXTRACT() function extracts date and time information from databases. In dialect 3, the EXTRACT operator allows you to return different parts of a TIMESTAMP value. The EXTRACT operator makes no distinction between dialects when formatting or returning the information. EXTRACT() has the following syntax:

EXTRACT (*part* FROM *value*)

The value passed to the EXTRACT() expression must be DATE, TIME, or TIMESTAMP. Extracting a part that doesn't exist in a datatype results in an error. For example:

```
EXTRACT (TIME FROM aTime)
```

A statement such as EXTRACT (YEAR from aTime) would fail.

The datatype of EXTRACT() expressions depends on the specific part being extracted:

Table 13.5 Extracting Date and Time Information

Extract	Resulting datatype	Representing
YEAR	SMALLINT	Year, range 0-5400
MONTH	SMALLINT	Month, range 1-12
DAY	SMALLINT	Day, range 1-31
HOUR	SMALLINT	Hour, range 1-23
MINUTE	SMALLINT	Minute, range 1-59
SECOND	DECIMAL(6,4)	Second, range 0-59.9999
WEEKDAY	SMALLINT	Day of the week, range 0-6 (0 = Sunday, 1 = Monday, and so on)
YEARDAY	SMALLINT	Day of the year, range 1-366

```
SELECT EXTRACT (YEAR FROM timestamp_fld) FROM table_name;
  =======
  1999

SELECT EXTRACT (YEAR FROM timestamp_fld) FROM table_name;
  =======
  1999

SELECT EXTRACT (MONTH FROM timestamp_fld) FROM table_name;
  =======
  6

SELECT EXTRACT (DAY FROM timestamp_fld) FROM table_name;
  =======
  25

SELECT EXTRACT (MINUTE FROM timestamp_fld) FROM table_name;
  =======
  24

SELECT EXTRACT (SECOND FROM timestamp_fld) FROM table_name;
  ============
  35.0000

SELECT EXTRACT (WEEKDAY FROM timestamp_fld) FROM table_name;
  =======
  5

SELECT EXTRACT (YEARDAY FROM timestamp_fld) FROM table_name;
  =======
```

```
   175

SELECT EXTRACT (MONTH FROM timestamp_fld) ||
 '-' || EXTRACT (DAY FROM timestamp_fld) ||
'-' || EXTRACT (YEAR FROM timestamp_fld) FROM table_name;

   ====================
   6-25-1999
```

DECIMAL and NUMERIC Datatypes

The following sections highlight some of the changes introduced by InterBase 6 and later when dealing with numeric values. They need to be considered carefully when migrating your database from dialect 1 to dialect 3. When considering these issues, keep in mind that in order to make use of the new functionality, the statements must be created with a client dialect setting of 3.

The most notable migration issues involve using the division operator and the AVG() function (which also implies division) with exact numeric operands. *Exact numeric* refers to any of the following datatypes: INTEGER, SMALLINT, DECIMAL, NUMERIC. NUMERIC and DECIMAL datatypes that have a precision greater than 9 are called “large exact numerics” in this discussion. Large exact numerics are stored as DOUBLE PRECISION in dialect 1 and as INT64 in columns created in dialect 3.

Important When you migrate an exact numeric column to dialect 3 it is still stored as DOUBLE PRECISION. The migration does not change the way the data is stored because INT64 cannot store the whole range that DOUBLE PRECISION can store. There is potential data loss, so InterBase does not permit direct conversion. If you decide that you want your data stored as INT64, you must create a new column and copy the data. Only exact numeric columns that are *created* in dialect 3 are stored as INT64. The details of the process are provided in **Migrating Databases to Dialect 3**.

You might or might not want to change exact numeric columns to INT64 when you migrate to dialect 3. See **“Do you really need to migrate your NUMERIC and DECIMAL datatypes?”** for a discussion of issues.

Dialect 3 features and changes include

- Support for 64 bit integers.
- Overflow protection. In dialect 1, if the product of two integers was bigger than 31 bits, the product was returned modulo 2^{32}. In dialect 3, the true result is returned as a 64-bit integer. Further, if the product, sum, difference, or quotient of two exact numeric values is bigger than 63 bits, InterBase issues an arithmetic overflow error message and terminates the operation. (Previous versions sometimes returned the least-significant portion of the true result.). The stored procedure **bignum** below demonstrates this.

Operations involving division return an exact numeric if both operands are exact numerics in dialect 3. When the same operation is performed in dialect 1, the result is a DOUBLE PRECISION.

To obtain a DOUBLE PRECISION quotient of two exact numeric operands in dialect 3, explicitly cast one of the operands to DOUBLE PRECISION before performing the division:

CREATE TABLE table 1 (n1 INTEGER, n2 INTEGER);

INSERT INTO table 1 (n1, n2) VALUES (2, 3);

SELECT n1 / n2 FROM table1;

```
======================
                0
```

Similarly, to obtain a double precision value when averaging an exact numeric column, you must cast the argument to double precision before the average is calculated:

SELECT AVG(CAST(int_col AS DOUBLE PRECISION))FROM table1;

Compiled Objects

The behavior of a compiled object such as a stored procedure, trigger, check constraint, or default value depends on the dialect setting of the client at the time the object is compiled. Once compiled and validated by the server the object is stored as part of the database and its behavior is constant regardless of the dialect of the client that calls it.

Example Consider the following procedure:

```
CREATE PROCEDURE exact1 (a INTEGER, b INTEGER) RETURNS (c
INTEGER)
AS BEGIN
  c = a / b;
  EXIT;
END;
```

When created by a dialect 1 client
EXECUTE PROCEDURE *exact 1* returns 1 when executed by either a dialect 1 or dialect 3 client.

When created by a dialect 3 client
EXECUTE PROCEDURE *exact 1* returns 0 when executed by either a dialect 1 or dialect 3 client.

Example Consider the following procedure:

```
CREATE PROCEDURE bignum (a INTEGER, b INTEGER) RETURNS (c
NUMERIC(18,0)
AS BEGIN
  c = a * b;
```

```
  EXIT;
END;
```

When created by a dialect 1 client
EXECUTE PROCEDURE *bignum* (65535, 65535) returns –131071.0000 when executed by either a dialect 1 or dialect 3 client.

When created by a dialect 3 client
EXECUTE PROCEDURE *bignum* (65535, 65535) returns *ERROR* can't access INT64 when executed by a dialect 1 client.

EXECUTE PROCEDURE *bignum* (65535, 65535) returns 4294836225 when executed by a dialect 3 client.

Generators

InterBase 6 and later generators return a 64-bit value, and only wrap around after 2^{64} invocations (assuming an increment of 1), rather than after 2^{32} as in InterBase 5. Applications should use an *ISC_INT64* variable to hold the value returned by a generator. A client using dialect 1 receives only the least significant 32 bits of the updated generator value, but the entire 64-bit value is incremented by the engine even when returning a 32-bit value to a client that uses dialect 1. If your database was using an INTEGER field for holding generator values, you need to recreate the field so that it can hold 64-bit integer values.

Miscellaneous Issues

- IN clauses have a limit of 1500 elements

Resolution If you have more than 1500 elements, place the values in a temporary table and use a SELECT subquery in place of the list elements.

- Arithmetic operations on character fields are no longer permitted in client dialect 3

Resolution Explicitly cast the information before performing arithmetic calculations.

- Using **isql** to select from a TIMESTAMP column displays all information when client dialect is 3.

Resolution In versions of InterBase prior to 6.0, the time portion of a timestamp displayed only if SET TIME ON was in effect. In 6.0 and later client dialect 3, the time portion of the timestamp always displays.

Migrating Servers and Databases

You can migrate your servers and applications to InterBase 6 at and later different times. They are separate migrations. Bear the following issues in mind as you plan your migration:

- Older clients can still access databases that have been migrated to InterBase 6 and later. You must be aware, however, that they cannot access new datatypes or data stored as INT64, and they always handle double quoted material as strings.
- InterBase strongly recommends that you establish a migration testbed to check your migration procedures before migrating production servers and databases. The testbed does not need to be on the same platform as the production clients and servers that you are migrating.

The migration path varies somewhat depending on whether you are replacing an existing server or installing a new server and moving old databases there. Upgrading an existing server costs less in money, but may cost more in time and effort. The server and all the databases you migrate with it are unavailable during the upgrade. If you have hardware available for a new InterBase 6 and later server, the migration can be done in parallel, without interrupting service more than very briefly. This option also offers an easier return path if problems arise with the migration.

“In-place” Server Migration

This section describes the recommended steps for replacing an InterBase 5 server with an InterBase 6 server.

1 Shut down each database before backup to ensure that no transactions are in progress.

2 Back up all databases on the version 5 server. Include *isc4.ib* if you want to preserve your configured user IDs.

As a precaution, you should validate your databases before backing up and then restore each database to ensure that the backup file is valid.

3 Shut down the version 5 server. If your current server is a Superserver, you are not required to uninstall the server if you intend to install over it, although uninstalling is always good practice. You cannot have multiple versions of InterBase on the same machine. If your current server is Classic, you *must* uninstall before installing InterBase 6.

4 Install the version 6 server.

Note The install does not overwrite *isc4.ib* or *isc4.gbk*.

5 Start the new server.

- On UNIX/Linux platforms, issue the following command to start the InterBase Superserver as user “InterBase”:

  ```
  # echo "/usr/InterBase/bin/ibmgr -start -forever" | su InterBase
  ```

Note that InterBase can run only as user “root” or user “InterBase” on UNIX.

6. To restore the list of valid users, follow these steps:
   a. Restore *isc4.gbk* to *isc4_old.ib*
   b. Shut down the server
   c. Copy *isc4_old.ib* over *isc4.gdb*
   d. Copy *isc4_old.gbk* over *isc4.gbk*
   e. Restart the server
7. Delete each older database file. Restore each database from its backup file. This process creates databases in the current version. For databases that were 5.x or older when they were backed up, the dialect is 1. For later databases, the dialect is preserved.
8. Perform a full validation of each database.

After performing these steps, you have an InterBase 6 and later server and InterBase 6 and later, dialect 1 databases. See **About InterBase 6 and Later, Dialect 1 Databases** to understand more about these databases. See **Migrating Databases to Dialect 3** for a description of how to migrate databases to dialect 3. See **Migrating Clients** for an introduction to client migration.

Migrating to a New Server

This section describes the recommended steps for installing InterBase 6 or newer as a new server and then migrating databases from a previous InterBase 5 or older installation. The process differs only slightly from an in-line install.

In the following steps, *older* refers to databases that are on a version 5 or older InterBase server. *Newer* and *new* refer to an InterBase version 6 or newer server.

1. Shut down the older databases before backup to ensure that no transactions are in progress.
2. Back up all databases that are on the older server. Include *isc4.ib* if you want to preserve your configured user IDs.
3. Install the new server.
4. Start the new server.

   - On UNIX/Linux platforms, issue the following command to start the InterBase Superserver as user “InterBase”:

     ```
     # echo "/usr/InterBase/bin/ibmgr -start -forever" | su InterBase
     ```

Note that InterBase can run only as user “root” or user “InterBase” on UNIX.

5 Copy the database backup files to the new server and restore each database from its backup file. This process creates databases that have the current version, ODS, and dialect. (Note: In later versions of InterBase, it creates the appropriate current ODS, but always dialect 1.)

   Save your backup files until your migration to dialect 3 is complete.

6 To restore the list of valid users, follow these steps:

   a Restore *isc4.gbk* to *isc4_old.ib*

   b Shut down the server

   c Copy *isc4_old.ib* over *isc4.gdb*

   d Copy *isc4_old.gbk* over *isc4.gbk*

   e Restart the server

7 Perform a full validation of each database on the new server.

After performing these steps, you have an InterBase 6 and later server and InterBase 6 and later, dialect 1 databases. See **About InterBase 6 and Later, Dialect 1 Databases** to understand more about these databases. See **Migrating Databases to Dialect 3** for a description of how to migrate databases to dialect 3. See **Migrating Clients** for an introduction to client migration.

About InterBase 6 and Later, Dialect 1 Databases

When you back up a version 5 database and restore it in InterBase 6, what do you have?

- A version 5 client can access everything in the database with no further changes.
- If there are object names—column or table names, for instance—that include any of the 17 new keywords, you must change these names in order to access these objects with a version 6 dialect 1 client. The new ALTER COLUMN clause of ALTER TABLE makes it easy to implement column name changes.
  - Version 5 clients can still access the columns.
  - Dialect 3 clients can access these columns as long as they delimit them with double quotes.
- The 17 new keywords are reserved words. However, the new datatypes TIME and DATE are not available to use as datatypes. DATE columns have the old meaning—both date and time. The new meaning of DATE—date only—is available only in dialect 3.
- All columns that were previously DATE datatype are now TIMESTAMP datatype. TIMESTAMP contains exactly the information that DATE did in previous versions.

- Exact numeric columns—those that have a DECIMAL or NUMERIC datatype with precision greater than 9—are still stored as DOUBLE PRECISION datatypes. All arithmetic algorithms that worked before on these columns still work as before. It is not possible to store data as INT64 in dialect 1.

Migrating Databases to Dialect 3

There are four major areas of concern when migrating a database from dialect 1 to dialect 3:

- Double quotes
- The DATE datatype
- Large exact numerics (for purposes of this discussion, NUMERIC and DECIMAL datatypes that have a precision greater than 9)
- Keywords

The process varies somewhat depending on whether you can create an application to move data from your original database to an empty dialect 3 database. If you do not have access to such a utility, you need to perform an in-place migration of the original database.

Overview

In either method, you begin by extracting the metadata from your database, examining it for problem areas, and fixing the problems.

- If you are performing an in-place migration, you copy corrected SQL statements from the metadata file into a new script file, modify them, and run the script against the original database. Then you set the database to dialect 3. There are some final steps to take in the dialect 3 database to store old data as INT64.
- If you have a utility for moving data from the old database to a newly created empty database, you use the modified metadata file to create a new dialect 3 database and use the utility to transfer data from the old database to the new.

In both cases, you must make changes to the new database to accommodate migrated columns that must be stored as INT64 and column constraints and defaults that originally contained double quotes.

The two methods are described below.

Method One: In-place Migration

1 If you have not migrated the database to version 6 and later, dialect 1, do so first. Back up the database again.

2 Extract the metadata from the database using **isql -x**. If you are migrating legacy databases that contain GDML, see **Migrating Older Databases**.

3 Prepare an empty text file to use as a script file. As you fix data structures in the metadata files, you will copy them to this file to create a new script.

Note You could also proceed by removing unchanged SQL statements from the original metadata file, but this is more likely to result in problems from statements that were left in error. Embarcadero recommends creating a new script file that contains only the statements that need to be run against the original database.

For the remaining steps, use a text editor to examine and modify the metadata and script files. Place copied statements into the new script file in the same order they occur in the metadata file to avoid dependency errors.

4 Search for each instance of double quotes in the extracted metadata file. These can occur in triggers, stored procedures, views, domains, table column defaults, and constraints. Change each double quote that delimits a string to a single quote. Make a note of any tables that have column-level constraints or column defaults in double quotes.

Copy each changed statement to your script file, but do not copy ALTER TABLE statements whose only double quotes are in column-level constraints or column defaults.

Important When copying trigger or stored procedure code, be sure to include any associated SET TERM statements.

Quoted quotes If there is any chance that you have single or double quotes *inside* of strings, you must search and replace on a case-by-case basis to avoid inappropriate changes. The handling of quotation marks within strings is as follows:

Table 13.6 Handling quotation marks inside of strings

String:	In "peg" mode
Double-quoted:	"In ""peg"" mode"
Single-quoted:	'In "peg" mode'
String:	O'Reilly
Double-quoted:	"O'Reilly"
Single-quoted:	'O''Reilly'

5 In the new script file, search for occurrences of the TIMESTAMP datatype. In most cases, these were DATE datatypes in your pre-6 database. For each one, decide whether you want it to be TIME, TIMESTAMP, or DATE in your dialect 3 database. Change it as needed.

6. Repeat step 5 in the metadata file. Copy each changed statement to your new script file.
7. In the new script file, search for occurrences of reserved words that are used as object names and enclose them in double quotes; that makes them delimited identifiers.
8. Repeat step 7 in the metadata file. Copy each changed statement to your new script file.
9. In each of the two files, search for each instance of a DECIMAL or NUMERIC datatype with a precision greater than 9. Consider whether or not you want data stored in that column or with that domain to be stored as DOUBLE PRECISION or INT64. See **Do you really need to migrate your NUMERIC and DECIMAL datatypes?** for a discussion of issues. For occurrences that should be stored as DOUBLE PRECISION, change the datatype to that. Leave occurrences that you want stored as INT64 alone for now. Copy each changed statement that occurs in the metadata file to your new script file.

Perform the following steps in your new script file:

10. Locate each CREATE TRIGGER and CREATE DOMAIN statement and change it to ALTER TRIGGER or ALTER DOMAIN as appropriate.
11. Locate each CREATE VIEW statement. Precede it by a corresponding DROP statement. For example, if you have a CREATE VIEW *foo* statement, put a DROP VIEW *foo* statement right before it, so that when you run this script against your database, each view first gets dropped and then re-created.
12. If you have any ALTER TABLE statements that you copied because they contain named table-level constraints, modify the statement so that it does nothing except drop the named constraint and then add the constraint back with the single quotes.
13. Check that stored procedure statements are ALTER PROCEDURE statements. This should already be the case.
14. At the beginning of the script, put a CONNECT statement that connects to the original database that you are migrating.
15. Make sure your database is backed up and run your script against the database.
16. Use **gfix** to change the database dialect to 3.

    gfix -sql_dialect 3 *database.ib*

    Note To run **gfix** against a database, you must attach as either the database owner or SYSDBA.
17. At this point, DECIMAL and NUMERIC columns with a precision greater than 9 are still stored as DOUBLE PRECISION. To store the data as INT64, read **Do you really need to migrate your NUMERIC and DECIMAL datatypes?** and, if necessary, follow the steps in **Migrating NUMERIC and DECIMAL datatypes**.

18 Validate the database using either IBConsole or **gfix**.

That's it. You've got a dialect 3 database. There is a little more work to do if you want your NUMERIC and DECIMAL columns with a precision of greater than 9 to be stored as INT64. At this point, they are still stored as DOUBLE PRECISION. To decide whether you want to change they way data in these columns is stored, read **Do you really need to migrate your NUMERIC and DECIMAL datatypes?** and Migrating NUMERIC and DECIMAL datatypes.

In addition, there are some optional steps you can take that are described in the following sections, **"Column Defaults and Column Constraints"** and **"Unnamed Table Constraints"**.

Important If you ever extract metadata from the dialect 3 database that you created using the steps above, and if you plan to use that metadata to create a new database, check to see if the extracted metadata contains double quotes delimiting string constants in column defaults, column constraints, or unnamed table constraints. Change any such occurrences to single quotes before using the metadata to create the new database.

Column Defaults and Column Constraints

The steps above permitted you to retain double quoted string constants in column defaults, column constraints, and unnamed table constraints. This is possible because, once created, InterBase stores them in binary form.

Following the steps above creates a dialect 3 database that is fully functional, but if it contains double quoted string constants in column defaults, column constraints, or unnamed column constraints, inconsistencies are visible when you SHOW metadata or extract it. You can choose to resolve these inconsistencies by following these steps:

1 Back up the database.

2 Examine the metadata to detect each occurrence of a column default or column constraint that uses double quotes.

3 For each affected column, use the ALTER COLUMN clause of the ALTER TABLE statement to give the column a temporary name. If column position is likely to be an issue with any of your clients, change the position as well.

4 Create a new column with the desired datatype, giving it the original column name and position.

5 Use UPDATE to copy the data from old column to the new column:

```
UPDATE table_name
  SET new_col = old_col;
```

6 Drop the old column.

Unnamed Table Constraints

Read the first two paragraphs under **Column Defaults and Column Constraints** to understand why you don't always need to change constraints with double quotes to single-quoted form, and why you might want to change them.

To bring unnamed table constraints that contain double quotes into compliance with the dialect 3 standard, follow these steps:

1. Back up the database.
2. Examine the metadata to detect each occurrence of an unnamed table constraint that uses double quotes.
3. For each occurrence, use SHOW TABLE to see the name that InterBase has assigned to the constraint.
4. Use ALTER TABLE to drop the old constraint, using the name given in the SHOW TABLE output and add a new constraint. For ease in future handling, give the constraint a name.

   If SHOW TABLE shows that InterBase stores the unnamed constraint as "INTEG_2", then issue the following statement to change the constraint:

```
ALTER TABLE foo
   DROP CONSTRAINT INTEG_2,
   ADD CONSTRAINT new_name
      CHECK (col_name IN ('val1', 'val2', 'val3'));
```

About NUMERIC and DECIMAL Datatypes

If you back up a NUMERIC or DECIMAL column with a precision greater than 9 (for example, NUMERIC(12,2)) in an InterBase 5 or earlier database and restore the database as InterBase 6 and later, the column is still stored as DOUBLE PRECISION. Because InterBase does not allow datatype conversions that could potentially result in data loss, you cannot use the ALTER COLUMN statement to change the column datatype from DOUBLE PRECISION to INT64. To migrate a DOUBLE PRECISION column to an INT64 column, you must create a new INT64 column and copy the contents of the older column into it.

In InterBase 6 and later dialect 3, when you create a NUMERIC or DECIMAL column with a precision of greater than 9, data in it is automatically stored as an INT64 exact numeric.

If you want NUMERIC and DECIMAL datatypes with a precision greater than 9 to be stored as exact numerics, you must take some extra steps after migrating to dialect 3. The following sections tell you how to decide whether you really need to take these steps and how to perform them if you decide you want the exact numerics.

Do you really need to migrate your NUMERIC and DECIMAL datatypes?

As you migrate your databases to dialect 3, consider the following questions about columns defined with NUMERIC and DECIMAL datatypes:

- Is the precision less than 10? If so, there is no issue. You can migrate without taking any action and there will be no change in the database and no effect on clients.
- For NUMERIC and DECIMAL columns with precision greater than 9, is DOUBLE PRECISION an appropriate way to store your data?
  - In many cases, the answer is “yes.” If you want to continue to store your data as DOUBLE PRECISION, change the datatype of the column to DOUBLE PRECISION either before or after migrating your database to dialect 3. This doesn’t change any functionality in dialect 3, but it brings the declaration into line with the storage mode. In a dialect 3 database, newly-created columns of this type are stored as INT64, but migrated columns are still stored as DOUBLE PRECISION. Changing the declaration avoids confusion.
  - DOUBLE PRECISION may not be appropriate or desirable for financial applications and others that are sensitive to rounding errors. In this case, you need to take steps to migrate your column so that it is stored as INT64 in dialect 3. As you make this decision, remember that INT64 does not store the same range as DOUBLE PRECISION. Check whether you will experience data loss and whether this is acceptable.

Migrating NUMERIC and DECIMAL datatypes

Read **Do you really need to migrate your NUMERIC and DECIMAL datatypes?** to decide whether you have columns in a dialect 1 database that would be best stored as 64-bit integers in a dialect 3 database. If this is the case, follow these steps for each column:

1. Migrate your database to InterBase 6 and later as described in **Method One: In-place Migration**.
2. Use the ALTER COLUMN clause of the ALTER DATABASE statement to change the name of each affected column to something different from its original name. If column position is going to be an issue with any of your clients, use ALTER COLUMN to change the positions as well.
3. Create a new column for each one that you are migrating. Use the original column names and if necessary, positions. Declare each one as a DECIMAL or NUMERIC with precision greater than 9.
4. Use UPDATE to copy the data from each old column to its corresponding new column:

   UPDATE *tablename*
   SET *new_col* = *old_col*;

5. Check that your data has been successfully copied to the new columns and drop the old columns.

Method Two: Migrating to a New Database

If you can create a data transfer utility that copies data between databases, the process of migrating a database to dialect 3 is considerably simplified.

Overview Extract the metadata from your database, examine it for problem areas, and fix the problems. Use the modified metadata file to create a new dialect 3 database and use an application to transfer data from the old database to the new.

1. If you have not migrated the database to version 6, dialect 1, do so first. Back up the database again.
2. Extract the metadata from the database using **isql**. If you are migrating a database that contains data structures created with GDML, see **Migrating Older Databases**.

For the following steps, use a text editor to examine and modify the metadata file.

3. Search for each occurrence of the TIMESTAMP datatype. In most cases, these were DATE datatypes in your pre-6 database. Decide whether you want it to be TIME, TIMESTAMP, or DATE in your dialect 3 database. Change it as needed.
4. Find all instances of reserved words that are used as object names and enclose them in double quotes to make them delimited identifiers.
5. Search for each instance of double quotes in the extracted metadata file. These can occur in triggers, stored procedures, views, domains, exceptions, table column defaults, and constraints. Change each double quote to a single quote.
6. Search for each instance of a DECIMAL or NUMERIC datatype with a precision greater than 9. Consider whether or not you want that data stored as DOUBLE PRECISION or INT64. See **Do you really need to migrate your NUMERIC and DECIMAL datatypes?** for a discussion of issues. For occurrences that should be stored as DOUBLE PRECISION, change the datatype to that. Leave occurrences that you want stored as INT64 alone for now.
7. At the beginning of the file, enter SET SQL DIALECT 3. On the next line, uncomment the CREATE DATABASE statement and edit it as necessary to create a new database.
8. Run the metadata file as a script to create a new database.
9. Use your data transfer utility to copy data from the old database to the new dialect 3 database. In the case of a large database, allow significant time for this.
10. Validate the database using **gfix**.
11. At this point, DECIMAL and NUMERIC columns with a precision greater than 9 are still stored as DOUBLE PRECISION. To store the data as INT64, read **Do you really need to migrate your NUMERIC and DECIMAL datatypes?** and, if necessary, follow the steps in **Migrating NUMERIC and DECIMAL datatypes**.

Migrating Older Databases

If you have legacy databases in which some data structures were created with GDML, you may need to extract metadata in a slightly different way.

1. Try extracting metadata as described in Step 2 above and examine it to see if all tables and other DDL structures are present. If they are not, delete the metadata file and extract using the **-a** switch instead of the **-x** switch. This extracts objects created in GDML.
2. You may have to change some of the code to SQL form. For example, the following domain definition

   ```
   CREATE DOMAIN NO_INIT_FLAG AS SMALLINT
      ( no_init_flag = 1 or
      no_init_flag = 0 or
      no_init_flag missing);
   ```

   needs to be translated to:

   ```
   CREATE DOMAIN NO_INIT_FLAG AS SMALLINT
      CHECK ( VALUE = 1 OR VALUE = 0 OR VALUE IS NULL );
   ```

3. Some code may be commented out. For example:

   ```
   CREATE TABLE BOILER_PLATE (BOILER_PLATE_NAME NAME,
       DATE DATE,
      CREATED_DATE COMPUTED BY /*  Date */);
   ```

   needs to be changed to:

   ```
   CREATE TABLE BOILER_PLATE (BOILER_PLATE_NAME NAME,
      "DATE" DATE,
      CREATED_DATE COMPUTED BY "DATE");
   ```

Migrating Clients

To migrate an older client application to InterBase XE, install the InterBase XE client onto the platform where the client application resides. An InterBase server then recognizes that client as a version 6 dialect 1 client.

It is good practice to recompile and relink the application and make note of field names, datatype use, and so on in the new application. When you recompile, state the dialect explicitly:

```
SET SQL DIALECT n;
```

Important If you have databases that use any of the new version 6 keywords as object identifiers and you are not migrating those databases to dialect 3, you might consider not migrating your version 5 clients. If you migrate them to version 6 dialect 1, you lose the ability to access those keyword columns. See **New InterBase Keywords**.

When you recompile an existing **gpre** client, you must recompile it with the **gpre -sql_dialect *n*** switch.

There are several paths that permit you to create dialect 3 clients that access all new InterBase 6 features:

- In Delphi, make calls to functions in the new InterBase Express (IBX) package. Because the Delphi beta includes InterBase 5, it ships with a version of IBX that does not include calls to the new InterBase XE Services, Install, and Licensing APIs.
- To write embedded SQL applications that address all InterBase XE dialect 3 functionality, compile them using **gpre -sql_dialect 3**.

Table 13.7 Migrating Clients: Summary

Client	How to migrate
Older applications such as InterBase version 5 applications	• Dialect is 1; there is no way to change the dialect • A version 5 client application becomes version 6 dialect 1 client whenever the InterBase XE client is installed on the machine with the client
isql	• Issue the command line option: -sql_dialect *n* • Or issue the command SET SQL DIALECT *n*;
GPRE	• Issue the command line option -sql_dialect *n* • Or issue the command EXEC SQL SET SQL DIALECT *n*;
BDE	All applications use SQL dialect 1. To access InterBase dialect 3 features from Delphi, use the IBX components
InterClient	InterBase 6: All applications use SQL dialect 1 InterBase 7 introduced InterClient 3, which is a dialect 3 client
Direct API calls	Set the dialect parameter on *isc_dsql_execute_immediate*(), *isc_dsql_exec_immed2*(), *isc_dsql_prepare*() API calls to the desired dialect value: 1 or 3

Migrating Data from Other DBMS Products

If you have a large amount of data in another DBMS such as Paradox, the most efficient way to bring the data into InterBase is to export the data from the original DBMS into InterBase external file format. (See the ***Data Definition Guide*** for more information about InterBase external files.) Then insert the data from the external files into the internal tables. It is best not to have any constraints on new internal tables; you can validate the database more easily once the data is in InterBase. If constraints do exist, you will need triggers to massage the incoming data.

Appendix

B

InterBase Limits

This appendix 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.

Various InterBase Limits

Table 14.1 InterBase Specifications

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. Applications that engage in high levels of contention or that perform complex or high-volume operations could cause the practical number of clients to be fewer. In applications that don't generate much contention, InterBase can support a large number of users, where "large" is not well-defined.
Maximum database size	**No limit is imposed by InterBase; maximum size is defined by** the operating system. Limit depends on database page size: • 1KB page size: largest DB is 2TB • 2KB page size: largest DB is 4TB • 4KB page size: largest DB is 8TB • 8KB page size: largest DB is 16TB • 16KB page size: largest DB is 32TB
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 cache pages per database	**750,000. Not all database page sizes will be able to** accommodate this limit in a 32-bit address space. When applying a large cache, other considerations must be taken into account such as the number of connections, statements or other database using memory at the same time. A large cache size will depend on whether a 32-bit executable is running on a 64-bit OS **or how a 32-bit OS has been configured.** This number depends on system memory, OS, InterBase version and DB page size: • 750,000 pages for 32-bit InterBase • 75,000,000 pages for 64-bit InterBase

Table 14.1 InterBase Specifications

Characteristic	Value
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.
Maximum number of tables per database	32,640.
Maximum table size	Limit depends on database page size. Note that the total size for all tables cannot exceed maximum database size. • 1KB page size: largest table is 2TB • 2KB page size: largest table is 4TB • 4KB page size: largest table is 8TB • 8KB page size: largest table is 16TB • 16KB page size: largest table is 32TB
Maximum versions per table	255; then no more metadata changes until the database has been backed up and restored.
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**.** Depends on row characteristics and compression, but as many rows as can be stored in maximum table size. The highest row number is 2**38 - 1 (274,877,906,943).
Maximum number of indexes per table	It is now possible to create 255 indexes on a single table in InterBase XE Update 1. A larger DB page size always enables more index definitions than smaller DB page sizes. If your DB page size is not sufficient, you will receive the error “cannot add index, index root page is full.”
Maximum number of indexes per database	**By design, 2^{32}, because you can create 2^{16} tables per database,** and each table can have 2^{16} indexes. Maximum number of tables x Maximum number of indexes per table 32,640 x 64 = 2,089,960

Table 14.1 InterBase Specifications

Characteristic	Value
Maximum index key size	With the ODS 15 databases the maximum index key size limit is increased. Now larger column data can use this for both single-byte character sets and multi-byte (such as UTF8) columns. Because InterBase XE supports UTF8 and multiple other multi-byte character sets, the limit has been increased. For example, a single-column key using 4-byte UTF8 character would calculate to 1020/4 = 254 UTF8 characters with a 4KB page size. • ODS 15 databases automatically allow index definitions where the underlying key size is now a factor of the database page size. An index key can now be up to 4 bytes less than 1/4th the page size. • By default, InterBase databases are created with a 4KB page size. This can be overridden up to 16KB page size by the database developer. • The 4KB page size database would allow indexes that can accommodate 1020 bytes per key. • A 16KB page size can accommodate a 4092 bytes per key and so on. **Caution**: Databases created with engines enabled with this functionality cannot be moved back to older versions of InterBase. Also a database restore to a smaller page size will fail if indexes with large key size cannot fit within the limit specified above. No user interface or actions are required by the user to enable this functionality. Each time a database restore is performed, the indices are recreated. Only ODS 15 and later databases have support for larger index keys. If you want to use this facility, restore your database to ODS 15. Other indices that use a smaller size than 252 bytes continue to have the same on-disk storage without any penalty. Note that multibyte character sets must fit within the key by counting bytes, not by counting characters. It is good practice to keep index keys as small as possible. This limits the depth of indexes and increases their efficiency.
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 14.1 InterBase Specifications

Characteristic	Value
Maximum Blob size	The size of the largest single Blob depends on the database page size: 1KB page size: largest Blob is 64MB 2KB page size: largest Blob is 512MB 4KB page size: largest Blob is 4GB 8KB page size: largest Blob is 32GB 16KB page size: largest Blob is 256GB A Blob is a stream of many segments. The maximum Blob segment size is 64KB.
Maximum tables in a JOIN	No restriction by design, but the task of joining tables is exponential in relation to 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	16
Levels of nesting per stored procedure or trigger	• 750 on Windows platforms • 1000 for UNIX platforms
Maximum size of key in SORT clause	32 KB.
Maximum size of external table file	64-bit file offset allows up to 16 Exabytes of information.
Range of date values	January 1, 100 a.d. to February 29, 32768 a.d.
Transaction Limits	• Databases with ODS <=15 need to be backed up and restored before they hit the 2 Billion transaction ID limit • Databases with ODS >=16 can continue to be online beyond 2 billion transactions since they support 64bit Transaction ID. A benefit of this is you can have your databases online to service your applications more closer to a 24/7 scenario without having to backup/ restore due to this earlier 32bit limit.

Index

A

accessing databases 6-2 to ??
activating shadows 7-18
adding
  comments in ISQL script files 11-59
  database file 7-5
  shadow files 7-19
  users 6-13, 6-17 to 6-19
ADMIN_DB 6-3
Altering a Journal Archive 9-12
application development
  InterBase Express 12-22
Archive Sweeping 9-13
archiving journal files
  commands used 9-12
  how often 9-12
assigning passwords 6-14, 6-16, 6-17
attachments *See* connections
AUTO mode 7-17
automatic commit of DDL statements 11-15, 11-23
average
  data length 10-11
  fill 10-7

B

backing up databases
  converting external files to internal tables 8-6, 8-29
  metadata only 8-6, 8-28
  options 8-27
  preventing sweeping 8-28
backup and restore options
  about 8-1
backup and restore tools 8-2
backup options
  about 8-1
backups
  incremental, about 8-7
  multi-file, about 8-5
  single-file
    about 8-5
  types of 8-2
  using gbak 8-4
batch functions
  updating 11-12
batch updates
  using 11-12
binary
  data 11-31
  files 11-31
blindmeta.sql 6-9
BLOB data
  editing 11-30
  ID numbers, retrieval 11-31
  improving access time 8-34
  saving 11-30
  segment size 12-17
BLOB filters 11-49
BLOBDUMP 11-30
buffered writes vs. forced writes 7-28

C

cache buffers, number of 7-26, 10-10
changing
  character set 11-9
  database page size 8-16, 8-34
  gsec entries 6-19
  user names 6-19
character set, changing 11-9, 11-15
CHECK constraints 11-46, 11-48
Checkpoint Interval 9-7
Checkpoint Length 9-6
checkpoints
  about 9-2
  determining intervals 9-9
checksums
  header page 10-9
  ignoring 8-6, 8-29
client dialect 11-3, 11-26
column headers, changing page length 11-24
commands
  gfix 7-38 to 7-41
  gsec 6-16 to 6-19
  isql 11-28 to 11-29
    displaying 11-32, 11-34
    editing 11-31
    executing 11-8, 11-10, 11-33
comments in ISQL scripts 11-59
COMMIT 11-27, 11-59
commit
  after each table 8-16, 8-35
  automatic 11-15, 11-23
conditional shadows 7-18
configuration parameters 4-20
CONNECT 11-25
connecting to databases 5-12 to 5-15
  denying new connections 7-33
  troubleshooting 5-28 to 5-32
connections
  databases 11-25

denying 7-33
monitoring 10-1 to 10-4
remote servers 5-34, 11-24, 11-25
constraints 11-46, 11-48
continuation prompt 11-23
corrupt databases
backing up by ignoring checksums 8-6, 8-29
repairing 7-31
CREATE DATABASE 11-23, 11-25
CREATE JOURNAL ARCHIVE command
about 9-11
CREATE JOURNAL command
options 9-6
using 9-6
creating
conditional shadows 7-18
databases 7-9
multifile databases 7-5
shadows 7-15 to 7-18, 8-35
creation date, database 10-10

D

data page slots 10-7
data pages 10-7
database administration 6-13
adding users 6-17
changing users 6-14, 6-19
overview 1-8
security utility 6-16 to 6-19
database backups
about 8-1
incremental 8-7
multi-file, about 8-5
single-file, about 8-5
types of 8-2
using IBConsole 8-25
database dialect 11-24
database restores
how-to 8-14, 8-17
using IBConsole 8-25
database statistics
gstat 10-13
IBConsole 10-5
databases
accessing 6-2 to ??
backing up 8-26
closing 5-14, 11-32, 11-35
connecting to 5-12 to 5-15
creating 7-9
creating multifile 7-5
creation date 10-10
deactivating indexes 8-35
deleting 7-12
denying new connections 7-33
disabling validity checking 8-16, 8-36
dropping 7-12
extracting metadata 8-6, 8-28
file naming conventions 7-4
file, adding 7-5
gbak and gsplit 1-10
locking statistics 10-16
name information 10-9
naming 6-3, 7-4
overwriting 8-16, 8-34
page size 8-16, 8-34, 10-9
read-only 7-9, 7-39, 8-16
repairing 7-31 to 7-32
replacing 8-16, 8-34
restarting 7-32, 7-35
security 6-12
shadowing 7-13 to 7-19, 8-35
shutdown and restart 7-32
shutting down 7-32
structure, retrieving 11-47
sweeping 7-22 to 7-24, 7-39
immediately 7-25, 7-40
sweep interval 7-24
testing connections 5-34
unregistering 5-14
upgrading the on-disk structure 8-3
validating 7-28 to 7-32, 7-41
viewing information 11-47
viewing metadata 11-18
DDL
automatic commits 11-15, 11-23
defined 11-28
extracting 11-23, 11-24, 11-27 to 11-28
processing 11-22
transactions 11-27
deactivating indexes 8-15, 8-35
deadlocks 10-16
DECLARE FILTER 11-22, 11-28, 11-49
default terminator 11-23, 11-36
deleting
databases 7-12
shadows 7-19
users 6-15
dialect 11-26
database 11-24
disabling
garbage collection 8-28
validity checking 8-16, 8-36
disabling automatic internet dialup 5-31
Disabling Journal Files 9-10
disaster recovery
about 9-1
commands used 9-13

using journal files and journal archives for 9-13
displaying
error messages 11-34
metadata 11-18, 11-29
objects 11-52
query plan 11-15, 11-43
statements 11-23
statistics 11-44
version numbers 11-24
DML 11-28
processing 11-22
domains 11-48
DROP JOURNAL ARCHIVE command 9-12
DROP JOURNAL command
about 9-10
DROP SHADOW 7-19
dropping
databases 7-12
shadows 7-19
users 6-15
Dropping a Journal Archive 9-12

E

echoing 11-23
EDIT 11-31
INPUT and 11-33
editing
BLOB data 11-30
input files 11-46
isql commands 11-31
editors 11-31
embedded user authentication 8-15
restoring 8-14
embedded user authentication (EUA)
about 6-6
adding and modifying users 6-8
enabling 6-6
encrypting network communication 5-15
examples 5-24
setting up 5-16
environments, isql 11-29
errors
connection refused 5-28, 5-32
connection rejected 5-30
gbak 8-21
gfix 7-41
gsec 6-20
isql 11-29
exceptions 11-48
executing SQL scripts
with IBConsole 11-10
EXIT 11-32
QUIT vs. 11-35
external tables 7-4
EXTERNAL_FILE_DIRECTORY 7-4
extracting
DDL 11-23, 11-24
metadata 8-28, 11-21, 11-27 to 11-28

F

files
input 11-10, 11-23
naming 6-3, 7-4
naming conventions 7-4
shadow 7-16, 7-19, 8-35
writing to 11-13, 11-24, 11-27
fill distribution 10-7, 10-11
filters, displaying 11-19, 11-49
forced writes vs. buffered writes 7-28

G

garbage collection, disabling 8-28
gbak command
about 8-4
error messages 8-20
performing backups and restores using 8-4
generators, displaying 11-19, 11-50
gfix
activating shadows 7-18
commands 7-38 to 7-41
errors 7-41
killing shadows 7-18
global header information 10-16
gsec 6-16 to 6-19
adding users 6-17 to 6-19
changing entries 6-19
commands 6-16 to 6-19
deleting entries 6-19
errors 6-20
exiting 6-16
help 6-17
options 6-18
running remotely 6-16
starting 6-16, 6-19
gsplit 1-10
gstat 10-13
Guardian 4-1, 4-10

H

header page generation 10-9
help
gsec 6-17
IBConsole 3-3
InterBase 3-3
ISQL commands 11-32

UNIX 4-12

I

ibconfig 4-19, 4-20
IBConsole
  activating journaling 9-10
  character sets 11-15
  commit and rollback 11-12
  denying new connections 7-33
  displaying metadata 11-18
  executing SQL statements 11-8 to 11-13
  extracting metadata 11-21
  performing backups and restores using 8-25
  restarting databases 7-35
  saving ISQL input and output 11-13
  script file, adding comments 11-59
  security 6-12
  toolbar 3-4
  Tree pane 3-5
  viewing statistics 10-5
  Work pane 3-7
ibguard 4-10
ignoring
  checksums 8-6, 8-29
  limbo transactions 8-6, 8-28
implementation ID 10-10
incremental backups
  about 8-7
index root page 10-7
indexes
  correcting duplicate values 8-35
  deactivating 8-15, 8-35
  depth 10-11
  displaying 11-29, 11-51
  improving performance 8-34, 12-18
  retrieving 11-51
INPUT 11-33
input files 11-10, 11-23, 11-33
  editing 11-46
interactive SQL. See isql
InterBase
  API, user administration 6-15
  developing database applications with IBX 12-22
  Guardian 4-1, 4-10
  on UNIX 4-11
  version numbers 11-24, 11-56
ISC4.GDB 6-16
  viewing contents 6-17, 6-18, 6-19, 8-19
isql 11-22 to 11-30
  commands 11-28 to 11-29
  connecting to databases 11-25
  displaying help 11-32
  editing 11-31
  errors 11-29
  executing 11-8, 11-10, 11-33
  exiting 11-25, 11-29, 11-32, 11-35
  options 11-22 to 11-25
  output 11-34
  saving input and output 11-13
  script files, adding comments in IBConsole 11-59
  SET statements 11-35 to 11-46
  setting environment 11-29
  setting SQL dialect 11-26
  specifying database 11-22
  starting 11-22 to 11-23, 11-25
  terminator characters 11-23, 11-24
isql scripts *See* SQL scripts

J

journal archives
  about 9-1
journal archiving
  about 9-1
  disabling 9-12
  implementing 9-11
  managing archive size 9-13
  overview of how-to 9-5
  subsequent dumps 9-12
  using 9-11
journal archiving commands
  about 9-11
journal files
  about 9-1
  about preallocating journal space 9-8
  checkpoint intervals, tips 9-9
  checkpoints, about 9-2
  creating 9-6
  disabling 9-10
  rollover frequency, tips 9-8
journaling
  about 9-1
  activating 9-6
  best practices 9-15
  enabling 9-6
  examples 9-15
  using IBConsole to activate 9-10

K

killing shadows 7-18

L

leaf buckets 10-11
License Manager

about 2-5
using 2-5
licensing
about 2-1
options 2-1
using the License Manager 2-5
limbo transactions
ignoring 8-6, 8-28
two-phase commit 7-35
lock
manager 10-16
table 10-16
locks 10-16
logging in to a server 5-9
logical backups
about 8-2

M

maintaining archive dumps 9-12
Managing Archive Size 9-13
managing archive size
sequence numbers 9-13
sweeping 9-13
MANUAL mode 7-18
max dup 10-11
metadata
command-line ISQL 11-22
displaying 11-18, 11-29
extracting 8-28, 11-21, 11-27 to 11-28
in IBConsole 11-18
monitoring attachments 10-1 to 10-4
multifile databases 7-5
multiprocessor support 4-3, 4-8

N

name information 10-9
naming
databases 6-3, 7-4
nesting INPUT commands 11-33
network communication
encrypting 5-15
network configuration
encrypting 5-15
next
connection ID 10-9
header page 10-10
transaction 10-9
node names, shared disks vs. 11-26
nodes 10-11
number of cache buffers 10-10

O

objects
deleting 11-52
displaying 11-52
ODS *See* on-disk structure
oldest
active transaction 10-9
transaction 10-9
on-disk structure
upgrading 8-3
version 10-9
operating system shells 11-46
OUTPUT 11-34
output
files 11-13, 11-24, 11-27
isql 11-13, 11-34
metadata 11-18, 11-29
redirecting 11-13, 11-34
statements 11-23
user-defined functions 11-19, 11-49
verbose 8-7, 8-29, 8-36
over-the-wire encryption 5-15
examples 5-24
setting up 5-16

P

Page Cache 9-7
Page Size 9-7
page size
changing 8-16, 8-34
default 7-11
displaying current 10-9
passwords
assigning 6-14, 6-16, 6-17
connecting to remote servers 11-24
physical backups
about 8-2
PLAN 12-22
PLAN option 11-15, 11-43
platforms, server 1-5
primary pointer page 10-7
procedures, listing 11-21, 11-29, 11-52

Q

query 11-22
displaying plan 11-15, 11-43
testing 12-24
QUIT 11-35
EXIT vs. 11-32

R

RBD$ARCHIVE_NAMW 9-14
RDB$ARCHIVE_LENGTH 9-14
RDB$ARCHIVE_SEQUENCE 9-14
RDB$ARCHIVE_TIMESTAMP 9-14
RDB$ARCHIVE_TYPE 9-14
RDB$DEPENDED_ON_SEQUENCE 9-14
RDB$DEPENDED_ON_TIMESTAMP 9-14
readmeta.sql 6-9
read-only databases 7-9, 7-39, 8-16
recovering a database
  using journal files and journal archiving 9-13
remote servers, connecting to 5-34, 11-24, 11-25
repairing databases 7-31 to 7-32
replacing databases 8-16, 8-34
restarting databases 7-32, 7-35
restore options 8-34
restoring a database
  commands 8-14, 8-17
ROLLBACK 11-27
running a SQL script *See* SQL scripts 11-10

S

saving ISQL input and output 11-13
scripts *See* SQL scripts
security
  adding a user 6-13, 6-17
  database 6-12
  displaying privileges 11-51
  dropping users 6-15, 6-19
  IBConsole 6-12
  modifying user configuration 6-14, 6-19
security database 6-1
  name 6-3
sequence number 10-9
Serial Number 2-1
server platforms 1-5
servers
  log 4-28
  logging in 5-9
  starting on UNIX 4-12
  unregistering 5-10
SET 11-35
SET AUTODDL 11-37
SET BLOBDISPLAY 11-38
SET COUNT 11-39 to 11-40
SET ECHO 11-40
SET LIST 11-41
SET NAMES 11-42
SET PLAN 11-43 to 11-44
SET statements 11-35 to 11-46
SET STATS 11-44
SET TERMINATOR 11-23
SET TIME 11-45
SET TRANSACTION 11-27
shadow count 10-10
shadows
  activating 7-18
  adding files 7-19
  advantages 7-14
  AUTO mode 7-17
  conditional 7-18
  creating 7-15 to 7-18, 8-35
  dropping 7-19
  killing 7-18
  limitations 7-14
  MANUAL mode 7-18
  overview 7-13 to 7-15
shared disks, node names vs. 11-26
SHELL 11-46
SHOW CHECK 11-46
SHOW DATABASE 7-16, 11-47
SHOW DOMAINS 11-48
SHOW EXCEPTIONS 11-48
SHOW FILTERS 11-49
SHOW FUNCTIONS 11-49
SHOW GENERATORS 11-50
SHOW GRANT 11-51
SHOW INDEX 11-51
SHOW PROCEDURES 11-52
SHOW SYSTEM 11-54
SHOW TABLES 11-54
SHOW TRIGGERS 11-55
SHOW VERSION 11-56
SHOW VIEWS 11-56
shutting down databases
  denying new connections 7-33
  denying new transactions 7-34
  forced shutdown after timeout 7-34, 7-39
  timeout options 7-33
SMP support 4-3, 4-8
SQL dialect 7-11, 11-24, 11-26
SQL scripts
  committing work 11-59
  creating 11-57
  executing with IBConsole 11-10
  running 11-58
  running with isql 11-33
SQL statements
  committing in IBConsole 11-12
  executing in IBConsole 11-8 to 11-13
  rolling back in IBConsole 11-12
SQLCODE 11-29
starting
  gsec 6-16, 6-19
  InterBase Guardian 4-10

InterBase Server 4-10
isql 11-22 to 11-23, 11-25
statements
displaying 11-23
terminator characters 11-23, 11-24
statistics
displaying 11-44
gstat 10-13
IBConsole 10-5
SuperServer architecture 1-10
sweeping databases 7-22 to 7-24, 7-39
disabling 7-24
immediately 7-25, 7-40
preventing during a backup 8-28
sweep interval 7-24
SYSDBA 6-2
system
editors 11-31
shells 11-46
tables 11-54
views 11-54
system table security 6-9
system temporary tables 10-1 to 10-4

T

tables
constraints 11-46, 11-48
listing 11-19, 11-29
TCP/IP 11-25
terminator characters
default 11-23
isql 11-23, 11-24
testing
database connection 5-34
text
editors 11-31
saving blocks to file 11-31
Timestamp Name 9-7
Timestamps
on backup files 8-11
toolbar 3-4
total dup 10-11
transactions
committing 11-27, 11-32
DDL 11-27
denying new transactions 7-34
isql 11-27
oldest 10-9
oldest active 10-9
rolling back 11-27, 11-35
rolling back limbo transactions 8-28
two-phase commit 7-35
Tree pane 3-5
triggers, listing 11-29, 11-55
troubleshooting, database connection 5-28 to 5-32
trusted host 6-2
two-phase commit 7-35, 8-28

U

unregistering
databases 5-14
servers 5-10
use all space 8-36
user administration with the InterBase API 6-15
user names 11-24
adding 6-16, 6-17 to 6-19
changing 6-19
dropping 6-15
UNIX 6-2
user-defined functions
listing 11-19, 11-49
viewing 11-49

V

validating databases 7-28 to 7-32, 7-41
validity checking, disabling 8-36
verbose output 8-7, 8-29, 8-36
version
numbers, displaying 11-24, 11-56
on-disk structure 10-9
views, listing 11-19, 11-56

W

warnings 11-24
Windows OS commands 11-46
Windows server platforms 1-5
Work pane 3-7
write-ahead logging 9-1
writemeta.sql 6-9