

Release Notes

InterBase 5.6.1

January 2001



Borland

100 Enterprise Way, Scotts Valley, CA 95066 <http://www.borland.com>

Borland may have patents and/or pending patent applications covering subject matter in this document. The furnishing of this document does not convey any license to these patents.

Copyright 2001 Borland. All rights reserved. All InterBase products are trademarks or registered trademarks of Borland. All Borland products are trademarks or registered trademarks of Borland Corporation. Other brand and product names are trademarks or registered trademarks of their respective holders.

Table of Contents

InterBase 5.6.1 Release Notes	
About this document	5
Contacting Borland Corporation	6
General information	6
The InterBase install directory	6
Installation instructions	7
Documentation in PDF form.	7
System requirements	8
New InterBase 5.6 feature for Netware	9
InterBase 5.5 features	11
Greatly enhanced stability and more efficient memory use	11
New ODBC driver	11
InterClient 1.5	11
Transparent upgrade.	11
New On-Disk Structure (ODS)	12
Changed UDF functionality and improved documentation	12
Specify database cache size on restore	13
Ownership retained on database restore	14
New error messages	14
Earlier versions: InterBase 5.0 and 5.1 new features	14
Migration issues	18
Comparing Classic and SuperServer	18
What's available	18
Why two implementations?	19
Lock management	19
Resource use	19
Threaded server and UDFs	19
The InterBase SuperServer	20
The InterBase SuperServer architecture	20
Invoking SuperServer	21
Threads and tasks	21
Security.	21
Managing the SuperServer	21
The InterBase Classic server	22
Invoking the Classic server	22
Lock management	22
Resource use	22
Local access method	23
Monitoring database connections	23
Security.	23
Installing and configuring Classic	23
Shutting down Classic.	24
Troubleshooting the Classic server	25
Limit on connections to inetd on Linux	25
Compiling and linking	26
Applications	26
User-defined function libraries	27
Java application development	28
Appendix A Bug Lists	
Known bugs with workarounds	29
Bug #3297	29
Bug #3446	30
Bug #7520	30
Bug #8251	30
Bug #8412	30
Bug #8429	31

Bug #8542	31
Bug #8591	31
Bug #8600	31
Bug #8813	32
Bug #10069	32
Bug #10072	33
Bug #10098	34
Connection error: “REMOTE INTERFACE not licensed”	34
Installation error: “Internal error near IBcheck”	35
Bugs fixed for InterBase 5.0	35
Bugs fixed for InterBase 5.1.1	39
Bugs fixed for InterBase 5.5	40
Bugs fixed for InterBase 5.6	43
Bugs fixed for InterBase 5.6.1	47

InterBase 5.6.1

Release Notes

About this document

These Release Notes contain:

- How to contact Borland Corporation
- General information
 - The install directory
 - System requirements
 - Documentation and Adobe Acrobat Reader
- New InterBase 5.5 and 5.6 features
- InterBase 5.0 and 5.1.1 features
- Migration issues
- Comparison of Classic server and SuperServer
- The InterBase Classic architecture. This applies only to those who are running InterBase on Linux or SCO.
- Compiling your InterBase 5 with workarounds
- Appendix A lists bugs fixed in InterBase 5.0 through 5.6.1

Note InterBase 5.6.1 is a maintenance release. There is no new functionality added to InterBase with the 5.6 and 5.6.1 releases: the changes consist entirely of bug fixes. For that reason, you will find that all discussion of installation and migration, features, and documentation in this document references the 5.5 release.

Contacting Borland Corporation

► *Mailing address*

Borland
100 Enterprise Way
Scotts Valley, CA 95066-3249
Phone: 1-831-431-1000

► *World-wide web sites*

- Borland maintains an Internet site on the world-wide web for general InterBase information. The URL of this site is:
<http://www.borland.com/interbase>
- Technical information, such as white papers and FAQs, can be found at:
<http://www.community.borland.com/interbase>
- For installation and presales questions, visit:
<http://www.borland.com/devsupport>
- To discuss issues, such as features and potential bugs, with other InterBase users, visit:
<http://www.borland.com/newsgroups>

► *Email addresses*

- For questions about product information, product release schedule, features, feature requests, and VAR partnerships, send email to:
interbase@borland.com
This email address is also helpful if you are a non-U.S. customer and you need to learn the best source of technical support or sales assistance in your region.
- For issues pertaining to content and presentation on our web site, send email to:
webmaster@borland.com
- For information about how to contact InterBase representatives outside the U.S. and Canada, look at the following web page:
<http://www.borland.com/bww/>

General information

This section describes document options for InterBase 5 and lists system requirements for each platform.

The InterBase install directory

Throughout this document, *interbase_home* refers to the InterBase install directory. By default, this is:

- For all MS Windows™ platforms: *C:\Program Files\Borland\InterBase*
- For all UNIX platforms: */usr/interbase*
- For Netware: *SYS:interbas*

You can use the INTERBASE environment variable to change this location. See the *Operations Guide*, Chapter 4, “Server Configuration” for more information on environment variables.

Installation instructions

Complete installation instructions for Windows and Solaris are in Chapter 3 of the *Operations Guide*. Linux users should read *Install.txt* in the */linux* directory on their CDROM and Netware users should read *Install.txt* in the */Intrbase* directory of their CDROM. In the latter two cases, note that the */Client32* directory contains exclusively files that relate to a Windows client install. The Release Notes and *Install.txt* in the */Client32* directory are for Windows.

Installation requires approximately 36MB of disk space for a full install that includes InterBase, InterClient, Adobe Acrobat Reader, and the full document set. Only 11MB is needed to install the InterBase product without InterClient, the documents, or Acrobat Reader.





Documentation in PDF form


InterBase provides all five books in the document set plus the Release Notes in PDF format. You have the option to install the complete document set in PDF form on your hard drive when you are installing the InterBase 5.5 product. This requires about 15MB of disk space. In addition, the document set and Release Notes are available on the CDROM in uncompressed PDF form in the */doc* directory. You can read them directly from that location if you don't want to install them on your system.

You need Adobe Acrobat Reader With Search to view and search these documents. See “[Installing Acrobat Reader With Search](#)” on page 8 of this document for how to acquire and install Acrobat Reader.

► Full-text searching

The five-book document set has been indexed for full-text searching. If you are viewing the documents using Acrobat Reader With Search, you can enter a query and receive a list of hits from all five books in the InterBase document set.

To use full-text searching, click the  button and search for a word or phrase. Acrobat Reader returns a list of books that contain the phrase. Choose the book you want to start looking in to display the first instance. You then use the  and  buttons to step forward and back through instances of your search target. Reader moves from one book to the next. To go to a different book at will, click the  button to display the “found” list.


Note that full-text searching is not the same as Find () , which searches only the current document.

Note The Linux version of Acrobat Reader is not available with full-text searching. Acrobat Reader With Search is included on the CDROM for use with the Windows client.

► *Links*

The PDF documentation set contains many hypertext links that take you to referenced points in the document with a single click. In addition, the Table of Contents and Index entries are hypertext links and therefore are clickable. Throughout the document set, clickable links appear **bold and green**.

► *Navigating in Acrobat Reader*

Acrobat Reader can display the document with a *bookmarks* window pane on the left. The bookmarks are a clickable list of the contents of the book. To display the bookmarks, click the second button on the toolbar () , press w-7, or use the menu item “Bookmarks and Page” under the View menu.

Reader also provides a rich array of navigation aids. Check the View menu for options, including keyboard shortcuts.

► *Installing Acrobat Reader With Search*

FROM THE INTERBASE CDROM

On Windows, you can install Acrobat Reader 4 With Search by choosing Install Adobe Acrobat Reader 4 from the Setup Launcher. Windows, UNIX, and Netware users can also install it directly by running the executable file in the */Adobe* directory of the InterBase CDROM.

For Linux users, the Acrobat version that installs on Linux does not have full-text search. Go to the Adobe subdirectory of the Linux directory, read the *readme* file, and execute *install*.

FROM THE ADOBE WEBSITE

If you don't have the InterBase CDROM handy, you can get Acrobat Reader for free from the Adobe website. It's at <http://www.adobe.com/prodindex/acrobat/readstep.html>. Be sure to download Acrobat Reader With Search, not the plain Acrobat Reader.

- Acrobat® Reader copyright © 1987–2001 Adobe Systems Incorporated. All rights reserved. Adobe and Acrobat are trademarks of Adobe Systems Incorporated.

System requirements

Microsoft Windows NT, Windows 95, and Windows 98

Operating system: Windows NT 4.0 with Service Pack 4 (Microsoft re-released this service pack. The newer version, a 72.8MB file called *SP4 Full Download*, is recommended),

Windows 95 with the Y2K patch, or Windows 98.

Memory: 16 megabytes minimum; 64 or more recommended

Processor/Hardware model: 486 minimum; Pentium II recommended

C/C++ compilers: Microsoft Visual C++ 4.2, Borland C++ 5.0, or C++ Builder 4.0

Linux

Operating system: Red Hat Linux 6.0 or SuSE Linux 6.2

Memory: 16 megabytes minimum; 64 or more recommended

Processor/Hardware model: 486 minimum; Pentium II recommended

C/C++ compilers: GNU C/C++ compiler and debugger

NetWare

Operating system: Novell Netware 4.2 or 5.0

Memory: 16 megabytes minimum; 64 or more recommended

Processor/Hardware model: Pentium II recommended

New InterBase 5.6 feature for Netware

InterBase distributes a library of UDFs. Since the Netware operating system does not support UDFs, these functions have not been available to Netware users in the past. In the InterBase 5.6 release for Netware, InterBase has statically linked the UDF library to the server, making these function available to Netware users for the first time.

Before you can use any of these external functions against a database, you must declare them to the database. To do this, follow these steps:

1. Edit the *ib_udf.sql* script in the *interbase_home\Examples\Api* directory to add a CONNECT statement at the top so that the script connects to the database where you are declaring the functions.
2. Use isql or IBConsole to run the *ib_udf.sql* script. This script declares each of the external functions to the database. You need to declare these functions only once per database.

Note There is a file called *udf.sql* in the *Examples\Api* directory. InterBase recommends that you not declare these additional functions in any environment where more than one user could be attached at the same time, since these functions are not thread safe.

Below is a list of the functions that are supported as built in functions in the InterBase V5.6 NLM server.

Function name	Description	Inputs	Outputs
ABS()	Absolute value	Double precision	Double precision
ACOS()	Arc cosine	Double precision	Double precision
ASCII_CHAR())	Return character based on ASCII code	Integer	Char(1)
ASCII_VAL()	Return ASCII code for given character	Char(1)	Integer
ASIN()	Arc sine	Double precision	Double precision
ATAN()	Arc tangent	Double precision	Double precision
ATAN2()	Arc tangent divided by second argument	Double precision, Double precision	Double precision
BIN_AND()	Bitwise AND operation	Integer	Integer
BIN_OR()	Bitwise OR operation	Integer	Integer
BIN_XOR()	Bitwise XOR operation	Integer	Integer

Function name	Description	Inputs	Outputs
CEILING()	Round up to nearest whole value	Double precision	Double precision
COS()	Cosine	Double precision	Double precision
COSH()	Hyperbolic cosine	Double precision	Double precision
COT()	Cotangent	Double precision	Double precision
DIV()	Integer division	Integer	Integer
FLOOR()	Round down to nearest whole value	Double precision	Double precision
LN()	Natural logarithm	Double precision	Double precision
LOG()	Logarithm of the first argument, by the base of the second argument	Double precision, Double precision	Double precision
LOG10()	Logarithm base 10	Double precision	Double precision
LOWER()	Reduce all uppercase characters to lowercase	Cstring(80)	Cstring(80)
LTRIM()	Strip preceding blanks	Cstring(80)	Cstring(80)
MOD()	Modulus operation between the two arguments	Integer, Integer	Integer
PI()	Return the value of π	—	Double precision
RAND()	Return a random value	—	Double precision
RTRIM()	Strip trailing blanks	Cstring(80)	Cstring(80)
SIGN()	Return -1, 0, or 1	Double precision	Integer
SIN()	Sine	Double precision	Double precision
SINH()	Hyperbolic sine	Double precision	Double precision
SQRT()	Square root	Double precision	Double precision
STRLEN()	Length of string	Cstring(32767)	Integer
SUBSTR()	Substring, starting at position equal to second argument, with length equal to third argument	Cstring(80), Smallint, Smallint	Cstring(80)
TAN()	Tangent	Double precision	Double precision
TANH()	Hyperbolic tangent	Double precision	Double precision

InterBase 5.5 features

InterBase 5.6 does not introduce new features other than the external functions for Netware. This section lists features introduced in the 5.5 version of InterBase. The following section reviews features that were introduced in InterBase 5.0 and 5.1.

Greatly enhanced stability and more efficient memory use

To respond to the priority requests of customers, the InterBase 5.5 release has great improvements in performance, stability, and reliability. The InterBase engineers have fixed many defects and have generally addressed all issues that affect stability in a multiuser environment.

- Memory use is significantly more efficient than in previous releases
- It is safe to update metadata in a database that is in use
- New error messages help database developers identify faults in their UDF code
- InterBase can serve a greater number of concurrent clients without loss of stability or reasonable performance

New ODBC driver

The INTERSOLV[®] DataDirect[™] 3.11.01 ODBC driver for InterBase implements the Microsoft ODBC 3.0 standard and is faster and more reliable than the Visigenic ODBC driver that shipped with earlier versions of InterBase. It supports multithreading, and the use of character sets.

Note The INTERSOLV ODBC driver supports **SQL Roles** as a DSN property and as a connection string option. This is a recent change to the driver, made after the print date for the InterBase 5.5 manuals. The manuals incorrectly state that the ODBC driver does not support SQL Roles. See the ODBC help file, *IBINT13.HLP*, for details on support for SQL Roles.

InterClient 1.5

InterBase 5.5 ships with InterClient 1.5. This latest release of InterClient runs at up to 30 times the speed of InterClient 1.0 and is more reliable. In addition, it supports 23 international character sets.

Transparent upgrade

The version 5.5 release of InterBase does not require any changes to client applications.

New On-Disk Structure (ODS)

The On-Disk Structure (ODS) for InterBase 5.5 has been updated to version 9.1, which supports cascading referential integrity, index garbage collection, and SQL roles. See the section on Migration in the *Operations Guide*, Chapter 1, “Introduction” for more information and for how to make existing tables compatible with this new ODS.

Changed UDF functionality and improved documentation

InterBase’s multithreaded server architecture requires some care in the way memory is allocated and released in user-defined functions (UDFs) and in the way these UDFs are declared. In the new single-process, multithreaded architecture, memory allocated dynamically is not released, since the process does not end. In addition, users running UDFs concurrently use the same static memory space, with predictably disastrous results. InterBase’s new `FREE_IT` keyword allows InterBase users to write thread-safe UDF functions without memory leaks.

Documentation of these changes has been greatly expanded in the InterBase 5.5 document set. For information about writing and declaring UDFs and about handling memory for them, refer to the entries for `DECLARE EXTERNAL FUNCTION` in Chapter 2 and Chapter 5 in the *Language Reference*, to Chapter 11 of the *Data Definition Guide*, and to Chapter 10 of the *Programmer’s Guide*.

Correction Previous versions of InterBase manuals instructed Delphi programmers to use `sysalloc()` to allocate memory. This is an error. Programmers writing UDFs for InterBase 5.5 should use `ib_util_malloc()`. See “Handling memory for return values” on page 219 of the *Programmer’s Guide* for more information.

InterBase 5.5 includes a mechanism for detecting exceptions in UDFs. If your UDF has a defect that raises a runtime exception, the `ibserver` process returns an ISC error code. You don’t need to change your UDF programming methods to take advantage of this feature. The error codes are returned to the client application; use `isc_interprete()` to retrieve the text of the error message based on the error code. The error message text is also written to the *interbase.log*.

The list of exception error codes and associated messages is listed below:

Error token	Error code	Error message
<i>isc_exception_access_violation</i>	335544768L	Access violation. The code attempted to access a virtual address without privilege to do so.
<i>isc_exception_datatype_misalignment</i>	335544769L	Datatype misalignment. The attempted to read or write a value that was not stored on a memory boundary.
<i>isc_exception_array_bounds_exceeded</i>	335544770L	Array bounds exceeded. The code tried to access an array element that is out of bounds.
<i>isc_exception_float_denormal_operand</i>	335544771L	Float denormal operand. One of the floating-point operands is too small to represent a standard float value.

TABLE 1 UDF exception handling error code and messages

Error token	Error code	Error message
<i>isc_exception_float_divide_by_zero</i>	33554477 2L	Floating-point divide by zero. The code attempted to divide a floating-point value by zero.
<i>isc_exception_float_inexact_result</i>	33554477 3L	Floating-point inexact result. The result of a floating-point operation cannot be represented as a decimal fraction.
<i>isc_exception_float_invalid_operand</i>	33554477 4L	Floating-point invalid operand. An indeterminate error occurred during a floating-point operation.
<i>isc_exception_float_overflow</i>	33554477 5L	Floating-point overflow. The exponent of a floating-point operation is greater than the magnitude allowed.
<i>isc_exception_float_stack_check</i>	33554477 6L	Floating-point stack check. The stack overflowed or underflowed as the result of a floating-point operation.
<i>isc_exception_float_underflow</i>	33554477 7L	Floating-point underflow. The exponent of a floating-point operation is less than the magnitude allowed.
<i>isc_exception_integer_divide_by_zero</i>	33554477 8L	Integer divide by zero. The code attempted to divide an integer value by an integer divisor of zero.
<i>isc_exception_integer_overflow</i>	33554477 9L	Integer overflow. The result of an integer operation caused the most significant bit of the result to carry.
<i>isc_exception_unknown</i>	33554478 0L	An exception occurred that does not have a description. Exception number <i>number</i> .
<i>isc_exception_stack_overflow</i>	33554478 1L	Stack overflow. The resource requirements of the runtime stack have exceeded the memory available to it.

TABLE 1 UDF exception handling error code and messages

IMPORTANT Do not rely on this feature as a substitute for writing robust UDF code. Defects in UDF code that raise exceptions are likely to cause other problems in the ibserver process, possibly resulting in data corruption. Use this feature during development to identify and diagnose defects in your UDFs. Do not use UDFs that raise exceptions on a production server. If InterBase raises any exception errors in your production environment, you are strongly recommended to shut down ibserver immediately, and to debug your UDFs.

Specify database cache size on restore

The new `-buffers` switch for `gbak` lets you specify a default cache size for a database when you are restoring it with `gbak`. (The switch is disallowed when creating backups.)

Ownership retained on database restore

gbak -r and gbak -c now retain the original ownership of a database and its objects when a database is restored. Earlier versions of InterBase assigned ownership to the login that was performing the restore.

New error messages

InterBase 5.5 adds the following error messages, in addition to those listed under the changed UDF functionality above.

Error token	Error code	Error message
<i>isc_sort_rec_size_err</i>	33554475 8L	An operation attempted to sort records that are too big
<i>isc_bad_default_value</i>	33554475 9L	Cannot assign a NULL default value to a column with a NOT NULL constraint
<i>isc_invalid_clause</i>	33554476 0L	“invalid clause <i>string</i> ” where “ <i>string</i> ” is the text that the user tried to enter that was determined to be invalid (for example, “NOT NULL DEFAULT NULL”)“
<i>isc_too_many_handles</i>	33554476 1L	Returned if a connection tries to open more than USHORT handles against a port
<i>isc_optimizer_blk_exc</i>	33554476 2L	Optimizer implementation limits are exceeded; for example, only 256 conjuncts (ANDs and ORs) are allowed

TABLE 1 New error messages in InterBase 5.5

Earlier versions: InterBase 5.0 and 5.1 new features

This section describes features that were introduced in InterBase 5.0 and 5.1.1.

Cascading declarative referential integrity

The definition for FOREIGN KEY has been extended to support the SQL 2 standard CASCADE feature for declarative referential integrity. This feature provides a mechanism for defining the actions to be taken in secondary tables when updating or deleting the primary key. You can use this new definition in CREATE TABLE and ALTER TABLE statements.

InterBase 5 enforces compliance with the SQL 92 standard. Refer to the *Data Definition Guide*, Chapter 6, “Working with Tables” for a full description of integrity constraints. Refer to the *Language Reference* for the complete syntax of the CREATE TABLE and ALTER TABLE statements.

New UDF library

InterBase now provides a number of frequently needed functions in the form of a UDF library, which is named *ib_udf.dll* on Windows platforms and *ib_udf* on UNIX platforms. These UDFs are all implemented using the standard C library. This section describes each UDF and provides its declaration.

Refer to *Language Reference*, Chapter 5, “User-Defined Functions” for documentation on declaring and using the functions found in the Interbase UDF library.

Index garbage collection

InterBase 5 performs garbage collection on indexes. Index garbage collection dynamically decreases the size of an index when an index page becomes empty as records are deleted. InterBase retains the recovered disk space for its own use; the space is not returned to the operating system. Index garbage collection is available only in newly created databases and in older databases that have been restored using the new InterBase 5 gbak utility.

New international character sets

InterBase 5 supports the following international character sets:

- BIG_5 (Chinese)
- KSC5601 (Korean)
- GB2312-80 (Chinese)

New security check for reference privileges (GRANT REFERENCES)

A security check for REFERENCES privileges allows the owner of a table to allow or disallow reference to its primary key from a foreign table.

Refer to the entry for the GRANT statement in the *Language Reference* for details on using GRANT REFERENCES.

SQL roles (CREATE ROLE, GRANT, CONNECT)

InterBase now supports SQL roles. This is a four-part process: you first use CREATE ROLE *rolename* to create the role. After creating the role, you use GRANT to grant privileges to the role and then use GRANT again to grant the role to users. A user must then state the role as part of a CONNECT statement to acquire the privileges granted to that role. Refer to the entries for the GRANT, REVOKE, and CREATE ROLE statements in the *Language Reference* for details on group privileges. Note This feature is not supported by clients that use the BDE middleware. To use SQL roles with visual components in Delphi, use InterBase Express (which is part of Delphi 5), or use the third-party components Free IB Components (found on www.interbase.com) or IB Objects (found on www.ibobjects.com).

Improvements in gbak and multifile backup (gsplit)

gbak speed has been improved for both backup and restore functions. Previously the output from gbak had to be a file. gbak can now backup using *stdout* as the backup device, allowing output to be piped to other utilities or sent directly to a tape device. Previously when you ran gbak *create/replace*, the input had to come from a file; now gbak accepts *stdin* as input. Note This feature is supported only on UNIX and Linux.

See the *Operations Guide*, Chapter 8, “Database Backup and Restore” for full documentation on gbak.

gsplit is a new command-line utility that works with gbak to create backup files larger than the OS limit for file size. You can use gsplit to split the backup file created by gbak into multiple files. You can also use gsplit to restore a database from several files.

Refer to the *Operations Guide*, Chapter 8, “Database Backup and Restore” for full documentation on using gsplit with gbak.

New temporary file management

InterBase 5 includes a whole new concept of how temporary file space is managed. InterBase 5 creates two types of temporary files: *sort files* and *history list files*. For full documentation on temporary file management, see the *Operations Guide*, Chapter 4, “Server Configuration.”

Cache configuration

You can set the size of the default cache for a specific database or server-wide and can modify the cache size for a specific ISQL connection. See the *Operations Guide*, Chapter 7, “Database Configuration and Maintenance” for documentation on cache specification.

New user-management API calls

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

Function	Description
<code>isc_add_user()</code>	Adds a user record to the password database
<code>isc_delete_user()</code>	Deletes a user record from the password database
<code>isc_modify_user()</code>	Modifies a user record in the password database

See the *API Guide* for reference documentation on use of these API functions.

Note These API functions are a temporary solution to provide programmatic user management capabilities to the product. InterBase 6.0 will introduce a more extensive collection of API functions for administering servers and databases; the three API functions above will become deprecated at that time.

Improved query optimization

Query optimization has been greatly improved in InterBase 5, reducing the need for you to formulate your own query plans. The following are some of the more notable improvements: The DISTINCT operator has been optimized to use an index where possible; ordering of multi-table joins has been improved with more accurate cost estimation techniques; SQL '92 JOIN syntax has been optimized for INNER JOINS; the optimizer now removes redundant sorting; the optimizer chooses the minimal index set; the PLAN clause now optimizes statements that include an OR specified as part of the WHERE clause; the plan report now includes SORT information;

InterBase Windows ISQL interface enhancements

The InterBase Windows ISQL window is resizable; the View menu and the Extract menu have been combined to form the Metadata menu; there is a new Query menu with entries for executing the current query and for displaying the previous or next query in the SQL Input Area; the Windows ISQL window has been updated to allow for the display of ROLE information; and you can now choose to extract metadata to a .sql file rather than a .ddl file. See the *Operations Guide*, Chapter 10, “Interactive Query” for full documentation on InterBase Windows ISQL and the isql command-line tool.

Performance monitoring

You can now use the `iblockpr` (`gds_lock_print` on UNIX) utility to monitor performance by checking lock requests. For details of using this utility, see the *Operations Guide*, Chapter 9, “Database and Server Statistics.”

New error codes

Applications written for InterBase V3.3 or V4.0 that check for specific values of `SQLCODE` should be reviewed against changes in the error code list. See the *Language Reference*, Chapter 6, “Error Codes and Messages” for a complete list of error codes. The following error codes are new in InterBase 5:

Error token	Error code	Error message
<i>isc_udf_exception</i>	33554474 0L	A fatal exception occurred during the execution of a user defined function.
<i>isc_lost_db_connection</i>	33554474 1L	connection lost to database
<i>isc_no_write_user_priv</i>	33554474 2L	User cannot write to RDB\$USER_PRIVILEGES
<i>isc_token_too_long</i>	33554474 3L	token size exceeds limit
<i>isc_max_att_exceeded</i>	33554474 4L	Maximum user count exceeded. Contact your database administrator.
<i>isc_login_same_as_role_name</i>	33554474 5L	Your login <i>login</i> is same as one of the SQL role name. Ask your database administrator to set up a valid InterBase login.
<i>isc_reftable_requires_pk</i>	33554474 6L	“REFERENCES table” without “(column)”; requires PRIMARY KEY on referenced table
<i>isc_username_too_long</i>	33554474 7L	the username entered is too long. Maximum length is 31 bytes.
<i>isc_password_too_long</i>	33554474 8L	the password specified is too long. Maximum length is 8 bytes.
<i>isc_username_required</i>	33554474 9L	a username is required for this operation.
<i>isc_password_required</i>	33554475 0L	a password is required for this operation
<i>isc_bad_protocol</i>	33554475 1L	the network protocol specified is invalid
<i>isc_dup_username_found</i>	33554475 2L	a duplicate user name was found in the security database
<i>isc_username_not_found</i>	33554475 3L	the user name specified was not found in the security database

TABLE 2 New error messages in InterBase 5.0

Error token	Error code	Error message
<i>isc_error_adding_sec_record</i>	33554475 4L	an error occurred while attempting to add the user
<i>isc_error_modifying_sec_record</i>	33554475 5L	an error occurred while attempting to modify the user record
<i>isc_error_deleting_sec_record</i>	33554475 6L	an error occurred while attempting to delete the user record
<i>isc_error_updating_sec_db</i>	33554475 7L	an error occurred while updating the security database

TABLE 2 New error messages in InterBase 5.0

Migration issues

See the section on Migration in the *Operations Guide*, Chapter 1, “Introduction” for an overview of new, changed, and obsolete components in InterBase 5 that could have some bearing on your upgrade to this current version.

Comparing Classic and SuperServer

This section discusses the similarities and differences between the Classic server and the multithreaded SuperServer.

What's available

SuperServer and Classic server are available as follows:

		InterBase version					
	4.0	4.2	5.0	5.1	5.5	5.6	5.6.1
Win32	Cl.	Sup.	Sup.	Sup.	Sup.	Sup.	Sup.
Solaris	Cl.	n/a	Sup.	Sup.	Sup.	n/a	Sup.
HP-UX	Cl.	n/a	Sup.	Sup.	Sup.	Sup.	Sup.
NetWare	Sup.	Sup.	n/a	n/a	n/a	Sup.	n/a
SCO	Cl.	n/a	n/a	n/a	Cl.	n/a	n/a
Linux	Cl.	n/a	n/a	Cl.	n/a	Cl.	Cl.
AIX	Cl.	n/a	n/a	n/a	n/a	n/a	n/a

Sup. = Superserver Cl. = Classic n/a = not available

Why two implementations?

The Classic implementation predates the SuperServer implementation, and the SuperServer implementation is the future of InterBase.

Classic configuration is used on operating systems that currently don't have the technology for threaded applications, which is required for SuperServer. InterBase also distributes the Classic version on platforms that have threading technology, but which benefit from the low-profile implementation.

SuperServer has a greater ability to meet the demands of a growing multiuser system, while retaining good performance and efficiency. SuperServer is implemented in InterBase product on all platforms where it is technically practical. It is the intention that SuperServer is the future direction of InterBase on all platforms.

Lock management

The lock manager in SuperServer is implemented as a thread in the ibserver executable. Therefore SuperServer does not use the `gds_lock_mgr` process for lock management as the Classic model does. Likewise, SuperServer uses interthread communication mechanisms rather than the POSIX signals used by the Classic server.

Resource use

The SuperServer implementation has less overhead and uses fewer system resources per client connection than the Classic model. SuperServer has one cache space for all client attachments, allowing more efficient use of cache memory. For these and other reasons, SuperServer has demonstrated an ability to efficiently serve a higher number of concurrent clients.

Threaded server and UDFs

User-Defined Functions (UDFs) are libraries of functions that you can add to extend the set of functions that the InterBase server supports. The functions in your UDF library execute within the process context of the InterBase server. Due to the threaded implementation of SuperServer, there are issues with UDFs that require that you write UDF functions more carefully than when writing UDFs for a Classic server.

You must design UDFs for SuperServer as thread-safe functions. You cannot use global variables in your UDF library, because if two clients run the UDF simultaneously, they conflict in their use of the global variables.

Do not use thread-local global variables to simulate global variables. SuperServer implements a sort of thread pooling mechanism, to share threads among all the client connections. It is likely that if a given client executes a UDF twice, that each execution is not executed in the context of the same thread. Therefore, you cannot depend on thread-local variables keeping values from one execution of the UDF to the next for a given client.

UDFs that allocate memory dynamically run the risk of creating a memory leak. Because SuperServer is supposed to stay up and running indefinitely, not just for the duration of the client connection, memory leaks can be more damaging in SuperServer than in Classic. If your UDFs return dynamically allocated objects, then you must use `malloc()` to allocate the memory for these objects (on Win32, you must use `ib_util_malloc()` or the `malloc()` that is part of the Microsoft Visual C++ runtime library). Do not use `new` or `globalalloc()` or the Borland `malloc()`. Finally, such functions must be declared in databases with the `FREE_IT` option of the `DECLARE EXTERNAL FUNCTION` statement.

By contrast, in Classic, there is a separate process for each client connection, so the UDFs are guaranteed not to conflict. Global variables are safe to use. Also, memory leaks are not as dangerous, because any leaked memory is released when the client disconnects.

InterBase recommends that you design UDFs for SuperServer, the more restrictive model, even if you use a version of InterBase implemented with the Classic model. Eventually InterBase will be implemented with SuperServer on the platform you use. If you design UDFs with this assumption, you can upgrade to a later version of InterBase without the risk that your UDFs must be redesigned to work with SuperServer.

The InterBase SuperServer

SuperServer is new on UNIX platforms except for Linux and SCO. On Windows platforms, it was implemented in Version 4.2, but it has not been formally documented on paper, since the previous full documentation release was for InterBase 4.0. The online Help system for Windows contains some information about SuperServer. SuperServer for both platforms is documented in this section.

For a comparison of InterBase's Classic and SuperServer servers, see [“Comparing Classic and SuperServer” on page 18](#).

Note Those using InterBase on Linux and SCO will find InterBase's Classic architecture discussed in [“The InterBase Classic server” on page 22](#).

The InterBase SuperServer architecture

SuperServer is a multiclient, multithreaded implementation of the InterBase server process. This implementation replaces the “Classic” implementation used for previous versions of InterBase.

Classic architecture, the design in InterBase 4.0 and earlier, was process-based. For every client connection, a separate server process was started to execute the database engine, and each server process had a dedicated database cache. The server processes contended for access to the database, so a Lock Manager subsystem was required to arbitrate and synchronize concurrent page access among the processes.

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. The benefits of SuperServer architecture include:

- Having a single server process eliminates bottlenecks resulting from arbitration for shared database pages and reduces the overhead required for multiple process startups and database queries.
- SuperServer improves message interaction performance because a shared library call is always faster than an interprocess communication request to a server process.

- SuperServer improves database integrity because only one server process has write access to the database, rather than one process for each client. All database engine functionality is encapsulated into a unified, protected subsystem that is isolated from user application error.
- SuperServer allows for the collection of database statistics and user information that InterBase's tools can use for performance monitoring and administrative tasks.
- SuperServer is more cost effective than the Classic architecture. All operating systems have limits on the number of OS processes that can run concurrently. SuperServer allows for a fixed number of database threads to be multiplexed over a potentially large number of concurrent database connections. Since these threads are not hardwired to any specific database connection, SuperServer can support a larger number of users with minimum resources use.

Invoking SuperServer

SuperServer runs as a single process, an invocation of the `ibserver` executable. `ibserver` is started once by the system administrator or by a system boot script. This process runs always, waiting for connection requests. Even when no client is connected to a database on the server, `ibserver` continues to run quietly.

The SuperServer process is not dependant on `inetd`; it waits for connection requests to the `gds_db` service itself.

Threads and tasks

The SuperServer process is a multithreaded application. Different threads within the process are dedicated to different tasks. For instance, one thread waits on the `gds_db` service port for incoming connection requests. Other threads are analogous to individual `gds_inet_server` processes in the Classic model, serving client queries.

Security

You can configure SuperServer to run as a non-root UID, for enhanced security. In SuperServer, you can restrict the permissions on database files to allow only the InterBase server UID to access the database.

Managing the SuperServer

For documentation on starting and shutting down the InterBase SuperServer process, see the *Operations Guide*, Chapter 4, "Server Configuration."

The InterBase Classic server

On Linux and SCO, InterBase uses the Classic architecture. This section gives a technical description of the differences and describes how to administer the Classic server software. This section describes features of the Classic server that differ from the SuperServer. The following section compares the two.

Invoking the Classic server

The InterBase Classic server runs on demand as multiple processes. When a client attempts to connect to an InterBase database, one instance of the `gds_inet_server` executable runs and remains dedicated to that client connection for the duration of the connection.

The initiator of `gds_inet_server` is `inetd`, the UNIX service turnkey process. It has a configuration file, `/etc/inetd.conf`, that associates services with the executable that is to receive the connection. When `inetd` receives a connection request for a given service, it looks up the appropriate program in `/etc/inetd.conf`, executes it, and transfers the network connection to the service program.

When the client chooses to disconnect, `gds_inet_server` closes its connection to the database and any other files, and then exits. When there are no clients connected to any database, there should be no invocations of `gds_inet_server` running.

Lock management

Lock management is taken care of by another process, `gds_lock_mgr`. This program is started when the *second* client attaches to a given database. The job of the lock manager is to serve (metaphorically) as a traffic cop. It grants locks on database resources to clients. It also requests that clients relinquish locks on a resource when that resource is in demand by other clients. The `gds_lock_mgr` remains running even after the last client disconnects. The next time a client connects, it can avoid the slight overhead of starting the lock manager process.

The `gds_lock_mgr` process communicates with each client process by using a shared memory area, and a signalling mechanism using the POSIX signals `SIGUSR1` and `SIGUSR2`. Signals are caught in signal handling routines in `libgdslib.a`, and for this reason user applications should not perform signal handling or any modification to the signal mask. Applications which need to use POSIX signals must compile with an alternate InterBase library, `libgds.a`. This library functions identically to `libgdslib.a`, but it handles signals sent by the lock manager in a child process called `gds_pipe`. All client applications compiled with `libgds.a` automatically run with this child process. No changes to application code are needed, only a different linking option.

Resource use

Each instance of `gds_inet_server` keeps a cache of database pages in its memory space, which is likely to result in some duplication of cached data across the system. While the resource use per client is greater than in SuperServer, Classic uses less overall resources when the number of concurrent connections is low.

Local access method

The Classic architecture permits application processes to perform I/O on database files directly, whereas the SuperServer architecture requires applications to request the ibserver I/O operations by proxy, using a network method. The local access method is faster than the network access method, but is only usable by applications which run on the same host as the database.

Monitoring database connections

The menu item Maintenance | Database Connections in the InterBase Server Manager for Windows always reports exactly one connection on a Classic server, no matter how many clients are connected to databases on that server. The reason for this is that every client connection, including that of Server Manager, has its own `gds_inet_server` process on the server, and each instance of that program knows only about its own connection. Only in SuperServer does the server process have the ability to report all client connections on the server.

Security

In order for InterBase Classic to work with a mixture of local and remote clients running as different user ID's, the server executables `gds_inet_server` and `gds_lock_mgr` must run as root. The processes must run with a real uid of root to set their effective uid to that of the client uid. The lock manager must have the superuser privilege to send signals to the processes.

In some IT environments, the presence of executables with `setuid` bits turned on raises concerns about security. Nevertheless, *do not* change the runtime configuration of InterBase server. The `setuid` root configuration of the Classic software is important to its function.

Because applications can run as any uid, database files must be writable by all uids that access the databases. To simplify maintenance, database files are created writable by the whole world. With care, you can restrict these file permissions, so that the database files are safe from accidental or deliberate damage. Make sure you understand file permissions completely before attempting this, because all local and remote clients need write access to the database, even if they intend only to read data.

Installing and configuring Classic

- When you install InterBase server, run the `/usr/interbase/install` script as described in the installation instructions. This script makes the edits to the `/etc/inetd.conf` and `/etc/services` files. The default text added to these files is also kept as `/usr/interbase/inetd.conf.isc` and `/usr/interbase/services.isc`, respectively.
- The line added to `/etc/inetd.conf` is as follows:

```
gds_db stream tcp nowait root /usr/interbase/bin/gds_inet_server gds_inet_server
```

This provides the information for the `inetd` daemon to invoke `gds_inet_server` when a connection request is made for the `gds_db` service.

- The line added to */etc/services* is as follows:

```
gds_db 3050/tcp
```

This maps the service name *gds_db* to the port number 3050. If you change this (for instance, if another software package is using port 3050), you must also change the mapping in the *services* files on all client hosts.

- Read the section on “Configuration parameters” in Chapter 4 of the *Operations Guide*. Some parameters in the *isc_config* file pertain particularly to the Classic server architecture:

- V4_LOCK_SEM_COUNT
- V4_LOCK_SIGNAL
- V4_LOCK_SHM_SIZE

Shutting down Classic

► *Terminating a single client connection*

To terminate one client connection in the Classic architecture, get a list of all *gds_inet_server* processes, with a long listing to show the user ID associated with that connection.

If the client is running on a trusted UNIX or Linux host, the *gds_inet_server* process adopts the user ID of the client application. If each of your clients has a distinct user, you can use the user ID of the server process to identify which instance of *gds_inet_server* corresponds to each client. If you need to terminate a particular user’s connection, you can use *kill* to send a signal to the appropriate *gds_inet_server* process.

If the client is running on a Windows platform, the user ID of the *gds_inet_server* is root. In this case, it becomes unclear which instance of the server process corresponds to each client. You can use other tools, like *netstat*, to trace network connections between processes and remote IP addresses. Clues like this can help you to match a particular instance of *gds_inet_server* with the associated client.

► *Terminating all client connections*

Occasionally you might need to terminate all client connections to InterBase databases on your server or prevent additional connections.

Use the command *gds_drop -a* to terminate all current client connections, both local and remote. This does not by itself prevent subsequent client attachment requests from being served, but by doing this after modifying the *inetd* configuration described later, you guarantee a state in which there are no connections to databases.

There is no tool analogous to SuperServer’s *ibmgr -shut* command. To prevent new remote client connections, you must alter */etc/inetd.conf*.

► *Preventing new connections*

You might have the need to prevent clients from connecting to any databases on your server. To close the server to all remote client connection requests, use one of the following techniques.

- Modify the *inetd* configuration file */etc/inetd.conf* such that it does not recognize the *gds_db* service port mapping:

1. Edit */etc/inetd.conf*
2. Comment out the line for *gds_db*
3. Save the file and exit the editor
4. Send a HUP signal to the process ID of the *inetd* process to force it to read the configuration:

```
ps -ef | grep inetd
kill -HUP inetd_PID
```

Note *inetd.conf* configures *inetd* for subsequent connection requests, but it does not terminate current connections.

- Remove the Server option in the InterBase license certificate file, */usr/interbase/isc_license.dat*. Your certificate file can have individual lines containing the certificate IDs and keys to enable specific product features. The option which enables the server as a daemon which accepts remote connection requests is the S option. Remove or comment out the line which contains the S option. The change takes effect when the next client attempts to connect to the server where you made the change.
- Keep multiple certificate files that contain different sets of options. To activate a certificate file, change its path and name to */usr/interbase/isc_license.dat*. To deactivate a certificate file, rename it or move it to a different directory. The change takes effect when the next client attempts to connect to a database on the server where you made the change.

Troubleshooting the Classic server

If your client applications are unable to connect to a database on a server running the Classic version of InterBase, you have to troubleshoot to determine the cause of the difficulty, and correct it.

Refer to chapter 5 of the *Operations Guide*, “Network Configuration” for a client/server troubleshooting guide. The questions and suggested solutions listed in this chapter help you to analyze an issue with connections. Most of the suggestions are relevant to Classic as well as SuperServer.

Limit on connections to *inetd* on Linux

For InterBase on Linux, the entry added to */etc/inetd.conf* is slightly different from the line added to that file for other ports of InterBase. The entry for Linux is:

```
gds_db stream tcp nowait.100 root /usr/interbase/bin/gds_inet_server
gds_inet_server
```

The *nowait.100* clause specifies that *inetd* is to accept up to 100 remote client connections to the InterBase server per minute. If an application attempts to make the 101st remote connection during a given minute, the application gets the following error message when it tries to connect:

```
Database: hostname:/path/database.gdb
Statement failed, SQLCODE = -902
```

```
Unable to complete network request to host "hostname"
-Failed to establish a connection
-Connection refused
```

If you need to raise this limit, edit the `gds_db` entry in `/etc/inetd.conf`, replacing the number 100 with a higher number that meets your requirements. Then restart `inetd`.

Refer to your Linux documentation for more details on this `inetd` issue.

Compiling and linking

This section discusses compiling and linking issues for platforms other than Netware, which is not a development environment.

Applications

This section explains the compiling and linking options that are available for creating InterBase 5 applications. These steps apply whether the code is output from the `gpre` embedded SQL preprocessor or you're using the InterBase API to code your application. You must link your applications only with the shared GDS library. The pipe server functionality is made obsolete by the SuperServer architecture.

Note It is highly recommended to use an IDE to create a Project for your applications on Windows, instead of using command-line compilation.

► *Windows NT, Windows 95, and Windows 98*

C and C++ Borland C++ 5.0

```
bcc32 -a4 -tWM -tWC -I <interbase_home>\include
    application.c -eapplication.exe <interbase_home>\lib\gds32.lib
```

C and C++ Microsoft Visual C++ 5.0

```
cl -W3 -G4 -Gd -MD -I <interbase_home>\include application.c
    <interbase_home>\lib\gds32_ms.lib /Feapplication.exe
```

► *Linux*

The following procedure for compiling applications applies both when the code is output from the `gpre` embedded SQL preprocessor and when you're using the InterBase API to code your application:

1. If you are programming in embedded SQL, run the `gpre` preprocessor first:

```
gpre -z -m -n filename.e
```

This creates a C code file with the embedded SQL statements converted to code written to use the InterBase API.

2. Be sure to include the appropriate header file, provided by InterBase, in your source file. If the source is the product of a `gpre` precompile, it has already been included for you. The appropriate header file for C and C++ is `/usr/interbase/include/ibase.h`

3. Compile the source code with the GNU compiler. For C:

```
gcc -c application.c -o application.o
```

For C++:

```
egcs -c application.cc -o application.o
```

4. Link the resulting object with the runtime library image:

To link C applications with the InterBase shared library:

```
gcc -o application application.o -lgds -ldl -lcrypt
```

To link C++ applications with the InterBase shared library:

```
egcs -o application application.o -lgds -ldl -lcrypt -lstdc++
```

The pipe library is used for applications that make use of POSIX signals. The InterBase lock manager uses signals to communicate with application processes. To avoid conflicting with the application's use of signals, the pipe library isolates the InterBase signal handlers in an auxiliary process called `gds_pipe`. This process starts automatically and communicates with your application via a POSIX pipe.

Important You must link with the pipe library if your application does its own signal handling or manipulation of the signal mask.

To link C applications with the InterBase pipe library:

```
gcc -o application application.o -static -lgds -ldl -lcrypt
```

To link C++ applications with the InterBase pipe library:

```
egcs -o application application.o -static -lgds -ldl  
-lcrypt -lstdc++
```

User-defined function libraries

This section details the compiling and linking commands to use to build a user-defined function (UDF) library.

Note It is highly recommended to use an IDE to create a Project for your UDF libraries on Windows, instead of using command-line compilation.

► *Windows NT, Windows 95, and Windows 98*

C and C++ Borland C++ 5.0

```
bcc32 -a4 -tWM -tWCD -I<interbase_home>\include  
udflib.c -eudflib.dll InterBase\lib\gds32.lib
```

C and C++ Borland C++ Builder 4.0

```
bcc32 -a4 -tWM -tWCD -I<interbase_home>\include  
udflib.c -eudflib.dll InterBase\lib\gds32.lib
```

C and C++ Microsoft Visual C++

Use the Visual C++ IDE to create a Project for a dynamic link library (DLL).

Delphi Borland Delphi 4.0

Use the Delphi IDE to create a Project for a dynamic link library (DLL).

All exported functions in your library are available as UDFs.

► *Compiling UDF libraries on Linux*

Listed below are compiling and linking instructions for making a dynamically linkable library of InterBase user-defined functions (UDFs) on the Linux platform.

1. Compile the code containing UDFs:

```
gcc -c -fPIC -fwritable-strings udflib.c
```

2. Link your code object file into a dynamically linkable shared library:

```
ld -shared udflib.o -lc -lgds -lm -o udflib.so
```

3. Declare UDF entry points in your database (see the *Language Reference* for full syntax of the DECLARE EXTERNAL FUNCTION statement):

```
isql mydatabase.gdb
SQL> DECLARE EXTERNAL FUNCTION func
CON> INTEGER
CON> RETURNS INTEGER BY VALUE
CON> ENTRY_POINT "myfunction" MODULE_NAME "udflib.so";
```

The UDF dynamic shared library should be located in */usr/lib*, but can be elsewhere, determined by the colon-separated list of directories in environment variables *LD_ELF_LIBRARY_PATH* or *LD_LIBRARY_PATH*. The environment variables must be defined in the process scope of the InterBase process, either local applications or *gds_inet_server* for remote connections. Refer to the *dlopen(3)* man page for details.

Java application development

Java applications written with InterClient require the Java Development Kit. If this doesn't come with your Linux operating system, you can get the JDK at the Java-Linux Porting Project web site. Visit the Internet location:

<http://www.blackdown.org/java-linux.html>

There are a number of free and commercial Java development tools available. Many people consider Java Workshop and Java Studio to be the best cross-platform Java development tools available. You can download patches to allow these tools to run on Linux from the Blackdown web site. Visit the Internet location:

<http://www.blackdown.org/java-linux/javatools.html>

InterClient 1.51 conforms to and requires the Java 2 environment.

A

Bug Lists

This chapter contains a list of some known bugs with workarounds, as well as a list of all bugs fixed for the 5.0 through 5.6.1 releases of InterBase.

Known bugs with workarounds

The following list provides details on how to solve or work around selected InterBase issues. Defect numbering has been changed, so if you are searching for a bug number that is less than 50,000, add 50,000 to the number before searching for the bug in any Borland search facility or bug table. A comprehensive list of all outstanding bugs is not available.

Bug #3297

Problem

Parentheses are disallowed around query-expressions. There is no way to clarify ambiguous queries like:

```
SELECT * FROM TABLE_A UNION  
SELECT * FROM TABLE_B UNION  
SELECT * FROM TABLE_C
```

It's the same problem as the classic programming issue of nesting IF...IF...ELSE. This especially affects queries that mix UNION and UNION ALL.

Workaround

Avoid this type of complex query, and use several simpler queries instead.

Bug #3446

Problem

gpre does not support BASED_ON in ANSI C-style function parameter declarations:

```
int foo( BASED_ON states.population fpop,  
        BASED_ON states.state fstate )
```

Workaround

Use traditional, “K&R” C-style function parameter declarations:

```
int foo(fpop, fstate)  
BASED_ON states.population fpop;  
BASED_ON states.state fstate;
```

Bug #7520

Problem

CREATE TABLE which references a field of another table as its foreign key fails if an explicit unique index has been created for that field in the other table.

Workaround

Do not create the unique index on a primary key column. A primary key already implicitly creates a unique index, so creating another one is unnecessary.

Bug #8251

Problem

isc_dsql_execute() API call does not work when used to submit an EXECUTE STORED PROCEDURE statement which does not take any input parameters.

Solution

Use isc_dsql_execute2() for EXECUTE statements that have no input parameters.

Use isc_dsql_execute() for EXECUTE statements that do have input parameters.

Bug #8412

Problem

Backing up with gbak is extremely slow when the database contains a large number of back record versions.

Workaround

Using gbak -b -g inhibits the normal garbage collection task; this alleviates most of the performance problem.

Bug #8429

Problem

If you fail to install a valid license file, and attempt to start the server by using `ibmgr`, the server starts even though `ibmgr` reports an error. But `ibmgr` doesn't shut down the server when requested if it has no access to verify the `sysdba` password.

Solution

To shut down the server forcibly, use `kill` to halt the `ibguard` and `ibserver` processes, in that order. This is safe to do, because if there is no license, then there is no way there could be any active transactions on the database.

Bug #8542

Problem

Applications with more than one `gpre'd` file don't link because of the duplicate symbol definitions.

For each `gpre'd` file there are variables created that are given a global scope (for example, `isc_trans`, or the status vector). When two or more `gpre'd` files are compiled/linked it always produces linker errors because these variables are defined multiple times.

Workaround

Use `#define` to change the name of the global variables to something specific to each file.

Bug #8591

Problem

Running `isql -extract` against an old-style database that has GDML definitions for domains, views, triggers or `COMPUTED BY` columns outputs a mixture of SQL and GDML. Those metadata constructs are stored in Blobs, and are not converted by the extraction step, as is all the other metadata. `isql` cannot read GDML, so in this case the output of `isql` cannot be read by `isql`.

Solution

After you extract the metadata with `isql -extract`, you must rewrite all GDML as equivalent SQL code before using it as an SQL script.

Note This bug affects only databases that contain certain types of GDML metadata. Any database that was conceived with InterBase V4.0 or later should not have any GDML metadata.

Bug #8600

Problem

The InterBase 4.2 client does not detect a server disconnection when connected via `Netbeui/Named Pipes`. Therefore client applications may occasionally fail with the error message, "no process is on the other end of the pipe."

Solution

The InterBase 5 client correctly detects a disconnected server and doesn't try to use a dead connection. Old client versions still have the bug. You should upgrade both the InterBase client and server software to InterBase 5.

Bug #8813

Problem

Loading DLL's on Windows NT doesn't work if you specify the DLL without using the .DLL file suffix, and the DLL is located under a directory whose name contains a dot. For example:

- C:\A.B\MYLIB does not work
- C:\A.B\MYLIB.DLL works
- C:\A_B\MYLIB works
- C:\A_B\MYLIB.DLL works

This is a Microsoft bug. It is independent of InterBase. The same behavior exists when you programmatically load a library with the Windows API function *LoadLibrary()*.

Workarounds

The easiest solution is to always explicitly use the .DLL file suffix (as in the second case above) when specifying DLL's, for instance in a DECLARE EXTERNAL FUNCTION statement.

Another solution is to make sure that the path to your DLL contains no dots.

TIP If you have existing UDFs declared in your database, and their module names do not have the .DLL file suffix, you can update the declarations by connecting as SYSDBA and executing the following SQL statement:

```
UPDATE RDB$FUNCTIONS
  SET RDB$MODULE_NAME = RDB$MODULE_NAME || '.DLL'
 WHERE RDB$MODULE_NAME NOT LIKE '%.DLL';
```

Bug #10069

Problem

Creating a specific kind of improper stored procedure results in corruption of a system index.

It is not permitted to create a procedure that references a generator that doesn't exist. This is an appropriate restriction. For example:

```
CREATE PROCEDURE BUG10069 RETURNS (X INTEGER) AS
BEGIN
  X = GEN_ID(MISSING_GENERATOR, 1);
END
```

If MISSING_GENERATOR does not exist, InterBase immediately returns a number of error messages such as:

```
Statement failed, SQLCODE = -104
```



```

invalid request BLR at offset 47
-generator MISSING_GENERATOR is not defined

Statement failed, SQLCODE = -902
internal gds software consistency check (invalid SEND request (167))

Statement failed, SQLCODE = -902
internal gds software consistency check (can't continue after
bugcheck)

```

InterBase fails to undo some of the work done by create procedure. Specifically, the system indexes RDB\$INDEX_18 on table RDB\$PROCEDURE_PARAMETERS, and RDB\$INDEX_21 and RDB\$INDEX_22 on table RDB\$PROCEDURES are left in an corrupted state. You can test a database for this kind of corruption by using the command: `gfix -v -f database.gdb`. `gfix` detects but does not correct the corruption. Look for errors:

```

DBSERVER (Server)      Mon Aug 31 13:47:05 1998
  Database: D:\DATABASE.GDB
  Index 1 is corrupt (missing entries) in table RDB$PROCEDURES(26)

DBSERVER (Server)      Mon Aug 31 13:47:05 1998
  Database: D:\DATABASE.GDB
  Index 2 is corrupt (missing entries) in table RDB$PROCEDURES(26)

DBSERVER (Server)      Mon Aug 31 13:47:05 1998
  Database: D:\DATABASE.GDB
  Index 1 is corrupt (missing entries) in table
  RDB$PROCEDURE_PARAMETERS (27)

```

Data in the database is unaffected. Only the indexes named above are damaged.

Solution

The easiest solution is prevention. Always create a generator before you use it in a stored procedure.

If you find that you have referenced a non-existent generator and you have corrupted the index, you can use the following methods to repair the damaged indexes:

- Rebuild the indexes by using the following commands:

```

ALTER INDEX RDB$INDEX_18 INACTIVE;
ALTER INDEX RDB$INDEX_18 ACTIVE;

```

Repeat these steps with RDB\$INDEX_21 and RDB\$INDEX_22.

- Backing up and restoring the database forces InterBase to rebuild all indexes, so this is also an effective way to repair the corrupted indexes.

Note `gfix -mend` does not fix this type of index corruption.

IMPORTANT Always use the `gfix -v -f` command after attempting to repair the corrupted indexes, to verify that the indexes have in fact been repaired in the correct database.

Bug #10072

Problem

The server crashes when you execute the following or similar SELECT statement:

```

SELECT RDB$DB_KEY FROM ANY_STORED_PROCEDURE;

```

Workaround

Do not select the RDB\$DB_KEY pseudocolumn in a query that invokes a Select Procedure.

Bug #10098

Problem

If you back up a database that has domains defined but no tables, restoring the backup file gives non-fatal error messages reporting deadlock. You can use the restored database, but this is not recommended because you cannot back it up again.

Workaround

Before you back up a database with domains, create at least one table.

Connection error: “REMOTE INTERFACE not licensed”

Problem

On Windows NT or Windows 95, after you uninstall InterBase 4.x and install InterBase 5.x, you are no longer able to connect to remote InterBase servers with BDE clients. You get the following error message:

REMOTE INTERFACE not licensed.

All InterBase tools (Windows ISQL, ComDiag) work normally. Local connections using BDE tools (for example, Delphi, Paradox, C++Builder) work normally. Only remote connections using BDE tools fail.

Solutions

There are several causes for this situation. Use the following troubleshooting methods:

- Find and remove old copies of *GDS32.DLL*, especially those that are in the BDE install directory. You must have only one copy of *GDS32.DLL* on a given system, and it must match the version of the InterBase client you use. Right-click on a file and check the Properties sheet to determine its version.
- If this does not fix the problem, it is possible that your InterBase client does not have a legitimate remote client software activation key. Use the License Manager tool (iblicense.exe) to verify that the InterBase Remote Client certificate is installed. If the certificate is not installed, add it by specifying the certificate ID ISC-30811, and the certificate key ca-9-3a-0.
- Your client workstation might not have the InterBase SQL-Links driver installed, or has an outdated version of the BDE or InterBase SQL-Links driver. Upgrade your software. Check the INPRISE web site for downloadable updates.
- If you uninstalled an old version of BDE before installing an updated version, you might get this error message. Remove all files from the old version of BDE before installing a new version.
- Finally, the entry in the Windows Registry for the BDE's DLLPATH might be invalid. Run regedit and find the Registry key: *HKEY_LOCAL_MACHINE/ Software/ Borland/ Database Engine*. The item *DLLPATH* should indicate the directory where BDE and SQL-Links DLL's are located. Check that the registry item lists the correct directory, and fix the entry if necessary.

Installation error: “Internal error near IBcheck”

Problem

Trying to install InterBase on Windows NT or Windows 95 from the JBuilder or Delphi CDROM produces the error message:

```
String variable is not large enough for string. Check the string
declarations. Error: 401
```

This is followed by the error:

```
Internal error near IBcheck
```

After this the installation aborts.

Solution

Run regedit and check the registry key *HK_CURRENT_USER/Environment*. The PATH item should be a registry item with a string type. JBuilder and Delphi use InstallShield to install the product. InstallShield has a bug such that it incorrectly creates PATH as a binary item. If the PATH item has a binary type, delete it or convert it to a string item. Then restart the installation of InterBase.

Bugs fixed for InterBase 5.0

If you are searching for a bug number that is less than 50,000, add 50,000 to the number before looking it up in the following table.

Bugs fixed for InterBase 5.0	
Bug No.	Brief Description
51640	Nested subqueries are very expensive
52214	Optimizer uses redundant indexes
53095	Near-identical query takes 400X cpu time
53516	Performance statistics produce negative number for elapsed time
53753	Performance problems over long-haul network connections
53870	Corrupt database
54181	Views based on equivalent queries were optimized differently
54742	Optimizer problem causing poor performance
54785	gfix -write sync is not saved if db is backed up & restored
55599	View performance problem
55977	Parser does not permit table update from another table
56498	Need a way to specify location of temp files and <i>interbas</i> directory
56879	gds consistency check (differences record too long (182))

TABLE 1 Bugs fixed for InterBase 5.0

Bugs fixed for InterBase 5.0	
Bug No.	Brief Description
57039	Bad performance with table joins and indexes
57173	gbak names everything “volume 0”
57459	gstat doesn’t indicate whether the database is shut down
57565	Large PLANs are not displayed;
57953	Implement \$INTERBASE for all product files
57973	Query on view fails to use index
57975	Vanishing client makes CPU grieve over new events
58001	GRANT ALL ON <i>table</i> TO <i>user</i> for 95 tables takes 5 minutes
58039	A join of views give db corruption error
58054	No warning when a field that has a multisegment index on it is dropped
58055	gbak fails if index on integer w/scale is changed to just integer
58066	Remove “D” license requirement for GRANT/REVOKE operations
58071	No security on the RDB\$USER_PRIVILEGES table
58072	Cannot restore gbaked database if stored procedures use PLAN option
58073	gbak doesn’t abort if stored procedures with PLAN clause depend on inactive indexes
58076	SQL UDFs cannot return Blobs
58086	<i>perf.c</i> elapse times are less than cpu times
58093	isql extract does not get the name of the RI constraint
58097	Remote access via serial networks is unreliable
58104	gds_inet_server processes start up when unlicensed client fails to connect
58105	gbak performance is slow
58106	Duplicate indexed fields are not identified when brought from external file
58126	DSQL dumps core on a long illegal statement
58129	Correlated subquery crashes NLM, core dump on UNIX
58132	SYSDBA can’t change privileges when moving V3 to V4
58134	gpre translates named transaction to <i>gds_trans</i>
58135	No error message return when connect in isql with bogus user and password
58141	<i>blr_store2</i> core dumps
58149	Can (sometimes) delete from RDB\$TRIGGERS

TABLE 1 Bugs fixed for InterBase 5.0

Bugs fixed for InterBase 5.0	
Bug No.	Brief Description
58150	Count of DISTINCT records returns wrong result
58158	DISTINCT clause causes ORDER BY clause to be ignored
58168	Reference privilege on tables is undocumented
58169	SQL doesn't allow GRANT to a group
58172	isql does not parse number of cache buffers
58180	DISTINCT causes incorrect ordering when used with ORDER BY
58183	Cannot transliterate character between 3.3 and 4.0 character sets
58185	gds_lock_print -i core drops
58189	Remote prefetch loses accurate error codes
58193	Dropping procedure creates access violation
58201	Security class restrictions do not migrate to IB version 4.0
58219	Trigger aborts but still deletes records
58238	Correlated subquery does not product right number of columns
58249	gds_lock_mgr, gds_pipe5, gds_pipe closing only 20 file descriptors
58253	gbak fails when stored procedure inserts to non-updateable view
58254	LIBS 4.1 gfix is broken
58255	gpre generates the wrong PIC X(8) value for COBOL programs
58261	isql -x does not extract domain information properly
58262	Error: Unsuccessful metadata update: depth exceeded (recursive definition)
58265	ON UPDATE SET DEFAULT feature (RI/CASCADE) fails
58269	Messages with a message length but no message are crashing the server
58270	A UDF returning a <i>cstring()</i> greater than 32752 causes a GPF in <i>ibserver.exe</i>
58284	Optimizer bug on same query with different positioning of fields
58285	db file owned by root may not be the best design
58290	Solaris system stalls when running gbak utility
58293	Dynamic gpre gives segmentation violation with COBOL
58294	External table feature is a security hole
58295	External table doesn't get properly gbaked and restored
58309	Solaris system hangs when declaring referential integrity actions

TABLE 1 Bugs fixed for InterBase 5.0

Bugs fixed for InterBase 5.0	
Bug No.	Brief Description
58314	No option for <i>-user</i> or <i>-password</i> in gsec
58321	'C' and 'COBOL' give different error codes with same program
58322	isql extract of international domain produces incorrect syntax
58332	Differences record is too long (182)
58338	Column aliases do not work with UNION
58342	Character set name is included with the default for a field
58343	Stored procedure consistently crashes server
58344	No documentation for trigger updating through a view
58345	gbak -o fails with UDF db
58348	Calling get_blob_segemnt, put_segemnt, and BLB_lseek hangs SuperServer
58349	SELECT query with ORDER BY crashes server
58350	Deadlock error in version 4.2.1
58354	NULL XSQLDA in isc_dsql_fetch() causes remote server to crash
58359	4.5 sync err, localhost err and server dies
58361	isql -x does not extract metadata from pre-4.5 databases
58365	gbak produces deadlock error during restore
58366	gpre -x does not work
58373	Problem dropping table in ISQL after stored procedures dropped
58377	Depth exceeded error from schema file in ISQL1_05 TCS test
58378	SuperServer uses "interbase", which is not an allowed username on HP10
58379	COBOL application returns sqlcode 0 when sqlcode -100 is expected
58395	Ambiguous positioned update of cursors with same name
58396	DSQL parser: cursor named after SQL keyword disables that keyword
58397	Constraint creates index, preventing DROP TABLE/COLUMN
58400	isql doesn't display NOT_NULL constraint
58402	Server crashes with SELECT statement involving large VARCHAR / ORDER BY
58406	Operations do not correctly inherit privileges of procedures
58408	gstat -x does not display all available flags
58409	Cannot read error for <i>isc_guard1</i> machine if started as interbase user

TABLE 1 Bugs fixed for InterBase 5.0

Bugs fixed for InterBase 5.0	
Bug No.	Brief Description
58414	Consistency chk error when GRANT is done before a grant to unix grp
58415	Error message with DISTINCT on subselect statements
58416	Error using SELECT DISTINCT on an indexed field with a stored procedure
58417	Server does not return memory to OS
58418	gfix shutdown timeout value appears not to work
58419	ALTER TABLE ... DROP CONSTRAINT command drops ibserver
58437	Lack of DROP GENERATOR statement should be explained in documentation
58441	gbak to tape does not work
58445	ALTER TABLE ... DROP (column) fails when it should succeed
58457	SELECT statement with singleton count crashes server
58465	gpre fails to add an "END - IF" in a COBOL program
58784	Indrxes were using character set ID instead of text type for length; usage of Portuguese collation (and others) would result in a "not found" error message

TABLE 1 Bugs fixed for InterBase 5.0

Bugs fixed for InterBase 5.1.1

Bugs fixed for InterBase 5.1.1	
Bug No.	Brief Description
54840	Adding a column with DEFAULT clause fills column with zeroes; should be default value
57995	Cannot initialize event subsystem when the OS has inadequate semaphores
58589	License tool needs to enforce evaluation key; user should not remove eval key
58640	Updating or inserting values from one table to a second table crashes the server
58698	Expiration date of evaluation key is not Year 2000 correct
58725	Server occasionally fails to increment next transaction ID
58729	InterBase Windows ISQL has no connection property for SQL roles
58732	SQL script hangs on GRANT statement; internal ACL data structure has static size

TABLE 2 Bugs fixed for InterBase 5.1.1

Bugs fixed for InterBase 5.1.1	
Bug No.	Brief Description
58734	isql's flag for SQL roles doesn't work
58749	isql -extract does not generate GRANT ROLE to <i>user</i>
58752	Memory leak on server
58754	Server process size exceeds OS limit under heavy load; server hangs
58757	Server process grows in size without releasing memory
58781	Server does not release DSQL statement memory pools on database detach operation
58812	Database corruption; same cause as in bug 8732
58825	Temporary deadlock caused by gfix -attach
n/a	Server exits when lock manager memory is exceeded (UNIX only)
n/a	Miscellaneous memory leaks on client
n/a	No clear error message when memory is exhausted
n/a	Need IPX/SPX support in Win32 client in order to connect to NetWare servers

TABLE 2 Bugs fixed for InterBase 5.1.1

Bugs fixed for InterBase 5.5

Bugs fixed for InterBase 5.5	
Bug No.	Brief Description
58580	New Install script and <i>iblicense</i> behavior not documented in the <i>readme.txt</i>
58676	Online help for WISQL not updated
58732	IB server stalls when a Lock Manager contention occurs
58775	Execute stored procedure from InterClient application crashes IB server
58780	Update from trigger doesn't get rolled back
58786	Transliteration exceptions converting Unicode_fss to DOS437
58787	Transliteration exceptions converting Unicode_fss to DOS850
58788	Transliteration exceptions and corruption converting Unicode_fss to DOS852
58789	Transliteration exceptions converting Unicode_fss to DOS857
58790	Transliteration exceptions & corruption converting Unicode_fss to DOS860
58791	Transliteration exceptions converting Unicode_fss to DOS861

TABLE 3 Bugs fixed for InterBase 5.5

Bugs fixed for InterBase 5.5	
Bug No.	Brief Description
58792	Transliteration exceptions converting Unicode_fss to DOS863
58793	Transliteration exceptions converting Unicode_fss to DOS865
58796	Transliteration exceptions converting Unicode_fss to NEXT
58798	Transliteration exceptions & corruption converting Unicode_fss to WIN_1250
58799	Transliteration exceptions & corruption converting Unicode_fss to WIN_1251
58801	Transliteration exceptions & corruption converting Unicode_fss to WIN_1253
58802	Transliteration exceptions converting Unicode_fss to WIN_1254
58803	Too many metadata versions corrupt a database (ALTER TRIGGER)
58812	A large number of GRANTS on a single table corrupts the database
58813	Cannot load <i>gdsintl</i> if the InterBase install path has a dot in it; document workaround
58822	Executing a view that is based on another view crashes the IB server
58825	gfix -attach does not allow active transaction to complete
58842	Multiuser test allows only 96 NT connections
58850	Restore of a backup fails if the database contains a stored procedure that references a view
58851	WARNING: column EM_FNAME is not defined in table RDB\$PAGES
58864	CAST() returns wrong result with NUMERIC numbers less than 1
58867	Asking for better documentation on IB 5.0 license
58880	Casting numerics to CHARs fails when CHAR field is too large
58881	Casting the largest negative integer to CHAR produces error
58883	Error selecting data from an external file
58891	Too many (200+) ALTER statements in a script produce "request depth exceeded" error
58906	<i>isc_info_base_level</i> returns wrong version
58921	gbak does not preserve ownership of stored procedures
58926	gbak failure for some databases containing triggers for referential integrity
58927	gbak does not preserve ownership of tables
58935	GRANT on stored procedures produces inconsistent behavior
58936	rtrim() in a SELECT statement drops connection to IBServer

TABLE 3 Bugs fixed for InterBase 5.5

Bugs fixed for InterBase 5.5	
Bug No.	Brief Description
58937	ltrim(), lower(), and rtrim() return faulty length of a CHAR field
58938	lower() returns incorrect results
58945	UDF compiled with Borland C++ doesn't release memory
58949	Dropping a trigger causes core dump on HP-UX
58952	SYSDBA cannot revoke privileges from user when privileges have been proliferated to others
58955	ALTER TRIGGER INACTIVE causes core dump
58963	IB server fails on NT when triggers are repeatedly dropped or altered
58967	UDF functions sometimes crash the IB server
58968	gbak for IB5.1.1 cannot restore a .gbk file from previous version of IB
58987	isql script crashes server
58988	Altering a procedure while it is in use by another procedure results in an abortive bug check
58990	SHOW TRIGGERS is very slow due to unavailable system index
58995	Multistep triggers may not execute all steps
59015	Procedure ownership is not preserved through gbak -restore
59016	Database objects are owned by the user performing a restore
59018	Cannot alter a procedure that calls another procedure after its parameter list has changed
59019	CHECK constraints don't check trigger results
59023	IB server fails when a trigger is dropped while it is in use
59026	Possible lock manager problem
59027	IB server fails when a stored procedure is dropped immediately after execution
59033	SHOW GRANT functionality skips TRIGGER keyword
59038	Invalid check on EXECUTE permission for stored procedure
59053	Too many informational messages in <i>interbase.log</i>
59061	Improper pathname for gsec -database argument hangs NT server
510064	Rampant memory growth during restore of databases with certain combinations of dependant stored procedures
510085	Committing certain types of stored procedures crashes the server
n/a	ibserver crashes and/or corrupts data when more than 50 users put heavy demands on the server simultaneously and for an extended period of time

TABLE 3 Bugs fixed for InterBase 5.5

Bugs fixed for InterBase 5.6

Below is a listing of bugs fixed in InterBase version 5.6:

Bugs fixed for InterBase 5.6	
Bug No.	Brief Description
57367	Memory leak in <i>jrd/sort.c</i> . During the process of sorting records memory was leaking merge blocks. This problem did not affect the success of a sort; it simply leaked resources.
57376	The parent relation lock used for refresh was getting released, but was later still assumed to be valid, which resulted in the error: Error: I/O Error during "send" operation for file "xxxxx" Unknown error 10054.
58193	Creating and dropping a stored procedure repeatedly causes a server crash on the third execution of the DROP command
58385	Queries using MIN in DELETE statements cause data loss; for example: DELETE FROM foo WHERE (SELECT MIN(bar) FROM foo);
58494	Long query causes other users' work to hang until the query is finished or terminated
58639	Referencing a computed field whose definition includes a stored procedure requiring an input argument crashes the server; for example: create procedure test returns (out_val integer) as begin out_val = 3; suspend; end create table t2 (f1 integer, f2 computed by ((select out_val from test))); select * from t2;
58712	A poorly written stored procedure or trigger that exhibits runaway recursion can overflow the internal stack and crash the server
58756	Query plan generation for outer joins was broken from V4.2.1 to V5.5
58847	Database validation reports bogus page pointer corruption error for external tables
58936	RTRIM() UDF function could crash the server; there were problems with string manipulation in LTRIM() and LOWER() as well

TABLE 4 Bugs fixed for InterBase 5.6

Bugs fixed for InterBase 5.6	
Bug No.	Brief Description
58958	<p>DISTINCT does not return a correct row count when a non-unique index is referenced; for example:</p> <pre>select distinct customer from sales s, customer c where s.cust_no = c.cust_no and total_value > 10000;</pre> <p>This generates the plan (where the CUSTNAMEX index is non-unique):</p> <pre>PLAN JOIN (C ORDER CUSTNAMEX,S INDEX (RDB\$FOREIGN25))</pre>
59014	The optimizer was improperly handling equivalency pairs found in non-unique indexes, which caused the server process to grow unnecessarily and sometimes crash
59023	Attempting to drop and recreate a trigger that is in use would sometimes crash the server
59080	<p>Using aggregates in a query on a view with aggregates could crash the server; for example:</p> <pre>create view vw_shipping (orderid, lag, shipvia, summation) as select o.orderid, o.shipdate - o.saledate, shipvia, (select sum(total) from lineitem li where li.orderid = li.orderid) from orders o;</pre> <pre>select lag, avg(summation) from vw_shipping group by lag order by lag;</pre>
60013	Repeatedly using SELECT COUNT(*) from a view performing a join could crash the server on the third execution
60064	Restore or build of a database with a very large and complex schema using gbak or isql could cause the server to crash
60085	Committing large numbers of stored procedures at one time could crash the server when virtual memory is low
60101	An automated or manual sweep can crash the server upon encountering a table with metadata containing indexes with null values that are not currently loaded into the server's memory space
60109	Repeatedly executing a stored procedure referencing a UDF without commits between execute statements could crash the server
60116	Left outer joins produce incorrect results with default values

TABLE 4 Bugs fixed for InterBase 5.6

Bugs fixed for InterBase 5.6

Bug No.	Brief Description
60124	<p>Cast produces the wrong result in type conversions; for example:</p> <pre> create table t1 (f1 integer); create table t2 (f1 numeric(15,1)); insert into t2 values(1.0); insert into t2 values(10.0); insert into t2 values(100.0); insert into t2 values(1000.0); commit; insert into t1 select cast(f1 as integer) from t2; commit; select * from t1; F1 ===== 0 1 10 100 </pre>
60135	<p>UPDATE fails inappropriately with ri constraint and throws the error: Statement failed, SQLCODE = -530 violation of FOREIGN KEY constraint "FK_B_A" on table "B" For example:</p> <pre> create table a (a1 integer not null constraint pk_a primary key, a2 integer not null constraint u_a unique); create table b (b1 integer constraint fk_b_a references a (a1)); insert into a values (1,1); insert into a values (2,2); insert into b values (2); commit; update a set a2 = 999 where a1 = 2; </pre>
60137	Non-unique indexes appeared to be corrupt, but the actual problem was with the walking of the index nodes
60164	Altering a stored procedure could crash the server and then cause a restore to fail after backing up a database
60166	Restore fails on a database with very large and complex schema even though the database was successfully backed-up; this is related to bug #60064

TABLE 4 Bugs fixed for InterBase 5.6

Bugs fixed for InterBase 5.6	
Bug No.	Brief Description
60206	Updating a table with VARCHARs with a total length greater than 32K could corrupt a database because internal variables pertaining to length calculations were wrapping around
60302	<p>Declaring a procedure which calls a UDF with an incorrect number of parameters could crash the server and cause problems when trying to alter or drop the procedure; for example:</p> <pre> declare external function lower cstring(20) returns cstring(20) free_it entry point 'ib_udf_lower' module name 'ib_udf'; commit; create procedure foo (arg_one char(10), arg_two char(10)) returns (out_val char(10)) as begin out_val = lower(:arg_one, arg_two); end </pre>
60313	Queries continue to run to completion after a client abnormally disconnects. Servers will now be able to detect that the client is gone and will terminate the request. This behavior is specific to a remote TCP connection.
60348	<p>A SELECT query in which the WHERE clause contains search criteria larger than the target field could crash the server; for example:</p> <pre> create table t1 (f1 integer, f2 varchar(10)); create index idx_f2 on t1(f2); select * from t1 where f2 = 'xxxxxxxxxxxxxxxxxxxxxxxx'; </pre>
60362	Certain instances of procedures that call the internal <i>node_match()</i> function could crash the server
60389	gfix -shut -force does not always shut the database down to exclusive access mode
60420	Using the IBX Events component over NetBEUI consumes 100% of the CPU upon disconnect
60435	Dynamically unloading gds32.dll leaks resources

TABLE 4 Bugs fixed for InterBase 5.6

Bugs fixed for InterBase 5.6	
Bug No.	Brief Description
60453	Database validation reports a false index corruption error: <pre>MyServer (Server)Wed Mar 24 15:02:42 1999 Database: c:\test\test.gdb Index 1 is corrupt (missing entries) in table foo (153)</pre> This problem is most commonly seen after a server crash or when the InterBase service is shut down while there are active connections writing to the database.
60469	Scripts which access non-existent external files could crash the server
60520	gds32.lib cannot be used by C++ Builder because it no longer supports aliases
60537	An update failure with a unique key constraint could corrupt a database; this is similar behavior to bug #60135, but with a different error: <pre>Statement failed, SQLCODE = -902 internal gds software consistency check (wrong record length (183))</pre>

TABLE 4 Bugs fixed for InterBase 5.6

Bugs fixed for InterBase 5.6.1

Below is a list of bugs fixed for InterBase release 5.6.1. For more information, search the Borland community site at <http://search.borland.com>.

Bugs fixed for InterBase 5.6.1	
Bug No.	Brief Description
71191	Server crashes when the client is abnormally terminated. This only occurs when the client is reading or receiving data from the server. (HP_UX only)
100627	The superserver on Windows NT could not service more than 256 users.
100649	Internal QA UDF use could have caused some database corruption.
100651	Some versions of InterBase allowed unsecure access to databases.

TABLE 5 Bugs fixed for InterBase 5.6.1