InterBase 6

Release Notes

Borland®

SOFTWARE CORPORATION

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

1INT0050WW21003   5E4R0898

9899000102-9 8 7 6 5 4 3 2 1

D4

CHAPTER

1 General Information

This chapter covers the following topics:

- Documentation and Adobe Acrobat Reader
- Contacting Borland
- Installation and Migration
- System requirements

IMPORTANT See *Getting Started* for detailed installation instructions and information about migrating existing databases and applications.

InterBase documentation

Only the *Release Notes* have been updated for InterBase 6.5. InterBase supplies online documentation in Adobe Acrobat PDF format. All documents except *Release Notes* and *Getting Started* are also available in soft-cover books. The document set consists of the following:

Document Title	Contents
Release Notes	Contact information, new feature descriptions, known problems and work-arounds
Getting Started	Complete installation information, how to migrate from previous versions of InterBase
Operations Guide	Configuring and operating InterBase Server, using command-line and GUI tools
Data Definition Guide	Designing and building InterBase databases, using database utilities
Developer's Guide	Developing InterBase database applications using Borland development tools
Embedded SQL Guide	Developing InterBase database applications using embedded SQL
API Guide	Developing InterBase database applications using the InterBase Application Programming Interface
Language Reference	InterBase SQL, DSQL, isql, and stored procedure and trigger language elements, syntax and semantics

TABLE 0.1 InterBase documentation set

Documentation installation

You have the option to install the complete document set in PDF form on your hard drive when you are installing the InterBase 6.5 product. This requires about 24MB of disk space. You also have the option of installing shortcuts to the documentation, which requires very little disk space, but which requires that you insert the CD-ROM every time you want to refer to the documentation.

IMPORTANT You need Adobe Acrobat Reader With Search to view and search these documents. See the Getting Started manual for information on 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 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.

Tip On UNIX and Linux, always open documents using an absolute pathname to the PDF file, to make Acrobat Reader With Search associate the index with the PDF document correctly.

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**.

Contacting Borland

Mailing address	Borland Software Corporation 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://community.borland.com/interbase** • For installation and presales questions, visit: **http://www.borland.com/devsupport** • To discuss issues, 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/**

TABLE 0.2

Installation and Migration

This section contains information on installation, migration, and setting up UNIX clients. Complete installation instructions are in the installation chapter of *Getting Started*.

InterBase Client for UNIX

InterBase 6 comes with a UNIX-based client package called "InterBase Client Only" (not to be confused with the InterBase JDBC driver called InterClient). The package contains a version for Solaris, which can be installed from the CD-ROM using the setup script. The Setup script can also be used to install the InterBase documentation and the Adobe Acrobat Reader from the CD-ROM. For more information on installing and setting up the InterBase Client for UNIX, refer to the installation chapter in *Getting Started*.

Migration to InterBase 6 and 6.5

Migrating databases from previous versions to InterBase 6 requires planning. First you must learn about:

- Transition features (features whose meanings have changed between InterBase versions)
- Dialects (indicators that specify how transition features and interpreted)
- On-disk structure (differences in architecture).

Begin by reading chapter three of these release notes to learn more about dialects and how features have changed in the 6.01 and 6.5 releases, then read the Migration chapter in *Getting Started*.

Running InterBase 6.5 as an application

In Windows NT, InterBase 6.5 runs as a service; that is, the InterBase Server and Guardian start automatically when you boot up your system. To run InterBase 6.5 as an application that you can start from the desktop:

1. In the Control Panel, choose Services.
2. In the Services window, select InterBase Guardian and click Stop.
3. Click Startup and select Disabled as the Startup type.
4. Create a shortcut to *ib_install_dir*\bin\ibguard.exe.
5. Open the Properties for the shortcut.

6. Click the Shortcut tab.
7. At the end of the Target field, add -a to the target statement, after the quotation mark. For example:

```
"C:\Program Files\Interbase Corp\Interbase\bin\ibguard.exe" -a
```

Once you have completed these steps, you can double-click the shortcut icon to start the InterBase Guardian and Server.

Uninstalling InterBase 6.5

The Windows InterBase installation allows you to choose between performing a complete install or selecting individual components. If you install a subset of the components, you can run the install again later to add more components. However, when you uninstall, only the components that you installed the last time are removed. For additional information regarding uninstalling on all platforms refer to the installation chapter in *Getting Started*.

System requirements

Installation requires approximately 44MB of disk space for a full install that includes InterBase, Adobe Acrobat Reader, and the full document set. Only 17MB is needed to install the InterBase product without the documents, examples, or Acrobat Reader. For a complete listing of disk requirements, system hardware, OS and compiler requirements, refer to *index.html*.

CHAPTER 1

InterBase 6.5 features

This chapter lists features introduced in InterBase 6.5 and provides an overview and summary of each.

Bold green type indicates clickable links.

Feature overview

- Security

  This new feature protects the metadata from modification by unauthorized users. See **"Metadata Security" on page 12**.

- Performance
  - 64 bit file I/O

    InterBase 6.5 now supports the creation of database files greater than 4 GB on the Windows, Linux, and Solaris platforms. See **"64 bit I/O" on page 15**.
  - Enhanced Page Cache Management

    A new database page cache manager for better large memory support.
  - Cancel Execution of a DSQL Execution

    This feature modifies and existing DSQL API to request that a statement have its execution cancelled and interrupted. See **"Asynchronous Cancel" on page 15**

- New isc_config, ibconfig File Parameters

  The parameters CPU_AFFINITY, SWEEP_QUANTUM, USER_QUANTUM, and SWEEP_YIELD_TIME have been introduced in InterBase 6.5. See **"New isc_config, ibconfig File Parameters"**

- Language Enhancements

  InterBase 6.5 contains new language elements, including new keywords and the ROWS clause. The ROWS clause is used in conjunction with the SQL ORDER BY clause to provide a ranking capability for SQL queries. See **"The ROWS Clause for SQL Data Manipulation Statements" on page 19**. See

- XML Generation

  This feature allows InterBase customers to generate XML documents directly from InterBase without having to learn anything new about XML. We have added 2 new API calls isc_dsql_xml_fetch() and isc_dsql_xml_fetch_all(), which are used by client to create the XML file. See **"XML Generation"**

- JDBC

  - Removing Deprecated JRE API calls from InterClient

    This feature has been implemented so that now the InterClient code is up to date with the latest shipping JREs (1.3). The only requirement for the implementation of this functionality is the existence of Java 2 on the platform. This implementation allows InterBase to be stronger now that the code has moved to the most currently maintained JRE. See **"Removing Deprecated JRE API calls from InterClient"**

  - New JDBC methods

    There are two new JDBC methods. See **"New JDBC methods"**

Metadata Security

This new feature protects the metadata from modification by unauthorized users. This feature adds the SQL GRANT/REVOKE security framework to InterBase's system tables. The default access privileges for PUBLIC have been changed to allow SELECT only access; this will prevent ordinary users from corrupting the database by modifying the system metadata. The database owner, SYSDBA and operating system administrator (root on UNIX and Administrator on Windows) will continue to have all privileges to the metadata as well as the privilege to grant access to the metadata to anyone else.

The backup/restore GBAK utility has been modified to restore security privileges on system metadata. NOTE: GBAK has always backed up privileges to all tables, it just never restored them to the system tables. GBAK is also being modified to record the ODS major and minor version attributes of the database that was backed up in the backup file. Knowing that information will help the engine know what system tables need to have default privileges added when the database is restored.

ODS Changes and Usability Issues

Note The InterBase 6.5 release will change the ODS (On Disk Structure) from 10.0 to 10.1. This is a minor version update (10.0 to 10.1) to indicate that system tables now have default privileges for PUBLIC access.

When an ODS 10.0 (InterBase 6.0 engine) database is attached by InterBase 6.5, it will automatically add the default SQL privileges of SELECT-only to the system tables. InterBase 6.5 can still attach ODS 9 databases but it will NOT update them with SQL privileges; ODS 9 is supported as a bridge and, as much as, possible those databases should not be modified.

Three SQL scripts are included in ./examples/security: readmeta.sql, writemeta.sql and blindmeta.sql. These scripts can be run against databases with ISQL to make wholesale changes to a database's metadata PUBLIC privileges.

- Readmeta.sql corresponds to the default PUBLIC access privileges of the IBv6.5 engine. It can be used to return a database with customized metadate privileges back to the default.
- Writemeta.sql can be run to grant all metadata privileges to PUBLIC. This corresponds to the metadata access profile that existed before this project.
- Blindmeta.sql can be run to revoke all metadata privileges from PUBLIC. This prevents any PUBLIC user from querying the system tables. This includes 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 will make it difficult, if not impossible, for a general user to generate ad hoc queries against a database.

A user can develop a custom script to grant privileges to some individual users while denying access to others. GBAK will always preserve privileges on system tables across a database backup/restore.

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 metadata directly. The project of granting privileges on a database to allow a user to define metadata objects (tables, procedures, indices etc.) is not within the scope of this project.

Requirements and Constraints

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

As mentioned above, preventing all access to metadata by running blindmeta.sql against a database will prevent some utitities from correct operation when run by a general user. The special users (database owner, SYSDBA etc.) will still be able to run these utitilities in the normal fashion.

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. The InterBase 6.5 engine had to be modified so that it could access the system tables as an agent for users who had no privilege to do so directly. Thus an InterBase 6.5 developer who runs blindmeta.sql on an InterBase database will not be able to ship that database to customers with InterBase 6.0 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 with older InterBase runtimes.

Migration issues

The InterBase 6.5 engine will automatically install the default SQL privileges for PUBLIC on the metadata when attaching ODS 10.0 databases. Thus users who expect that all users should be able to write to database metadata will have to run writemeta.sql to restore that behavior.

It is not expected that this will be the general case but users may have to add those privileges if they have certain individuals needing that access.

Note ODS 10.1 databases created anew by InterBase 6.5 will naturally have the default SQL privileges for PUBLIC installed at create time.

Pre-ODS 10 databases will have to be migrated, as usual, by database backup/restore to get the metadata protection. If the security scripts are run against Pre-ODS 10 databases it is not clearly defined what the behavior will be. Readmeta.sql and writemeta.sql may function as expected but blindmeta.sql will prevent access to the database by normal users. In no case, will a backup/restore preserve metadata privileges across a restoration. The scripts would have to be manually run against the restored database for secure metadata. It is not advised that customers run these scripts from older kits (InterBase 6.0 and older) unless they have done full life-cyle testing of such a configuration to their satisfaction.

Going forward, InterBase will always be able to figure out the required system privileges to add to new InterBase system tables in future releases. Since GBAK has been changed to record the ODS major and minor versions of the database that was backed up, InterBase will always be able to compute the incremental privileges required to bring the restored database up-to-date. It will be able to this, while at the same time, preserving any customized system privileges the user may have made to their databases.

Performance

64 bit I/O

InterBase 6.5 now supports the creation of database files greater than 4 GB. In previous versions of InterBase to support files sizes greater than 4 GB one had to create multi-file databases. If one did not create a multi-file database and their database file exceeded the 4 GB limit file corruption could occur.

Asynchronous Cancel

The ability to cancel an executing DSQL statement is an often requested feature that is now included in InterBase 6.5. This feature modifies an existing DSQL API to request that a statement have its execution cancelled and interrupted.

The existing DSQL API *isc_dsql_free_statement()* has been modified to include a new option which requests that an executing statement be cancelled.

A third option of DSQL_cancel has been added to indicate that statement execution should be cancelled.

A successful return does not mean that statement execution was cancelled, only that the statement was marked for cancellation in the server.

Upon statement cancellation, the party which was executing the statement will be returned a status code of "isc_cancelled". The statement will not be able to execute further but will continue to return a status code of "isc_cancelled". However, the statement can be unwound and re-executed anew.

isc_dsql_free_statement()

Frees a statement handle and all resources allocated for it, or closes a cursor associated with the statement referenced by a statement handle.

Syntax

```
ISC_STATUS isc_dsql_free_statement (ISC_STATUS *status_vector,
isc_stmt_handle *stmt_handle, unsigned short option);
```

Parameter	**Type**	**Description**
status_vector	*ISC_STATUS* *	Pointer to the error status vector
stmt_handle	*isc_stmt_handle* *	Pointer to a statement handle previously allocated with *isc_dsql_allocate_statement()* or *isc_dsql_alloc_statement2()*; the handle returns an error in status_vector if it is NULL
option	*unsigned short*	Either DSQL_close, DSQL_drop, or DSQL_cancel

Description *isc_dsql_free_statement()* either frees a statement handle and all resources allocated for it *(option = DSQL_drop)*, or closes a cursor associated with the statement *(option = DSQL_close)*.

Note *isc_dsql_free_statement()* does nothing if it is called with an option value other than *DSQL_drop* or *DSQL_close*.

.DSQL_close

Call *isc_dsql_free_statement()* with the *DSQL_close* option to close a cursor after it is no longer needed, that is, after fetching and processing all the rows resulting from the execution of a query. A cursor need only be closed in this manner if it was previously opened and associated with stmt_handle by *isc_dsql_set_cursor_name()*. *DSQL_close* closes a cursor, but the statement it was associated with remainsavailable for further execution.

If you have used a cursor to perform updates or deletes on all the rows returned from the execution of a query, and you want to perform other update or delete operations on rows resulting from execution of the same statement again (possibly with different input parameters), follow these steps:

1. Close the cursor with *isc_dsql_free_statement()*.
2. Re-open it with *isc_dsql_set_cursor_name()*.
3. If desired, change the input parameters to be passed to the statement.
4. Re-execute the statement to retrieve a new select list.
5. Retrieve rows in a loop with *isc_dsql_fetch()* and process them again with *isc_dsql_execute_immediate()*.

.DSQL_drop

Statement handles allocated with *isc_dsql_allocate_statement()* must be released when no longer needed by calling *isc_dsql_free_statement()* with the *DSQL_drop* option. This option frees all resources associated with the statement handle, and closes any open cursors associated with the statement handle.

.DSQL_cancel

Statement handles allocated with *isc_dsql_allocate_statement()* must be released when no longer needed by calling .

Chapter 6 "Working with Dynamic SQL" in the API Guide provides additional information on how to use API dynamic SQL (DSQL) functions to handle dynamically created SQL statements for data definition and manipulation.

New isc_config, ibconfig File Parameters

- CPU_AFFINITY

  This parameter specifies the processors to be used by the InterBase server on a SMP machine. This parameter is only available on Windows. The default value is 1. The InterBase server will use the 1st CPU on the system by default. A process affinity mask is a bit vector in which each bit represents the processor on which the threads of the process are allowed to run.

Example

For processor 1:	1 (1st bit in bit vector)
For processor 2:	2 (2nd bit in bit vector)
For processors 1 & 2:	3 (11, 1st and 2nd bits in bit vector)
For processor 3:	4 (3rd bit in bit vector)
For processors 1,2 and 3:	7 (111 in bit vector)
For processors 2 and 3:	6 (110 in bit vector)

etc.

- SWEEP_QUANTUM

  This parameter 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. The default value is 10.

- USER_QUANTUM

  This parameter 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. The default value is 100.

- SWEEP_YIELD_TIME

  This parameter specifies the number of milli seconds the sweeper or the garbage collector thread sleeps. The default value is 1 milli-second. Note that this does not affect worker threads.

Note These configuration parameters are platform independent i.e., they are applicable on all platforms. Note that the SWEEP_QUANTUM and SWEEP_YIELD_TIME effects the sweeper/garbage collector thread hence, ideally when one is changed the other warrants change too. For example if one ends up changing the SWEEP_YIELD_TIME to a larger value say 1000 then it is advisable to change SWEEP_QUANTUM to a larger value also since the sweeper thread sleeps for a larger time so its good to do more work when its awake. This is particularly true when there are lots of back versions in the database. The default values are fine for most applications however there might be applications where in a bit of tuning may be needed for optimal performance.

Specifying a value of 0 or less will be ignored by the server and results in using the default value for that particular parameter. However there is no upper limit applied to these parameters. The server will report the values for each of these parameters in interbase.log file on startup.

Language Enhancements

New Keywords

InterBase 6.5 introduces the following new keywords:

PERCENT
ROWS
TIES

The ROWS Clause for SQL Data Manipulation Statements

The ROWS clause is used in conjunction with the SQL ORDER BY clause to provide a ranking capability for SQL queries. The ROWS clause is not part of the SQL standard but an InterBase SQL language extension to the provide similar functionality to InterBase's GDML FIRST clause. However, the InterBase SQL ROWS clause adds two extra capabilities useful for ranking that are not part of GDML FIRST. The ORDER BY ... ROWS syntax imposes an ordinal ranking on the result set. Hence, there is always a semantic meaning to the result set based on this ordering.

With such a ranking relation, it becomes meaningful to allow both SQL UPDATE and DELETE to take an ORDER BY clause for the purpose of restricting row modifications to a top / bottom percentage or number of rows.

The ROWS clause can also be used in isolation to restrict the number of rows returned from the result set. There is a positional, though random and unpredictable, semantic to the result set. It is used to partition a result set into subsets for retrieval purposes by applications of limited means.

The ROWS clause subsets the number of rows from the result set of an arbitrary table expression. The feature is used primarily by Web developers to parcel pieces of a larger result set from the Web server to a client browser. This type of Web application has a stateless interface with the database server and cannot gradually scroll the full result set via a cursor or download the entire result set into a cached dataset on the client. Rather the same SQL query is iteratively submitted to the database server but with a ROWS clause to specify which numbered rows from the full result set should be returned.

The ROWS clause can be used with the SQL SELECT, UPDATE and DELETE statements. It has a general syntax of ROWS <lower_value> [TO <upper_value>] [BY <step_value>] [PERCENT] [WITH TIES]. The <value_expressions> are positive numbers which specify how many rows to include in the ranking.

When ROWS... PERCENT is used, <value_expression> evaluates to a number less than or equal to 100. Conceptually, 1) the COUNT of the result set is calculated; 2) multiplied by <value_expression> and 3) divided by 100 with integer rounding per SQL conformance.

ROWS ... WITH TIES will always include rows with duplicate, equal rankings beyond the count of rows requested. Asking for the ORDER BY SCORES ROWS 3 WITH TIES with scores of 98, 96, 92, 92, 92, 87 will return 98, 96, 92, 92, 92 (5 rows and not 3) because tie scores have been requested.

Note The ORDER BY clause is required when using the TIES clause.

The BY <step_value> allows intermediate rows from the result set to be skipped over. It can be used to break the result set into percentage bands. For example, ROWS 100 BY 5 PERCENT would reduce an arbitrary number of rows to 20 rows with each row distanced from its neigboring row by 5 percent of the total rows.

Note The ROWS clause introduces three new InterBase SQL keywords: **ROWS**, **TIES**, and **PERCENT**.

Syntax:

```
SELECT <select_list>
FROM <table_list>
WHERE [GROUP BY clause]
[HAVING clause]
[UNION clause]
[ORDER BY clause]
[ROWS <lower_value> [TO <upper_value>] [BY <step_value>] [PERCENT] [WITH TIES]
```

An informal UPDATE syntax would be as follows:

```
UPDATE <table>
```

```
SET <column_assignmnent_list>
WHERE <search_predicate>
[ORDER BY clause]
[ROWS <lower_value> [TO <upper_value>] [BY <step_value>] [PERCENT] [WITH TIES]
```

An informal DELETE syntax has similar semantics to UPDATE:

```
DELETE FROM <table>
WHERE <search_predicate>
[ORDER BY clause]
[ROWS <lower_value> [TO <upper_value>] [BY <step_value>] [PERCENT] [WITH TIES]
```

Examples:

The following two isql statements selects the top 100 performing sales people. This example shows the ability for a Web developer to split the result set in half for display purposes.

```
SELECT SALESMAN, SALES_DOLLARS, SALES_REGION
FROM SALESMEN
ORDER BY SALES_DOLLARS DESC
ROWS 1 TO 50;
```

This second isql statement makes use of the TIES keyword.

```
SELECT SALESMAN, SALES_DOLLARS, SALES_REGION
FROM SALESMEN
ORDER BY SALES_DOLLARS DESC
ROWS 51 TO 100 WITH TIES;
```

The following isql statements selects the best 100 performing salesmen and rewards them with a 15% bonus:

```
UPDATE SALESMEN
SET SALES_BONUS = 0.15 * SALES_DOLLARS
ORDER BY SALES_DOLLARS DESC
ROWS 100 WITH TIES;
```

The following isql statements selects the worst 5% performing members of sales force and deletes them:

```
DELETE FROM SALESMEN
ORDER BY SALES_DOLLARS
ROWS 5 PERCENT WITH TIES;
```

Support for ROWS clause in GPRE

GPRE now supports this additional functionality of the ROWS clause. Currently this feature is only supported with the C / C++ programming language.

XML Generation

This feature allows InterBase customers to generate XML documents directly from InterBase without having to learn anything new about XML. We have added 2 new API calls isc_dsql_xml_fetch() and isc_dsql_xml_fetch_all(), which are used by client to create the XML file.

These new functions will be a part of a new client side DLL (dynamically linked library) called "ibxml.dll" on Windows, "ibxml.so" on Solaris and Linux. The new structures defined for these functions will be located in a new header file called "ibxml.h", the new prototype definations are included in the file "ibxml_proto.h", this header file also internall include "ibxml.h". The users wanting to use this feature will have to add this new library to their link path, and the new header file to their compiler include files.

The new function prototypes are:

- `int isc_dsql_xml_fetch(ISC_STATUS *status, isc_stmt_handle *stmt, unsigned short da_version, XSQLDA *sqlda, IB_XMLDA *ib_xmlda);`
- `int isc_dsql_xml_fetch_all(ISC_STATUS *status, isc_stmt_handle *stmt, unsigned short da_version, XSQLDA *sqlda, IB_XMLDA *ib_xmlda);`
- `int isc_dsql_xml_buffer_fetch(ISC_STATUS *status, isc_stmt_handle *stmt, USHORT da_version, char *buffer, int buffer_size, XSQLDA *sqlda, IB_XMLDA *ib_xmlda);`

Here the status is a pointer to the ISC_STATUS error status long array. The stmt is the pointer to the statement handle previously allocated using calls like isc_dsql_allocate_statement(). The da_version indicates the version of the extended XSQLDA passed to the function. sqlda is a pointer to the extended SQL descriptor Area (XSQLDA). ib_xmlda is a pointer to an initialized XML descriptor area (IB_XMLDA) explained in the section above.

In case of the first call the user still has access to the data in the cursor using the sqlda The new IB_XMLDA structure is explained below (this structure is contained in the ibxml.h file) structure like before.

```
typedef struct ib_xmlda

           {

char ISC_FAR  *xmlda_file_name      /* contains the file name which will
                                    be opened in append mode for writing
                                    the xml into. */

char ISC_FAR  *xmlda_header_tag;    /* points to the string which is
                                    printed out as the version tag*/

char ISC_FAR  *xmlda_database_tag;  /* points to the string which is
                                    printed out as the database tag in
                                    the xml file */

char ISC_FAR  *xmlda_table_tag;     /* points to the string which is
                                    printed out as the database tag in
                                    the xml file */

char ISC_FAR  *xmlda_row_tag;       /* points to the string which is
                                    printed out as the rowname tag in
                                    the xml file */

FILE          *xmlda_file_ptr;      /* points to a FILE structure which
                                    has been opened for writing the data
                                    in xml format, internal use only*/

char ISC_FAR  **xmlda_temp_buffer;  /* internal use only, used for
                                    storing the String array from
                                    fetch() */

ISC_STATUS    Xmlda_fetch_stat      /*this element holds the return
                                    value from the isc_dsql_fetch()
                                    call, it indicates if we have
                                    received all the records or if we
                                    have a error */

ULONG         xmlda_flags;          /* flags explained below */

ULONG         xmlda_more_data;      /* used by the buffer call to
                                    maintain status of the last record,
                                    0 if there is no more data 1 if there
                                    is more data has been fetched but
                                    not put out in the buffer */

ULONG         xmlda_temp_size;      /* internal use only, used to store
                                    the last records size */
USHORT        xmlda_version;       /* version of XMLDA */

USHORT        xmlda_status;        /* reserved for future use */

USHORT        xmlda_more;          /* this flag is used in conjunction
                                   with the buffered mode, if there is
                                   more XML data this is set */

USHORT        xmlda_version        /* version of XMLDA */

USHORT        xmlda_array_size;    /* internal use only */

SLONG         xmlda_reserved;      /* reserved for future use */

                                            } IB_XMLDA;

#define XMLDA_ATTRIBUTE_FLAG 0x01 /* if set the data is output as
attributes */

#define XMLDA_NO_NULL_DISPLAY_FLAG 0x02 /* if set the null data is not
displayed */(not implemementing right for 6.5)

#define XMLDA_NO_HEADER_FLAG 0x04 /* no additional header to be
displayed */
```

Before calling the functions the user is expected to set the following elements of the structure:

1. "xmlda_file_ptr" - This should be assigned to a previously opened FILE pointer. The file is assumed to be open for writing. The function will start writing from the location of the write pointer, the user can set or reset the write pointer to a specific location if he/she wishes. It is recommended that you do not modify this FILE structure once it is in use by the function. The user is responsible for closing the file.
2. "xmlda_version" - This should be currently set to 1, the function does not look for any value but can in the future.
3. "xmlda_status" - this should be set to 0 the by the user the first time *isc_dsql_xml_fetch()* is called. Does not have any effect in *isc_dsql_fetch_all()*, but is recommended that you set it to 0. Used internally by the XML functions to keep status.

Optional variables

1. "xmlda_header_tag" - Points to a character string which will be used as the xml header. If set to NULL the default header is printed, which is "<?xml version="1.0">"
2. "xmlda_database_tag" - Points to a character string which can be used in place of the Database tag (see example XML document below), if set to NULL the XML tag defaults to "Database".
3. "xmlda_table_tag" - Points to a character string which can be used in place of the Tablename tag (see example XML document below), if set to NULL the XML tag defaults to "Tablename".
4. "xmlda_row_tag" - Points to a character string which can be used in place of the Row tag (see example XML document below), if set to NULL the XML tag defaults to "Row".
5. "xmlda_flags" - Currently only 2 allowable values XMLDA_ATTRIBUTE_FLAG if this is set the XML document is generated as attributes instead of as tags, and XMLDA_NO_NULL_DISPLAY_FLAG (not implemented for 6.5) if set the null data is not displayed at all, the associated tags are also skipped.
6. "xmlda_file_ptr" - This should be assigned to a previously opened FILE pointer. The file is assumed to be open for writing. The function will start writing from the location of the write pointer, the user can set or reset the write pointer to a specific location if he/she wishes. It is recommended that you do not modify this FILE structure once it is in use by the function. The user is responsible for closing the file.

Setting the "XMLDA_ATTRIBUTE_FLAG" will result in all the output data being generated as attributes instead of elements. This flag will only affect the actual data generated from InterBase as the user can control all the other tags by inputting the desired attribute as tags.

Setting the "XMLDA_NULL_NO_DISPLAY_FLAG" will cause the API to skip generating rows for data that is null. The default behavior is to generate empty strings.

BLOBs and Arrays are not supported in this initial version.

In order to use the isc_dsql_xml_buffer_fetch(), you need to ensure that you have allocated at least a 1024 character buffer, which is passed to the function in the buffer argument, the buffer_size argument reports the size of this passed buffer. The functions returns either the size of characters written into the buffer wihtout the terminating null character, in case of error it returns -1 if there is not enough memory for it to continue, or -2 if you must have a larger buffer size. The function does not return incomplete headers, footers or records. Also the "xmlda_more_data" is set if the call should be made once again to get the complete XML buffer.

In order to make the calls work with Delphi, please use the regular POINTER type to hold space in the structure IBXMLDA for the FILE * type.

For code examples see: xml_api_buffer.c and xml_api_file.c in examples directory.

JDBC (InterClient 2.5)

Removing Deprecated JRE API calls from InterClient

This feature has been implemented so that now the InterClient code is up to date with the latest shipping JREs (1.3). The only requirement for the implementation of this functionality is the existence of Java 2 on the platform. This implementation allows InterBase to be stronger now that the code has moved to the most currently maintained JRE.

New JDBC methods

```
Class: DatabaseMetaData
Methods: supportsMixedCaseQuotedIdentifiers()
         storesMixedCaseQuotedIdentifiers()
         getIdentifierQuoteString()
```

Description: Added quoted identifier support for InterBase 6, dialect 3 onwards for the above methods in DatabaseMetaData class.

```
Class: ResultSet
Method: getBigDecimal()
```

Description: Added the getBigDecimal() functions without scale. The older and now depricated functions took 2 arguments getBigDecimal(int, int scale) or getBigDecimal(string, int scale), the new ones take getBigDecimal(int) and getBigDecimal(String).

CHAPTER

1

InterBase 6 features

This chapter lists features introduced in InterBase 6 and provides an overview and summary of each.

Bold green type indicates clickable links.

Feature overview

- IBConsole

  InterBase now provides a single, intuitive graphical user interface. From within this environment, you can configure and maintain an InterBase server, create and maintain databases on that server, and execute interactive SQL. IBConsole replaces the earlier Server Manager and InterBase Windows ISQL GUI environments. See **"IBConsole" on page 29**.

- SQL Dialects

  You now specify a dialect for both clients and databases to specify whether double quotes, large exact numerics, and the DATE datatype are interpreted as InterBase 5 or InterBase 6 features. See **"SQL Dialects" on page 31**.

- Read-only databases

  Databases can now be set to read-only mode for security purposes or for distribution on read-only media. See **"Read-only databases" on page 33**.

- Language Elements

  InterBase 6 contains several new and upgraded language elements, including new keywords, delimited identifiers, large exact numerics, additions to the ALTER DOMAIN syntax, and a new ALTER COLUMN clause of ALTER TABLE to change the name, position, and datatype of columns in tables. InterBase 6 includes three datatypes for handling date and time: TIMESTAMP, DATE, and TIME. InterBase 6 also provides the new EXTRACT() function for extracting date and time information, as well as well as CURRENT_DATE, CURRENT_TIME, and CURRENT_TIMESTAMP functional operators, and three new date/time literals. See **"Language elements" on page 33** for more information.

- Command-line tool enhancements

  InterBase 6 contains enhancements to the gbak, gfix, gpre, and isql utilities:

  - The gbak utility can back up to and restore from multiple files, perform server-side backups using InterBase's new Service Manager, and can be accessed from within IBConsole. See **"New gbak functionality" on page 39.**
  - The gfix and gpre utilities now allow you to set the database dialect. This functionality is also available from within IBConsole. See **"gpre and gfix enhancements" on page 40**.
  - You can now specify the dialect for isql at the command line when you invoke it. In addition, you can use the SET SQL DIALECT *n* statement to change the dialect of the isql client during a session.You can run isql sessions from within IBConsole. See **"isql enhancements" on page 39**.

- API Enhancements

  InterBase 6 contains several API enhancements, including:

  - **Services API**. Services API library functions allow you to programmatically monitor and control InterBase servers and databases. See **"Services API" on page 40**.
  - **Install API**. Install API library functions allow you to integrate the installation of your own product with the deployment of an embedded copy of InterBase. The InterBase portion of the install can be *silent*: it does not display billboards and need not require intervention from the end user. See **"Install API" on page 41**.
  - **Licensing API**. Licensing API library functions allow you to check, add, remove, and view certificate ID and key pairs (authorization codes). See **"Licensing API" on page 41**.
  - **InterBaseExpress™ (IBX)**. InterBase 6 includes a version of IBX that addresses InterBase 6 features, including the new Install API and Licensing API. This IBX version is a superset of the version included with Delphi 5. See **"InterBase Express for Delphi and C++ Builder users" on page 42**.

- **Status vector**. The status vector now provides the error code, type of message, and where the message was generated, and allows the use of SQL warnings and informational messages. See **“Status vector and warning messages” on page 41**.

- New connectivity tools

  The ODBC driver provided with earlier versions of InterBase has been replaced with EASYSOFT, an InterBase-specific server that can access databases on multiple servers using existing operating system and network infrastructures, while increasing network speed and security.See **“New connectivity tools” on page 42** for more information.

- Arithmetic Operations

  InterBase 6 contains several changes to arithmetic operations, SQL functions, generators, and numeric input and exponents. See **“Arithmetic operations” on page 43** for more information.

- Changes to System Tables

  InterBase 6 contains enhancements to the RDB$FIELDS and RDB$FUNCTION_ARGUMENTS tables. For a descriptions of these enhancements, see **“Changes to system tables” on page 46**.

- Replication

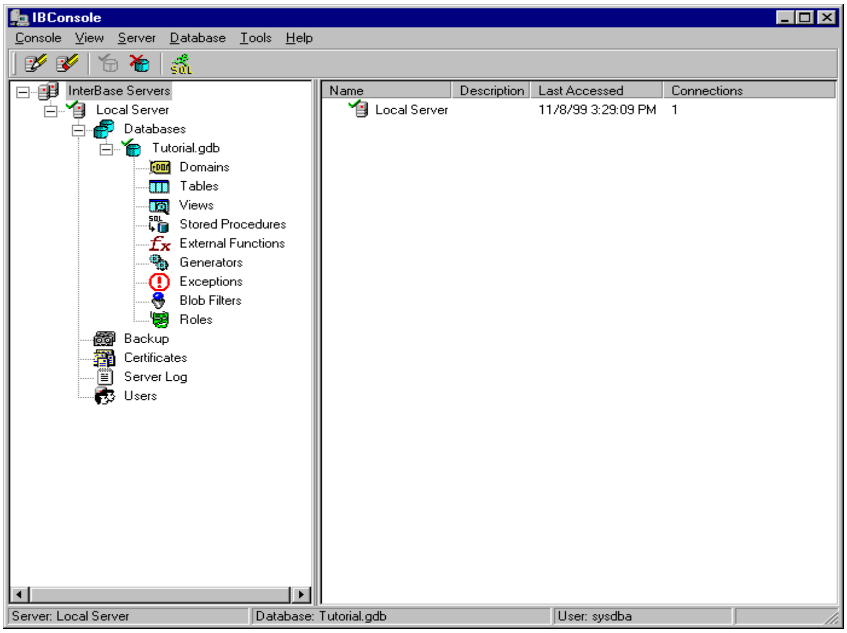
  InterBase 6 contains IBReplicator, which facilitates replication and synchronization between multiple InterBase databases that have similar structure. See **“Replication” on page 47**.

IBConsole

InterBase provides an integrated graphical user interface called IBConsole. Using IBConsole, you can configure and maintain an InterBase server, create and administer databases on the server, run interactive SQL, manage users, and administer security.

IBConsole is a GUI front end for InterBase’s command-line tools. It replaces the Server Manager and InterBase Windows ISQL interfaces found in earlier versions of InterBase. IBConsole runs on Windows, but can manage databases on any InterBase server on the local network, and on UNIX, Linux, and NetWare.

FIGURE 0.1 IBConsole



Name	Description	Last Accessed	Connections
Local Server		11/8/99 3:29:09 PM	1

Server: Local Server | Database: Tutorial.gdb | User: sysdba

You can use IBConsole to:

- Perform data entry and manipulation
- Configure and maintain a server
- Enter and execute interactive SQL
- Manage server security
- Backup and restore or sweep a database
- View database and server statistics
- Validate the integrity of a database
- Recover “in limbo” transactions

For details on using IBConsole, see the *Operations Guide*.

SQL Dialects

InterBase 6 introduces the concept of dialects to allow users to move ahead with new features that are incompatible with older versions of InterBase: delimited identifiers, large exact numerics, and the SQL DATE, TIME, and TIMESTAMP datatypes.

Database and client dialects

As InterBase moves forward in complying with SQL standards, some new features and usages are incompatible with older usage. Dialects assist in this transition. Dialect 1 guarantees compatibility with older databases and clients. Dialect 3 allows full access to new features. There is also a dialect 2 available for clients. It is a diagnostic mode only.

Features that behave differently in dialect 1 and dialect 3 are called *transition features*. The transition features are:

· Anything delimited by double quotes

· Date/time datatypes

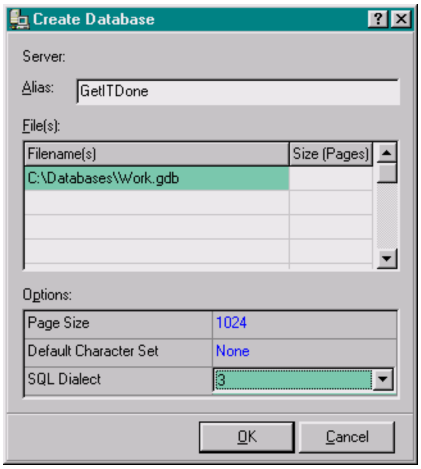
· Large exact numerics (DECIMAL and NUMERIC datatypes with precision greater than 9)

Both clients and databases must be assigned a dialect. Databases are created in dialect 3 by default. The isql client takes on the dialect of the database to which it is attached unless you specify a different dialect.

IMPORTANT To display the dialect of both client and attached database, use ISQL from either IBConsole or the command line to issue the new SHOW SQL DIALECT statement.

SETTING THE DATABASE DIALECT IN IBCONSOLE

- In IBConsole, you set the dialect for a database you are creating in the Create Database dialog:



Alias: GetITDone

Filename(s)	Size (Pages)
C:\Databases\Work.gdb	

Page Size	1024
Default Character Set	None
SQL Dialect	3

CHANGING THE DATABASE DIALECT IN IBCONSOLE

In IBConsole, select the database in the Tree pane and choose Properties to display the Database Properties dialog. 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.

SETTING/CHANGING THE CLIENT DIALECT IN IBCONSOLE

To set or change the dialect of a client in IBConsole:

1. Access the **Tools** menu.
2. From the **Tools** menu, select the **Interactive SQL** entry.
3. From the **Interactive SQL Edit** menu, select **Options**.
4. In the **Options** tab, click the number next to **Client Dialect** and select a dialect from the dropdown list.

For further information on using IBConsole and **isql** to set client and database dialects, see the *Operations Guide*.

Read-only databases

You can now change InterBase databases to *read-only mode*. This provides enhanced security for databases by protecting them from accidental or malicious updates. Databases are always in read-write mode at creation time. You can change any InterBase 6 database, regardless of dialect, to read-only mode using gbak or gfix.

Both gbak and gfix now have a -mode option, which has a value of either read_only or read_write. The new syntax is as follows:

```
gbak -create -mode read_only old_database.gbk new_database.gdb
gfix -mode read_only database.gdb
```

Both of these operations require exclusive access to the database.

If an InterBase 6 client tries to set an InterBase 5 or older database to read-only, the server silently ignores the request. No error or warning is issued.

An InterBase 5 or older client can select from a read-only database. However, these older clients cannot distinguish between a read-only and read-write database. If an older client attempts to do anything other than select from an read-only database, the attempt fails with an error.

For information on creating a read-only database, see “Read-write and read-only databases” in the *Operations Guide*.

Language elements

New keywords

InterBase 6 introduces the following new keywords:

COLUMN	SECOND
CURRENT_DATE	SQL
CURRENT_TIME	TIME
CURRENT_TIMESTAMP	TIMESTAMP
DAY	TYPE
EXTRACT	WEEKDAY
HOUR	YEAR
MINUTE	YEARDAY
MONTH	

Note To use a keyword as an object identifier (such as a column name), in dialect 3 you must enclose the identifier in double quotes ("). These keywords are described further in the following sections.

If you are upgrading from a previous version of InterBase, see the Migration chapter in *Getting Started* for more information on how keywords are affected by the new dialects.

SQL delimited identifiers

InterBase now supports SQL delimited identifiers. This means that InterBase object names can:

- Be a keyword
- Use spaces, except for trailing spaces
- Use non-ASCII characters
- Be case-sensitive

SQL delimited identifiers are permitted only in InterBase 6 clients and databases using dialect 3. In InterBase 6 clients and databases using dialect 3, a string constant is delimited by single quotes, and an SQL delimited identifier is delimited by double quotes. Because the quotes delimit the boundaries of the name, the possibilities for object names are greatly expanded from previous versions of InterBase.

Previous versions of InterBase allowed the use of both single and double quotes around string literals. In InterBase 6, double quotes are used only for delimited identifiers, and strings must be enclosed in single quotes. If you are thinking of migrating to dialect 3 or to any future versions of InterBase at any time in the future, you should always use single quotes to delimit strings. Refer to the Migration chapter of *Getting Started* for more information.

New On-Disk Structure (ODS)

InterBase 6 databases stores data using the On-Disk Structure (ODS) version 10, which is required to use delimited IDs, exact numerics, and SQL DATE, TIME, and TIMESTAMP datatypes.

IMPORTANT InterBase 6 servers can connect only to ODS version 10 databases, and InterBase 5 servers can connect only to databases with an ODS version of 9 or 8. See the section on Migration in *Getting Started* for more information on how to make existing tables compatible with this new ODS.

SQL DATE, TIME, and TIMESTAMP

The CURRENT_DATE, CURRENT_TIME, and CURRENT_TIMESTAMP functional operators return the date and time values based on the moment of execution of an SQL statement using the server's clock and time zone. It is no longer necessary to cast TODAY or NOW as DATE to obtain the current date, time, or timestamp. CURRENT_DATE, CURRENT_TIME, and CURRENT_TIMESTAMP can be specified as the default clause for a domain or column definition.

For a single SQL statement, the same value is used for each evaluation of CURRENT_DATE, CURRENT_TIME, and CURRENT_TIMESTAMP within that statement. This means that if multiple rows are updated, as in the following statement, each data row will have the same value in the aTime column.

```
UPDATE aTable SET aTime = CURRENT_TIME;
```

Similarly, if row buffering occurs in a fetch via the remote protocol, then the CURRENT_TIME is based on the time of the OPEN of the cursor from the database engine, and not the time of delivery to the client.

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.

Migration

The old InterBase DATE datatype, which contains both date and time information, is being replaced with the SQL92 standard TIMESTAMP, DATE, and TIME datatypes in dialect 3.

- 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, all the DATE columns and domains are automatically restored as TIMESTAMP.
- 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.

TIMESTAMP is a 64-bit datatype and DATE and TIME are 32-bit datatypes.

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. The primary migration problem will exist in the source code of application programs that use the InterBase 5 DATE datatype. In InterBase 6, the DATE keyword represents a date only datatype, while in InterBase 5 DATE represents a date and time datatype. For more information on this and other migration issues, refer to the Migration chapter of *Getting Started*.

EXTRACT() function

The EXTRACT() function extracts date and time information from databases.

EXTRACT() has the following syntax:

```
EXTRACT (part FROM value)
```

The value passed to the EXTRACT() expression must be a DATE, TIME, or TIMESTAMP datatype. Extracting a part that doesn't exist in a datatype results in an error. For example, the following expression would fail:

```
EXTRACT (YEAR FROM aTime)
```

For more information, refer to EXTRACT() in the *Language Reference Guide*.

CAST() function

You can use the CAST() function in SELECT statements to translate between date/time datatypes and various character-based datatypes.

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*.

Note that there are some differences from behavior in past versions of InterBase:

- Casting DATE to string results in YYYY-MM-DD where" M"M 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 DATE supports YYYY-MM-DD and other unambiguous input formats such as YYYY-Mon-DD. YYYY-DD-MM is not supported. Two-digit years are not allowed. In previous versions, input format was MM-DD-YYYY HH:mm:SS.hundreds or DD.MM.YYYY HH.mm:SS.hundreds. Two-digit years were interpreted as being within 100 years of the current year.

Changing column and domain definitions

You can now alter the name, datatype, and position of a column using the ALTER COLUMN clause of the ALTER TABLE statement. Extensions to the ALTER DOMAIN statement allow you to alter the name and datatype of a domain. This functionality is available in InterBase 6 dialects 1 and 3.

Altering domains

The ALTER DOMAIN statement has new options that allow you to change the name and datatype of a domain. For example, to change the name of a domain:

```
ALTER DOMAIN domain1 TO domain2;
```

Altering columns in tables

The new ALTER COLUMN clause of the ALTER TABLE statement allows you to change:

- The datatype of a field
- The name of a field
- The position of a field with respect to the other fields

For example, to change the name of a column:

```
ALTER TABLE table1 ALTER COLUMN field1 TO field2;
```

To change the datatype of a column:

```
ALTER TABLE table1 ALTER COLUMN field1 TYPE char(20);
```

To change the position of a field:

```
ALTER TABLE table1 ALTER COLUMN field1 POSITION 4;
```

The ALTER COLUMN clause of ALTER TABLE and the TYPE clause of ALTER DOMAIN do not allow you to make datatype conversions that could lead to data loss. For example, they do not allow you to change the number of characters in a column to be less than the largest value in the column.

Table **0.1** lists valid datatype conversions. The columns list the target datatype and rows are the source datatype.

	Blob	Char	SQL Date	Decimal	Double	Float	Integer	Numeric	Timestamp	Time	Smallint	Varchar
Blob												
Char		X										X
SQL Date		X	X						X			
Decimal		X		X				X				X
Double		X			X	X						X
Float		X			X	X						X
Integer		X		X	X		X	X				X
Numeric		X						X				X
Timestamp		X	X						X	X		
Time		X							X	X		
Smallint		X		X	X	X	X	X			X	X
Varchar		X										X

TABLE 0.1 Possible datatype conversions using ALTER COLUMN and ALTER DOMAIN

Converting a numeric datatype to a character type requires a minimum length for the character type as listed in table **0.2**:

Datatype	**Minimum length for converted character type**
Decimal	20
Double	22
Float	13
Integer	11
Numeric	22
Smallint	6

TABLE 0.2 Minimum character lengths for numeric conversions

Command-line tool enhancements

New gbak functionality

The InterBase 6 gbak command incorporates the functionality of the version 5 gsplit utility, allowing the database owner or SYSDBA to back up to and restore from multiple files.

gbak now allows you to set databases to read-only with the -mode read_only switch. Refer to **"Read-only databases" on page 33** for more information.

You can use gbak's new -service switch to perform server-side backups. In this mode, gbak uses the new Services API and performs the backups on the server, incurring significantly less network traffic. Previously, backups were all performed on the client.

When using the -service switch, make sure that all path names to databases and backup files must be given relative to the server.

When backing up without using the Services API, backups are performed on the client platform where gbak is running. All pathnames to databases and backup files must be given relative to the client.

For more information on the gbak command, see the *Operations Guide*.

isql enhancements

- At the command line, you can invoke isql with the -sql_dialect *n*, where *n* is 1, 2, or 3.
- Within a s SQL ession, use SET SQL DIALECT *n*, where *n* is 1, 2, or 3. This overrides the dialect set in the command line when isql is invoked.
- If you do not specify a dialect in one of these two ways, isql acquires the dialect of the database to which it is attached.

When you create a database with a client such as command-line isql, the database initially has the dialect of the client. For example, you can create a dialect 3 database in either of the following ways:

Specifying the dialect on the command line:

```
C:\> isql -dialect 3
CREATE DATABASE C:\databases\work.gdb;
```

Specifying the dialect during a session:

```
C:\> isql
```

```
SET SQL DIALECT 3;
CREATE DATABASE C:\databases\work.gdb;
```

If you create a database with isql and have not specified an isql dialect, isql creates a dialect 3 database. You can also use API functions to set and query dialects:

- *isc_dpb_set_db_SQL_dialect*

  Use with the *isc_attach_database*() API function to overwrite the database's dialect.
- *isc_info_db_SQL_dialect*

  Use with the *isc_database_info*() API function to query the database's dialect.

gpre **and** gfix **enhancements**

Use the gpre command line option -sql_dialect *n*, where *n* is 1, 2, or 3, to set the dialect of a database or client. EXEC SQL SET SQL DIALECT *n*, where *n* is 1, 2, or 3, sets the dialect of a database or client. SET SQL DIALECT overrides any value set at the command line.

Use the gfix command line option -sql_dialect *n*, where *n* is 1 or 3, to set the dialect of an ODS 10 version database.

API enhancements

Services API

The InterBase 6 Services API allows you to write applications that monitor and control InterBase servers and databases. Tasks that you can perform with this API include:

· Performing database maintenance tasks such as database backup and restore, shutdown and restart, garbage collection, and scanning for invalid data structures

· Creating, modifying, and removing user entries in the security database

· Administering software activation certificates

· Requesting information about the configuration of databases and the server

The Services API is a group of functions in the InterBase client library (gds32.dll on Windows, libgds.a on UNIX/Linux). The Services API family consists of the following four functions:

- *isc_service_attach*( ) initiates a connection to a specified Services Manager
- *isc_service_start*( ) invokes a service task

- *isc_service_query*( ) requests information or task results from the Services Manager
- *isc_service_detach*( ) disconnects from the Services Manager

The features that you can exercise with the Services API include those of the command-line tools gbak, gfix, gsec, gstat, and iblicense. The Services API can also perform other functions that are not provided by these tools.

For more information about the Services API, refer to "Working with Services" in the *API Guide.*

Install API

InterBase provides developers with a new group of functions that facilitate the process of silently installing InterBase as part of an application install on the Win32 platform. In addition, it allows you to interact with users if desired, to gather information from them and to report progress and messages back to them.

For more information refer to "Using the Install and Licensing APIs" in the *Developer's Guide*.

Licensing API

The InterBase server functionality must be activated by installing *authorization codes* that are provided on software activation certificates obtained from InterBase. Each authorization code consists of a Certificate ID and Certificate key. You can activate the server as part of your install by using functions provided in the InterBase License API. If you do not activate the server as part of the install, it will be inactive until the end user provides authorization codes using IBConsole.

For more information refer to "Using the Install and Licensing APIs" in the *Developer's Guide*.

Status vector and warning messages

The InterBase status vector is a mechanism that holds information about the current operation, where it is accessed via API calls. In previous versions of InterBase, the status vector contained only the error code, but in InterBase 6 it also contains the information about the source of the error message, and the error message type, either error, warning, or informational. Warning and informational messages do not impede normal client/server operations, but may advise the client of a problem that needs investigation.

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
- API calls which will be replaced in future versions of the product
- Pending database shutdown

By default, isql displays any message returned in a status vector, even if no error occurred. To display warning messages if and only if an error occurs, specify -nowarnings on the command line when starting isql.

InterBase Express for Delphi and C++ Builder users

Borland Delphi and C++ builder users can now use the InterBase Express™ (IBX) components to build InterBase database applications without the overhead of using the Borland Database Engine (BDE). IBX accesses the InterBase API directly, allowing increased speed and control within InterBase applications.

The version of IBX that comes with Delphi 5 addresses only InterBase 5 features. The IBX version that is included with InterBase 6 addresses all InterBase 6 features, using calls to the new Service API, Install API, and Licensing API, as well as the newest InterBase API.

The InterBase 5 version of IBX provides one additional tab in Delphi, labelled *InterBase*, that contains the IBX components for InterBase 5. The InterBase 6 version of IBX provides two tabs in Delphi: the InterBase tab is the same as in version 5 IBX; in addition there is an *InterBase Admin* tab. The InterBase Admin tab contains components that address the Services API, Install API, and Licensing API. It contains configuration, backup, restore, licensing, statistics, logging, and install, and uninstall components. The InterBase Admin tab is found at the extreme right of the tabs in Delphi 5. You will have to scroll to find it.

To install the InterBase 6 version of IBX, ensure that Delphi 5 is installed, Then display the InterBase Launcher and choose InterBase Express.

New connectivity tools

InterBase now ships with a series of standards-based client/server middleware products that allow clients to access database on multiple servers using existing operating system and network infrastructures, while increasing network speed and security. The thin-client architecture facilitates installation and administration, and prevents client conflict with multiple DLLs, databases, and network stacks.

Arithmetic operations

All arithmetic operations in dialect 3 produce the results called for by the SQL92 standard, which in some cases are not the same results produced by previous versions of InterBase. The most notable problems when migrating to InterBase 6 involve the division operator and the AVG() function, which also implies division, with exact numeric operands.

SUM and AVG

SUM() and AVG() return an exact numeric type if the subject row has an exact numeric type and the scaled sum fits in 64 bits; otherwise an exception is raised.

SUM() and AVG() operations on an exact numeric column return type NUMERIC(18,*S*) or DECIMAL(18,*S*), where *S* is the scale of the column.

If the datatype of the column is an approximate numeric (FLOAT, REAL, or DOUBLE PRECISION), SUM() and AVG() are computed using floating point arithmetic.

MIN and MAX

MIN() and MAX() operations on an exact numeric column return an exact numeric result having the same precision and scale as the column.

Addition and subtraction

If both operands are exact numeric, then adding or subtracting the operands produces an exact numeric with a precision of 18 and a scale equal to the larger of the two. For example:

```
CREATE TABLE t1 (n1 NUMERIC(16,2), n2 NUMERIC(16,3));
INSERT INTO t1 VALUES (12.12, 123.123);
COMMIT;
```

The following query returns the integer 135.243. The largest scale of the two operands is 3; therefore, the scale of the sum is 3.

```
SELECT n1 + n2 FROM t1;
```

Similarly, the following query returns the integer –111.003:

```
SELECT n1 – n2 FROM t1;
```

If either of the operands is approximate numeric (FLOAT, REAL, or DOUBLE PRECISION), then the result is DOUBLE PRECISION.

Multiplication

If both operands are exact numeric, then multiplying or dividing the operands produces an exact numeric with a precision of 18 and a scale equal to the sum of the scales of the operands. For example:

```
CREATE TABLE t1 (n1 NUMERIC(16,2), n2 NUMERIC(16,3));
INSERT INTO t1 VALUES (12.12, 123.123);
COMMIT;
```

The following query returns the integer 1492.25076 because n1 has a scale of 2 and n2 has a scale of 3. The sum of the scales is 5.

```
SELECT n1*n2 FROM t1
```

If one of the operands is approximate numeric (FLOAT, REAL, or DOUBLE PRECISION), then the result is DOUBLE PRECISION.

Division and AVG

If both operands are exact numeric, then multiplying or dividing the operands produces an exact numeric with a precision of 18 and a scale equal to the sum of the scales of the operands. If at least one operand of a division operator has an approximate numeric type (FLOAT, REAL, or DOUBLE PRECISION), then the result is DOUBLE PRECISION.

For example, in the following table defined in InterBase 6, division operations produce a variety of results:

```
CREATE TABLE t1 (i1 INTEGER, i2 INTEGER, n1 NUMERIC(16,2),
   n2 NUMERIC(16,2));
INSERT INTO t1 VALUES (1, 3, 1.00, 3.00);
COMMIT;
```

The following query returns the integer 0 because each operand has a scale of 0, so the sum of the scales is 0:

```
SELECT i1/i2 FROM t1
```

The following query returns the NUMERIC(18,2) value 0.33, since the sum of the scales 0 (operand 1) and 2 (operand 2) is 2.

```
SELECT i1/n2 from t1
```

The following query returns the NUMERIC(18,4) value 0.3333, since the sum of the two operand scales is 4.

```
SELECT n1/n2 FROM t1
```

In InterBase 5 and older, any of the above division operations return the DOUBLE PRECISION value 0.3333333333333333.

To obtain an InterBase 5 result when using InterBase 6, alter your query to cast at least one of the operands into an approximate type. For example,

```
SELECT i1/cast(i2 as double precision) from t1;
```

Generators

Any value that can be stored in a DECIMAL(18,0) can also be specified as the value on a SET GENERATOR statement.

Generators return a 64-bit value, and only wrap around after 2^{64} invocations (assuming an increment of 1) rather than 2^{32} as in InterBase 5. Use an ISC_INT64 variable to hold the value returned by a generator.

Numeric input and exponents

Any numeric string in dsql or isql that can be stored as a DECIMAL(18,*S*) is evaluated exactly, without the loss of precision that might result from intermediate storage as a DOUBLE. A numeric string is recognized by the dsql parser as a floating point value only if it contains an “e” or “E” followed by an exponent, which may be zero. For example, dsql recognizes 4.21 as a scaled exact integer, and passes it to the engine in that form. On the other hand, dsql recognizes 4.21E0 as a floating point value.

Large exact numerics

In dialect 3, InterBase 6 conforms with the SQL92 standard by storing NUMERIC and DECIMAL datatypes with 10 to 18 digits of precision as 64-bit integers (INT64 datatype). InterBase has always implmented NUMERIC and DECIMAL datatypes with precision less than 10 as exact numerics, but those with precision of 10 thorugh 15 were implmented as DOUBLE PRECISION. Now, NUMERIC and DECIMAL datatypes are all stored as exact numerics. They are 16, 32, or 64 bit, depending on the precision. NUMERIC and DECIMAL datatypes with precision greater than 9 are referred to as “large exact numerics” in this discussion.

- These new 64-bit integer types ar available in all contexts where datatypes are defined or used.
- NUMERIC and DECIMAL datatypes with a precision of 9 and scale *S* that caused arithmetic error messages in InterBase 5 return correct 64-bit results in InterBase 6.

- When an arithmetic operation on exact numeric types overflows, InterBase 6 reportws an overflow error, rather than returning an incorrect value.
- If one operand is an approximate numeric, then the result of any dyadic operation (addition, subtraction, multiplication, division) is DOUBLE PRECISION.
- Any value that can be stored in a DECIMAL(18,*S*) can also be specified as the default value for a column or a domain.

Changes to system tables

Both the RDB$FIELDS table and the RDB$FUNCTION_ARGUMENT table contain a new column, RDB$FIELD_PRECISION, to store the precision for NUMERIC and DECIMAL datatypes. The RDB$FIELD_PRECISION column is of type SMALLINT.

In the RDB$FIELDS and RDB$FUNCTION_ARGUMENTS tables, the RDB$FIELD_SUB_TYPE column now holds the subtype for NUMERIC and DECIMAL datatypes. IF RDB$FIELD_TYPE is 7 (SMALLINT), 8 (INTEGER), or 16 INT64), then RDB$FIELD_SUB_TYPE has the following possible values:

0 or NULL	RDB$FIELD_TYPE
1	NUMERIC
2	DECIMAL

For example, if a column is defined as DECIMAL(13,4), the row in RDB$FIELDS has the following values:

	RDB$FIELD_TYPE	RDB$FIELD_SUB_TYPE	RDB$FIELD_PRECISION	RDB$FIELD_SCALE
Value	16	2	13	-4
Meaning	Stored as INT64	Type is DECIMAL	Precision is 13	Scale is 4

Migration

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, 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 dialect 3, when you create a NUMERIC or DECIMAL column with a precision greater than 9, data in it is automatically stored as an INT64 exact numeric.

For more information, refer to the Migration chapter in *Getting Started.*

Replication

IBReplicator is a component of the InterBase 6 product that facilitates replication and synchronization between multiple InterBase databases that have similar structure. IBReplicator synchronizes databases by copying changes of the entire database or a subset of the database, as you specify. Replicant databases can reside on different servers or on the same server. This product component includes a replication server executable for each respective InterBase server platform, and also a Windows graphical tool for configuring and invoking data replication. You can find documentation for configuring and using IBReplicator in the "Data Replication" chapter of the *Operations Guide*.

The version of IBReplicator for InterBase 6 contains the following new features:

- A new command, **Remove System Objects**, removes all triggers and other system objects from a source database without having to remove target databases from the schema.
- All configuration databases now have an option to change/enter the Current Schema Number. This is required in complex replication environments; see IBReplManager.hlp for details.
- The Replication Server can automatically map all tables and fields on the source database to tables and fields of the same name on the target database.

This release also fixes bugs related to the **Create System Objects** command, recovery from lost/completed connections, replicated deletes with NUMERIC and INT64 primary keys.

APPENDIX

A Error Information

This chapter lists error messages introduced in InterBase 6 and provides descriptions and workarounds, if available, for known bugs.

New error messages

Below are tables listing new error messages for this release of InterBase. These error messages are sorted by feature, for example, dialects error messages or gbak error messages. InterBase 6 also has the following new global error messages, in addition to those listed below.

Error Code	Number	SQL Code	Error message text
isc_ext_readonly_err	335544651L	-816	I/O error for file <string> -Error while trying to write to file -Cannot insert because the file is read-only or is on a read only medium.
isc_ext_file_delete	335544786L	-901	Cannot delete rows from external files.
isc_ext_file_modify	335544787L	-901	Cannot update rows in external files.

TABLE 0.1 InterBase 6 error messages

The following new error messages are associated with the use of dialects:

Error Code	Number	SQL Code	Message Text
isc_ddl_not_allowed_by_db_sql_dial	335544793	-817	Metadata update statement is not supported by the current database SQL dialect %d
isc_sql_dialect_datatype_unsupport	335544796	-804	Client SQL dialect %d does not support reference to %s datatype

TABLE 0.2 Dialect error message

The following new error messages are associated with delimited identifiers:

Error Code	Number	SQL Code	Message Text
isc_ddl_not_allowed_by_db_sql_dialect	335544793	-817	Metadata statement is not allowed by the current database SQL dialect %d
isc_gfix_opt_SQL_dialect	335741039	None	Set database dialect *n*
isc_gfix_sql_dialect	112	None	SQL dialect must be one of %s
isc_invalid_string_constant	335544763	-104	Feature not supported

TABLE 0.3 Delimited ID error messages

The datetime feature introduces the following new error messages:

Error Code	Number	SQL Code	Message Text
isc_extract_input_mismatch	335544789	-105	Specified EXTRACT part does not exist in input datatype
isc_datype_notsup	335544801	-901	Datatype not supported for arithmetic
isc_expression_eval_err	335544606	-902	Expression evaluation not supported/old.

TABLE 0.4 Datetime error messages

Alter column introduces the following new error messages:

Error Code	Number	SQL Code	Message Text
isc_dyn_char_fld_too_small	336068816	-604	New size specified for column %s must be greater than %d characters.
isc_dyn_dependency_exists	336068814	-616	Column %s from table %s is referenced in %s.
isc_dyn_domain_name_exists	336068812	-612	Cannot rename domain %s to %s. A domain with that name already exists.
isc_dyn_dtype_conv_invalid	336068818	-688	Cannot change datatype for column %s from a character type to a non-character type.
isc_dyn_dtype_invalid	336068815	-688	Cannot change datatype for column %s. Changing datatype is not supported for BLOB or ARRAY columns.
isc_dyn_field_name_exists	336068813	-612	Cannot rename column %s to %s. A column with that name already exists.
isc_dyn_invalid_dtype_conversion	336068817	-604	Cannot change datatype for column %s. Conversion from base type%s to base type %s is not permitted.

TABLE 0.5 Alter column error messages

The read-only database feature introduces the following new error messages:

Error Code	Number	SQL Code	Message Text
isc_read_only_database	445	817	Attempted to update read-only database
isc_gfix_opt_mode	109	901	-mode “read_only” or “read_write”
isc_gfix_mode_req	110	901	“read_only” or “read_write” required
isc_gbak_opt_mode	278	901	-mode access “read_only” or “read_write”
isc_gbak_mode_req	279	901	“read_only” or “read_write” required

TABLE 0.6 Read-only database error messages

The gbak feature introduces the following new error messages:

Error Code	Number	SQL Code	Message Text
isc_gbak_page_size_missing	336330754L	-901	size specification either missing or incorrect for file <string>
isc_gbak_file_outof_sequence	336331015L	-901	file <string> out of sequence
isc_gbak_join_file_missing	336331016L	-901	can't join -- one of the files missing
isc_gbak_stdin_not_supptd	336331017L	-901	standard input is not supported when using join operation
isc_gbak_stdout_not_supptd	336331018L	-901	standard output is not supported when using split operation

TABLE 0.7 gbak error messages

Fixed bugs, known bugs, and workarounds

Here are brief descriptions of the bugs fixed in the 6.5 release of InterBase. For more information, search the Borland community site at http://search.borland.com

Bug Number	Description
58979	Creating and dropping foreing keys requires exclusive access
59034	Continuous sweeps when sweep interval hit and OIT stuck
59043	InterClient: DDL Statements invalidate live preparedStatement handles
60099	Minor errors in UDF exception messages
60237	#include <stdio.h> is in all 'c' code examples
60470	File length is incorrectly calculated is certain instances when creating multifile database
60507	database cache hinders performance
67648	InterClient: Bug Code 10010 occurs when reusing one statement after another statement has issued DDL

TABLE 0.8 Fixed bugs

Bug Number	Description
69510	CAST (<val> AS DATE) causes unpredictable errors in sql dialect 1
71192	Typo in error messages
73555	Trigger fires twice for each insertion into the subject table when trigger_source is set to NULL on an after-insert trigger
102222	Guardian launches on boot even though disable automatic launch on boot is set in Win2k control panel
105406	Views with quoted identifiers in dialect 3
106760	Default location of InterBase Replication Directory in Replication Server setup
107296	ISQL PLAN output buffer length need increased
108567	erroneous results when computing rounding values with dialect 1
109320	varchar truncation from server to client
109609	Multi-threaded application has problems with GDS client library on Linux
109914	Deleting sources from system tables with triggers causes some triggers to execute twice
110816	segfault for gds_inet_server
114677	TIP pages sequence needs to be LONG
116274	Populate description for InterBase Server and InterBase Guardian registry keys
117152	PROCEDURE handling numerics(x,x)
117890	Multi-threaded application gives SigSegv on Linux
118206	Sort files in temp directory when running ESQL
119520	Information for -r and -t options for gstat in documentation
120005	Multi-threaded application using InterBase on Windows Local Access hangs
120548	After gbak and restore only SYSDBA and owner can access database.
122004	Incomplete Alter Domain command
122135	libncurses.so.4 not found in RedHat 7.1, 7.2 for InterBase install

TABLE 0.8 Fixed bugs

Bug Number	Description
122495	Allow setting of CPU affinity on Windows for ibserver
122723	isql extract operation delimits all table and field names with for SQL_DIALECT 3
122907	Loading gdsintl2.dll module
123281	Server crashes when you try to make a unique index
123635	Possible database corruption on Unix databases by allowing the server to attach it as separate database instances

TABLE 0.8 Fixed bugs

Here are brief descriptions of the bugs fixed in the 6.0.1 release of InterBase. For more information, search the Borland community site at http://search.borland.com.

Bug Number	Description
58983	GRANT statement to Stored Procedures and Triggers fail
58985	NOT NULL constraints are evaluated prior to triggers
58988	Altering a procedure being run by another procedure fails
59012	Changes are not undone when exception is raised in before insert trigger
59014	Incorrect use of index on duplicate values causes crash
59022	SQL Error -924 when specifying and using alternate character set
59035	GBK permisison error for db owner when other user creates a table
59050	Error -607 when valid user revokes select privilege
59080	Server crash doing aggregate on aggregate view
60124	Conversion problem: Cast produces wrong result
60166	GBAK fails to restore a gdb that was successfully backed up
60188	Stored Procedures can be made to be Undeletable
60220	Location of ib_udf.dll is not documented

TABLE 0.9 Fixed bugs

Bug Number	Description
60221	Data Definition Guide section on declaring external files no clear
60255	Set options in ISQL are not parsed, until connecting to a database
60278	Adding multiple licenses concurrently causes crash
60288	Attempting to remove license using badly formatted key/id causes crash
60310	UNIX Install: /usr/interbase link is not correct if directory exists
60314	UNIX Install: All links are pointing directly to install dir
60316	ibserver does not recognize invalid WRITE error, and runs forever
60320	UNIX Install: Install scripts use ps cmd that is not portable
60325	Restore of ISC4.GDB generates an error but it created anyway
60360	Altering a domain type does not rebuild any indicies which were defined
60365	Server appears to start as both service and App (Icon on task bar)
60403	GPRE crashes with examples/stat3.e
60422	don't do - appears to not be in released build
60425	GPRE does not give error for DROP INDEX 'ind1'
60428	GPRE does not handle delimited cursor names properly
60430	ISQL showing wrong datatype for UDFs
60451	EXIT followed by when SQL SQLCODE causes rollback
60496	Incorrect result for "select -999.99 ..." in GPRE
61481	Creating db with stored procedure results in Parameters mismatch for procedure
61504	Install not recognizing non-administrative user
62001	GFIX with no parameters does not generate an error
62174	Triggers can created on non-existing fields
64315	Command line help for ibmgr lists passwd instead of password
64318	If lock table is out of room the server crashes

TABLE 0.9 Fixed bugs

Bug Number	Description
66143	Stored Procedure produces different results the second time it is called
66339	Cannot transliterate characters between character sets error when concatenating
67046	Dates with year greater than 10000 and less than 1 can be inserted successfully
68063	Adding id and key in wrong order when adding a license causes server to crash
68159	A UDF can point to any entry point to any system call
69057	Incorrect results for timestamp calculations
70109	All work done by triggers is not undone when an exception is raised
70443	-pa is incorrectly documented as command line option for gbak
70628	The guardian continually tries to start the server if a license cannot be found
70788	Procedure mismatch during alter procedure causes procedure to become useless
71191	Killing a client during a read crashes the server
71495	Queries with LIKE against Korean data set crash the server
71953	Create procedure statement which references a non-existent generator causes hang
72112	It is possible to enter 0000 as a valid year
72458	Dynamically loading and unloading of gds32.dll causes resource leaks
73766	SELECT .. WHERE .. BETWEEN crashes server
74091	Subqueries are not supported with the between predicate
75310	When Guardian restarts the server a thread handle is not closed
75637	Error 997;Overlapped I/O operation is in progress when stopping service on WIN2K
75902	Granting permission to update a column doesn't work for roles
76353	GBAK truncates metadata names after the first space, causes damaged backup
76500	Selecting the rdb$db_key with a group by will crash the server
87659	Stored Procedure returns incorrect result set
100331	SELECT from View with grouping causes server crash

TABLE 0.9 Fixed bugs

Bug Number	Description
101628	Creating too many generators can corrupt your database
102351	The make files in the examples all have InterBase 5.0 comments
102882	Select count(e.rdb$db_key) from employee e causes connection lost to database
100627	Superserver on Windows NT does not service more than 256 users
100649	Internal UDF enables database corruption
100651	Access to InterBase databases insecure on some versions

TABLE 0.9 Fixed bugs

Here are brief descriptions of the known bugs in the 6.0 release. For more information, or possible workarounds, search the Borland community site at http://search.borland.com.

Bug Number	Description
57972	Inappropriate bug check when out of disk space
60130	Operations on corrupted databases crash IBServer.exe
60131	Restore on multiple tapes does not work

TABLE 0.10 Known bugs