

Tuning Borland® InterBase® configuration parameters

Using InterBase configuration parameters to optimize
application performance

A Borland White Paper

By Sriram Balasubramanian, Borland InterBase R&D

September 2003

Borland®

Contents

Borland® InterBase® configuration parameters 3

- Configuration requirement decision..... 3
- What is provided and where can I find it? 3
- Parameters used by the database server 4
 - Database file related configuration 4
 - Operating system-related configuration..... 8
- Parameters used by the database client 15
- Using InterBase® Server Manager to modify some configuration parameters..... 16

Conclusion..... 21

- Appendix A – List of Configuration parameters in IBConfig..... 22

Borland® InterBase® configuration parameters

Borland® InterBase® is a powerful, high-performance, low-maintenance database that is particularly well-suited for embedding in applications that will be deployed where there is no IT support. To achieve this low maintenance requirement, InterBase is designed to be essentially self-tuning and possesses the intelligence necessary to adjust to dynamic requirements.

InterBase does possess a number of tunable configuration parameters that allow the high-end user to tune the database server (or the database) to perform optimally in particular environments. As a result, a single release of the product is able to address a wide range of usage scenarios and deployment environments.

Configuration requirement decision

InterBase is used for a wide variety of database applications. It is used in 2-tier, n-tier and multiple operating system (OS) platform environments. Each of these scenarios place a different burden on the database. Choices made in tuning the database depend upon the number of users anticipated over the life of the deployment (scalability) and availability (24/7, realtime) and reliability (transaction control, crash recovery, online backup) requirements. The configuration parameters discussed below and in the InterBase documentation are designed to help the developer or database administrator determine the best tuning for their individual needs.

What is provided and where can I find it?

InterBase provides a list of configuration parameters in a file by the name of *ibconfig*. This file is distributed with all versions of InterBase on all platforms and is found at the root of the InterBase install directory. (This was called *isc_config* on some UNIX® platforms in releases prior to InterBase 6.5).

The *ibconfig* file provides all the parameter names with default values assumed by the server, and a range of valid values for that parameter. In order for the database server to assume a different value than the default, you will have to edit the file and uncomment the parameter in question and update with a new value to it.

The *ibconfig* file is used by both the database server and the client. The client includes the database client library provided by InterBase and used by the database applications. This is namely *gds32.dll* on Windows® and *libgds.so* on UNIX® platforms.

InterBase documentation contains detailed information on all configuration parameters. Please see the chapter “Configuring parameters in IBConfig” in the “Operations Guide” PDF manual.

Parameters used by the database server

The server uses a variety of configuration parameters to manage its operation. It is pre-tuned to run well as installed, but may be helped with some specifics tuning based on the installation needs. Some of the tuning parameters are dependent on the resources available on the machine (memory, temporary file space, single CPU or SMP systems, etc.), others are dependent on the size of the database and the scalability requirements, while others are dependent on how the installation should be secured. The following sections identify parameters that address the above classifications.

Note: Please see Appendix A, for default value and valid values for each configuration parameter.

Database file related configuration

ADMIN_IB: InterBase maintains a security database containing valid username/password combinations of all users allowed to access databases available on a particular database server. This username and password combination is passed by the database clients either through the “-user and -password” options from command-line tools, or provided by the client application during the database connection call. The security database can be renamed to any name, but it has to reside within the InterBase install directory. Renaming the security

database is a simple but effective security mechanism to hide the security database from external applications.

For example, an InterBase security database might be renamed to `myapp_security.ib`. It would then be placed in the InterBase install directory. The parameter in the `ibconfig` file would be modified as follows:

```
ADMIN_IB    myapp_security.ib
```

After restarting the InterBase server all user/password authentication is done using the new security database '`myapp_security.ib`'. InterBase utilities such as '`gsec`', which allow administering the security database, can still make use of the new security database as follows:

```
# gsec -d myapp_security.ib
```

(Note: The security database was previously known as '`isc4.gdb`' on versions of InterBase prior to InterBase 7.0. From InterBase 7.0 onwards this database was called '`admin.ib`'. This was due to performance degradation experienced by users of Microsoft® Windows XP® which has a System Restore function that automatically backs up any file with the extension `.gdb`. To avoid this problem, the default database name was changed.)

DATABASE_CACHE_PAGES: Every database that is loaded by the InterBase server has a set of buffers in memory which are used to cache the data from the database. Every buffer has a one-to-one correspondence with a page on the database file.

If you want more of the database pages in memory, increase the `number_of_pages` by using `gfix`. Configure a specific number of cache pages for that database using the InterBase utility '`gfix -buffers <number_of_pages> <database_name>`'.

To set a large database cache for an individual database, set the database buffer cache value as stated above. The database server will then honor the value set in the database instead of the server-wide setting defined in the `DATABASE_CACHE_PAGES` parameter.

DATABASE_CACHE_PAGES parameter sets a value for all databases served by the server. If your intention is to define a global number for all the databases served by the database server, do it using the DATABASE_CACHE_PAGES parameter.

For example, say you are a Value-added Reseller (VAR) and you have an application which uses 2 databases, and both have a requirement to cache as many as 10,000 pages in memory. Your application is deployed at 2 different customer sites. Site A has a machine which has lots of memory, but Site B has a machine which has very low system memory. In order to tune your database to make best use of system memory, you do not have to modify the 'buffers' inside each database. Instead, you can modify the DATABASE_CACHE_PAGES value in the *ibconfig* file in each of these sites. As a result each database will consume the appropriate amount of available resources on the system.

Site A: *ibconfig* file DATABASE_CACHE_PAGES will contain 10000

Site B: *ibconfig* file DATABASE_CACHE_PAGES will contain 2000

The page size of a cache page is equal to the database page size. Typically, the database page size can be 1KB, 2KB, 4KB or 8KB for each page

EXTERNAL_FILE_DIRECTORY: InterBase databases allow you to query external flat files from within the database. These are called as External Tables. They are defined as any normal table within the database that also has an association with an external file.

You can accumulate all your external files to be used by the database onto one or multiple location(s). InterBase by default looks under the 'ext/' sub-directory of the InterBase install directory for such files. This allows you to control the areas that are visible to the database server for external data. This enhances security by ensuring that data from unsecure external locations are not available. In InterBase 6.x and before, it was possible to define an external table which mapped to the system file '/etc/passwd'. This file was open to query. This is no longer possible since '/etc/' does not belong to the list defined by

EXTERNAL_FILE_DIRECTORY.

The external file must reside in the directories defined in this parameter. This parameter can be defined multiple times. Each occurrence will define a different directory to be used by the server to authenticate external files for the database(s).

For example:.

```
EXTERNAL_FILE_DIRECTORY /usr/bsriram/
```

```
EXTERNAL_FILE_DIRECTORY /usr//local/data_files/
```

For more information on defining external tables, please see the “Language Reference” manual of the InterBase documentation set.

EXTERNAL_FUNCTION_DIRECTORY: InterBase allows you to define User Defined Functions (UDF) and deliver them as a library of functions to enhance the capabilities of the engine. For example you could have some custom functions (CONVERT_TO_MY_CURRENCY(), LOWERCASE_NAME(), NEGATE_NUMBER() etc.) which are used in your applications. These functions are coded and delivered as a native library for that Operating system and used in the SQL queries as an UDF.

These native libraries need to be placed in locations that are understood by the InterBase server, so it may dynamically load them when needed. InterBase, by default, looks under the ‘UDF/’ sub-directory of the InterBase install directory for such native DLLs. This enables a much more secure environment under which your libraries are loaded. If you would like to enable more directories containing such native DLLs for the UDFs, then you should use this parameter in the *ibconfig* file.

For example

```
EXTERNAL_FUNCTION_DIRECTORY /usr/bsriram/mylibs/
```

```
EXTERNAL_FUNCTION_DIRECTORY /usr/lib/applibs/
```

When the time comes to dynamically load a UDF library, InterBase will look in the above directories in addition to the ‘UDF/’ sub-directory. This will still maintain a secure platform

from where only known libraries are available to the InterBase databases, and no malignant libraries can be executed in the name of the InterBase server. This is very important to understand and use, since the InterBase server runs with administrative privileges on your system and thus can execute any function on your system. This is a sand-box model, enabling secure boundaries for your installation.

Operating system-related configuration

CPU_AFFINITY: This is specific to the Windows OS versions of InterBase. It is useful only on SMP (multiple CPU) systems. If you intend to dedicate specific CPUs on a system to be used by the InterBase server, you should define the set of CPUs as a bit-vector to this parameter.

For example to use processors 1 and 2 in a 4-CPU system, you need to define...

```
CPU_AFFINITY      3
```

The value 3 above is a bit vector of CPUs 1 and 2 (011).

InterBase comes licensed by default to use 1 CPU on the system. If you intend to use additional CPUs on the system with the InterBase processor, you need to purchase additional CPU licenses from Borland. The CPU_AFFINITY parameter will allow you to decide which licensed CPUs you want to use on the system. The number of CPUs mentioned in the CPU_AFFINITY should be less than or equal to the number of CPUs licensed to be used with InterBase.

The CPU_AFFINITY setting also disables the hopping of InterBase database server threads from one processor to another on a SMP system where only 1 CPU is licensed to be used with InterBase. This is basically a defect in the Windows O.S, which is fixed by setting the InterBase database server process to use only the specified CPU in the CPU_AFFINITY setting.

See Also: chapter “Expanded processor control” in the “Operations Guide” manual of the InterBase documentation set.

ENABLE_HYPERTHREADING: This is specific to the Windows OS versions of InterBase. Intel's hyper-threading technology enables "virtual-processors" to be used in parallel with "physical processors." This gives a performance boost to the order ranging from 10% to 70% for some applications according to Intel. On systems that come with hyper-threading (HT)-enabled processors (namely Intel® Pentium® 4 XEON™ processors, and other P4 processors), this may give a marginal boost to productivity, depending on what threads inside the InterBase server can run in parallel. It may also be a detriment in some scenarios where server threads are contending for the same resource, but are limited by the pipelining and sharing introduced by the HT technology. Basically, the operating system has a big impact on how HT-enabled systems perform.

We have heard conflicting reports from some InterBase customers on whether this really helps their application needs. HT-enabled systems seem to perform well under Windows Server™ 2003, and not so well under Windows® 2000. The reason might be because Microsoft claims that complete support for HT is not available on Windows 2000, but Windows Server 2003 is completely enabled for HT.

The parameter ENABLE_HYPERTHREADING is provided so that users may choose to enable the use of such "virtual-processors" on a HT-enabled system, if they feel the need for it. It is disabled by default. Since the performance boost is not perceived to be doubled because of the HT processors, InterBase recommends that SMP systems be used to provide more horse power, if needed, for the database server.

This parameter can be combined with the CPU_AFFINITY setting to carefully choose the "first" logical processor of each "physical" processor. This will give a better performance when compared to choosing both the "logical" processors from the same "physical" processor in the CPU_AFFINITY setting.

You can read more about Intel's hyper-threading technology at Intel's:

[Hyper-threading technology homepage](#),

or at Intel's online [tutorial on detecting Hyper-threading](#).

Or Microsoft white paper on [Windows support for Hyper-threading](#)

TMP_DIRECTORY: InterBase creates temporary files on the file system to store sorted and merged data. Data is sorted when a SQL query defines an ORDER BY clause, or a merge operation occurs on multiple streams of data. In both cases, the intermittent (sorted) data is stored on disk locally by the database server. The InterBase server uses the system temporary space (/tmp on UNIX, C:\temp on Windows) to store these files if it does not find any values in the system environment variables INTERBASE_TMP and TMP.

If you want the server to use a dedicated directory to store temporary files for your installation, please use the parameter TMP_DIRECTORY to define one or more directories. Each entry of this parameter in the *ibconfig* file should be accompanied with a size definition in bytes. This size defines the quota for the InterBase server temporary file size in that directory.

For example:

```
TMP_DIRECTORY      1000000      "d:\myhome\tmp"  
TMP_DIRECTORY      3000000      "d:\AppDir\tmp"
```

InterBase will use one million bytes from the "d:\myhome\tmp" directory first, and then use three million bytes from "d:\AppDir\tmp".

Please note that the directory names mentioned as values to this parameter should be enclosed in double quotes. This parameter is very useful when you do not want to clog the system temporary space used by all other applications on your system, with the InterBase temporary sort files. Server runtime configuration

DEADLOCK_TIMEOUT: Every page in a database is a resource. When a request (server-side execution) on behalf of a query executes, it may want to lock a resource to do some modifications. In a multi-user system where the database server is servicing multiple clients for the same database, this resource may be in-use and hence locked by someone else.

This parameter defines the time limit, in seconds, of how long to wait before doing a scan to check for deadlocks.

If you expect the system to be under heavy contention due to multi-user load, it may be advisable to extend this time to scan to, say, 30 seconds. This will delay deadlock scans from happening every 10 seconds and make it every 30 seconds. The result will be improved performance since the other threads can run, instead of the “deadlock scan” operation utilizing system resources.

DUMMY_PACKET_INTERVAL: In a client-server environment over the network, the client may provide some work to the server from time-to-time. In between queries, there may be a prolonged duration of time when the client does not communicate with the server. But the server has no way to know whether the client is alive, dead, or just idle.

In order to verify the existence of the client, the server sends a dummy packet of information to the client, which the client acknowledges on receipt. This lets the server know that everything is fine with the connection.

If the client did not respond with an ACK (acknowledgement code), the server can then terminate the application connection, and use it for some new connection. Since the number of connections to the server is limited to the licensed number of connections, it is a very useful feature to clean-up dead connections automatically. This leverages licensed usage very well.

If you have installations where you need to check for dead connections sooner, you may then need to reduce the DUMMY_PACKET_INTERVAL value. On the other hand, if you are very sure that the client applications have minimal to no chance of terminating abnormally, you may want to increase this value to, say, 300 seconds or so. This will reduce the number of dummy packets that are sent during idle time, thereby reducing network traffic, and also reducing the workload for the server to monitor these connections.

The default value for the DUMMY_PACKET_INTERVAL is 60 seconds.

MAX_THREADS: With the support of SMP systems in InterBase 7.0 and onwards, the database server can make use of more than one CPU on the system to service multiple client requests. This allows multiple threads within the engine to run in parallel making use of all the licensed processors on the system.

The thread scheduler inside the database server has been enhanced to allow multiple threads into the engine. This introduces some more contention for specific resources within the engine when more than one thread is present within.

On a multi-processor system, this makes perfect sense and also provides better scalability. On a single processor system, we have noticed that allowing only one thread into the scheduler has better performance. Hence, by default, InterBase defines this MAX_THREADS value to 1 on single-CPU systems, and to 1 million (practically infinite database threads) inside the engine for licensed SMP systems.

If you have only one database application running at a time (GBAK for example), you may notice better performance with MAX_THREADS set to 1, since there is no perceived contention for the specific database resources.

Another case where MAX_THREADS is set to 1, is if you have a SMP system, but have licensed only 1 CPU to be used by the database server. The server treats this as a single-CPU system.

SWEEP_QUANTUM: A sweep operation on a database essentially looks for all the garbage that it can collect and discard from the database. InterBase provides this operation so it can be performed at such a time when there is minimal load on the system hosting that database. This operation visits all the database pages, and hence does a lot of page I/O. As a result, it is best to schedule this activity at a time when there is minimal impact on operations.

InterBase also has a garbage collector thread that is running at all times while the database is online. When requests visit any page and find candidate records that have garbage versions in them, they are handed over to the 'garbage collector.' The 'garbage collector' then does its job of cleaning up all the provided list of records.

Both the above operations, in effect, take the pain of doing the garbage collection, thereby reducing the workload on the client applications. This provides better performance to the clients who are interested just in the results of the queries and not how data is managed in the file.

A SWEEP_QUANTUM defines the limit to a set of records worked upon by either the SWEEP or the ‘garbage collector’ before control is passed back to the InterBase thread scheduler, to schedule some other worker thread into the InterBase engine. This is co-operative work and not monopolistic. It basically ensures that the system resources are shared amongst all who need it within the InterBase server.

A default value of 250 is defined as the QUANTUM of records to be visited by either the sweeper or the garbage collector, before relinquishing control to the scheduler.

You may want to tune this parameter if you want the ‘garbage collector’ (GC) to work on more records at a time, or if you want the sweep operation to complete more quickly. The side-effect of increasing this value is that other user requests will have to wait longer before they can be scheduled by the InterBase thread scheduler, thereby affecting performance. If your application and database have very minimal garbage (back versions) in the database, you may choose to leave the values alone. On the other hand, if you anticipate a lot of garbage due to UPDATE/DELETE operations, you may want to give more QUANTUM to the sweeper or the GC thread.

SWEEP_YIELD_TIME: The parameter SWEEP_QUANTUM above defined the number of records visited by the sweeper or GC thread before yielding control to the scheduler. This parameter defines the time the sweeper or the GC thread waits before asking to be rescheduled, effectively putting them in the “ready to run” state. A one millisecond, which is the default, ensures that they are taken out of the “run” queue by the operating system. If you mention a value 0 (zero) here, you are effectively asking them to run continuously without yielding control to the thread scheduler. Making the value zero has the effect of a much faster GC or a sweep operation, but make note that other working threads in the system will not be yielded to during this time period.

You can try different combinations of the above 2 parameters SWEEP_QUANTUM and SWEEP_YIELD_TIME to suit your needs of sweep and GC.

For example:

SWEEP_QUANTUM 2000 and SWEEP_YIELD_TIME 10

SWEEP_QUANTUM 50000 and SWEEP_YIELD_TIME 50

SWEEP_QUANTUM 2000 and SWEEP_YIELD_TIME 0

USER_QUANTUM: This parameter is analogous to the SWEEP_QUANTUM, but applies to any worker thread that is working on behalf of a USER request. A user query might visit multiple pages in a database. This quantum value defines the number of records to be worked on by any worker thread before relinquishing control to the scheduler. This ensures that resources are shared amongst multiple threads in the server.

User_Quantum is also set to a default value of 250.

ANY_LOCK_MEM_SIZE: InterBase maintains one lock table per database server to maintain a list of lock request for resources. In a highly-used system, there may be many requests coming in from multiple clients that relate to creating multiple lock blocks. As the scalability requirements increase, you may want to change this value to a higher size (in bytes). By default, on Windows systems it is 256KB, and on UNIX/Linux® systems it is 96KB. This lock table is maintained by the InterBase server and is valid only for the duration when the server is online. You will also notice that this file is maintained on the file system by the name “<machine>.lck” on Windows, and, “isc_lock1.<machine>” on Unix/Linux systems under the InterBase install directory. Once this value is changed in the *ibconfig* file, you will have to restart the database server. The size change on the file system will confirm your change to this parameter.

ANY_EVENT_MEM_SIZE: InterBase has a very useful feature in the name of Event Alerters. Database client applications can register to listen for certain events, and they will be notified asynchronously by the database server, when the events are posted either by the Stored

Procedures or the Triggers within the database. This is a powerful feature—it means that clients do not have to keep polling for a certain value change. The SP or the Trigger logic can post events to all clients who have expressed interest in them.

These registered event requests are maintained by the database server in an Event table structure on the server. If you expect many event requests from the clients to the server, you may want to allocate a bigger Event table size. Modify the value in the *ibconfig* file and restart the InterBase server. The event file is named “<machine>.evn” on Windows, and, “isc_event1.<machine>” on Unix/Linux systems under the InterBase install directory.

SERVER_CLIENT_MAPPING: On Windows, InterBase allows another medium of communication between client and servers. This is the IPC mechanism, known as InterProcess Communication. This is possible only if the client applications and database server are running on the same machine.

The client and the server make use of shared memory on the system to pass messages (or data) between them. This buffer space is allocated by the server when it is started, and a chunk of space is allocated per client from this shared memory when the client initiates the connection to the server. Think of it as the bucket loaded with data that is used by the server and client to transmit data between themselves. Increasing the capacity of this bucket lets them communicate with more data during one transmission. But, this is the same size allocated for all buffers allocated by the server to all the clients on the same machine.

Valid values are 1024, 2048, 4096, 8192 bytes.

Parameters used by the database client

CONNECTION_TIMEOUT: A database application client might want to wait for a certain period of time before concluding that the server has not accepted its connection request. This may be due to heavy load on the server or because the server is not responding to any request. If the connection is not established within the specified time limit, the client reports the error back to the application.

If you have a low-speed connection medium (such as a low speed modem line), or a request across a WAN, you may want to increase this value to allow for network or communication medium-induced latency. The default value of 180 seconds is more than sufficient for internal networks and LANs.

TCP_REMOTE_BUFFER: This parameter applies to both the client and the server. InterBase allows communication between client and server on multiple standard transport protocols, including TCP/IP and NETBeui. TCP/IP in particular uses sockets to communicate between the client and server. Each end of the socket has a buffer defined to hold the data being communicated. This buffer is configurable with this parameter. The buffer used for the socket end is also dependent on system resources available on that system.

This buffer can be increased to accommodate transmission of more data on the network from the client to the server and vice-versa. Bulk transmission of data (which include backup/restore, and bulk retrieval) benefits greatly by this increase.

Please note that increasing this value on the server machine sets the send and receive buffers for each socket to that value. This may have an adverse impact on the system resources available. As a result, one should be careful when modifying this parameter, especially if hundreds of clients are serviced over the network.

Using InterBase® Server Manager to modify some configuration parameters

InterBase on Windows comes packaged with a “Server Manager” (aka “InterBase Manager”). You can use the InterBase Server Manager to modify DATABASE_CACHE_PAGES and SERVER_CLIENT_MAPPING parameters. This is available from the “Control Panel” as shown here.

Tuning Borland® InterBase® configuration parameters

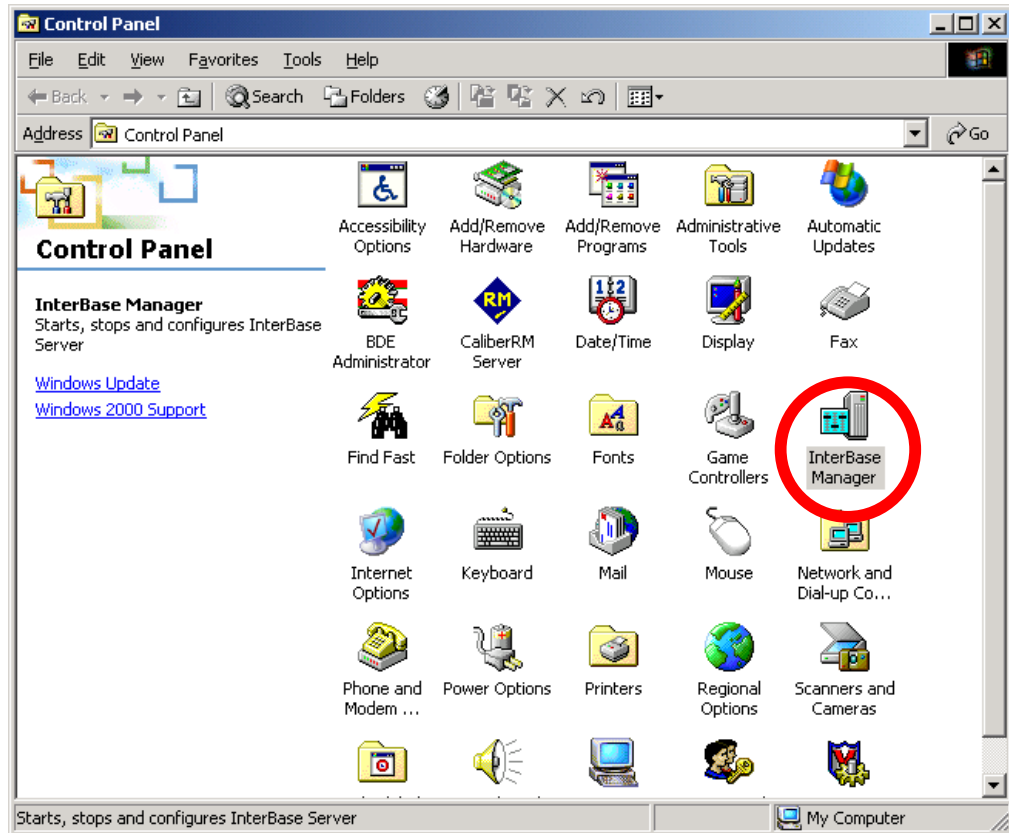


Figure 1: Windows Control Panel, select InterBase Manager

Launch the **InterBase Manager** as shown below

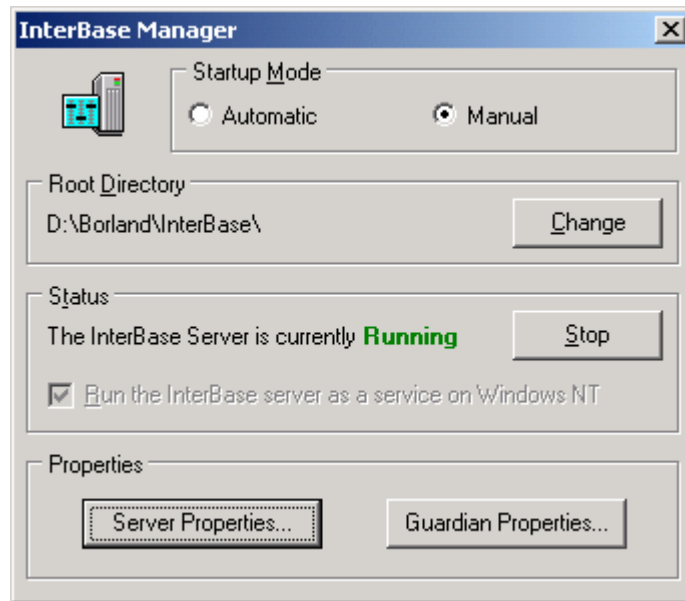


Figure 2: *InterBase Manager, select Server Properties button*

Click on the **Server Properties** button.

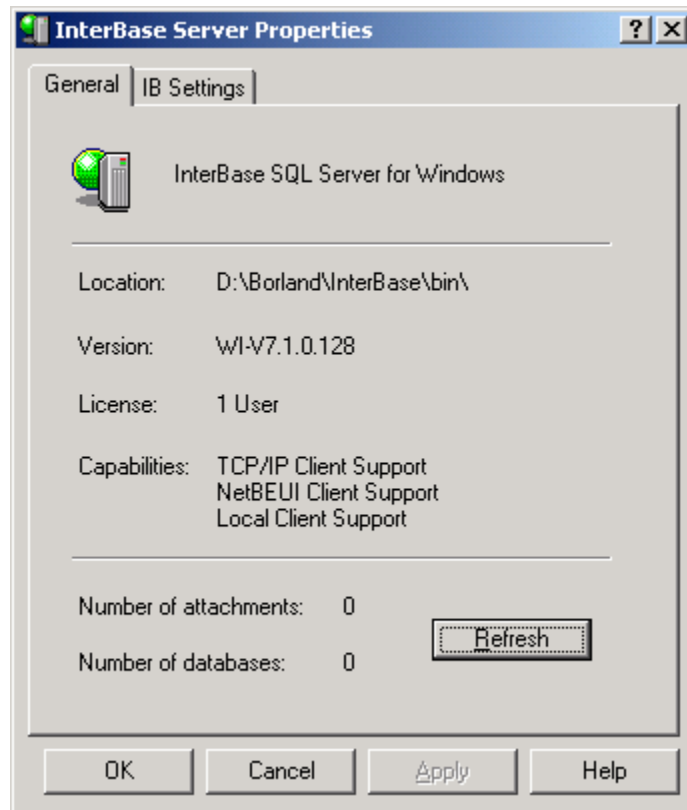


Figure 3: *InterBase Manager Server Properties*

Now, click on **IB Settings** tab.

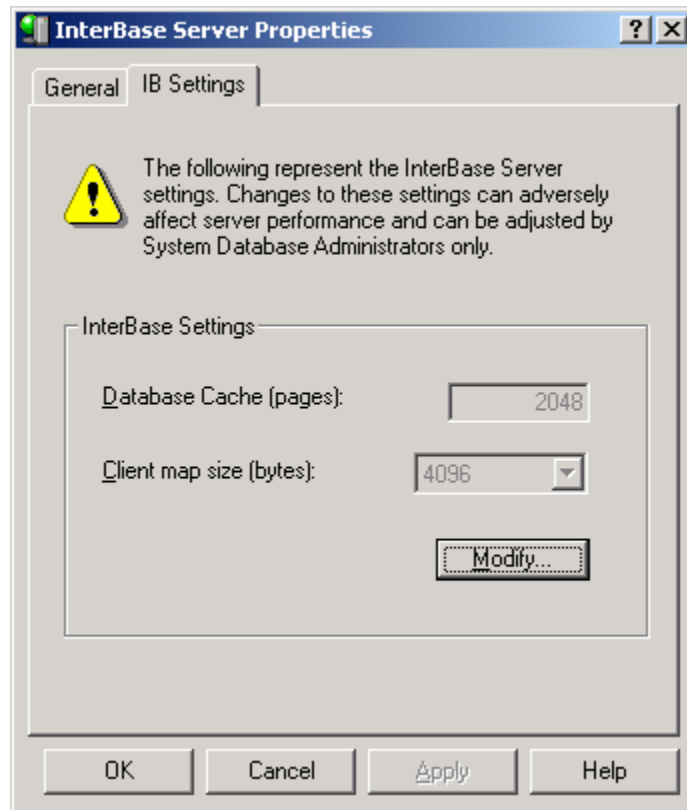


Figure 4: *InterBase Manager Server Properties, IB Settings*

Click on the **Modify** button and provide your SYSDBA user password. The 2 settings that are settable in this window are:

Database Cache (pages) which is equivalent to DATABASE_CACHE_PAGES parameter (described above) in the *ibconfig* file, and,

Client map size (bytes) which is equivalent to SERVER_CLIENT_MAPPING parameter (described above) in *ibconfig* file.

Click **OK** to save the changes.

Conclusion

InterBase is designed to be virtually maintenance free in production environments. As a result, it is essentially self-tuning. However, developers have a number of tuning parameters at their command that can help improve performance and this document has described them. Taking the time to work with the parameters and find the best mix of parameters and their settings can prove extremely worthwhile.

Appendix A – List of Configuration parameters in

IBConfig

The following is a listing of the IBConfig file found in the InterBase file directory. Each listing is preceded with the # symbol, indicating that the entry is a comment. The default value is listed to the right of each entry. You may cut and paste from this listing into the IBConfig file that you are tuning. To activate an entry, uncomment it by removing the # symbol (s) and supply the desired value.

```
#ADMIN_DB          "admin.ib"
```

```
## Specifies a filename for the InterBase security database.
```

```
## Needed only if you do not want to use the default name, admin.ib.
```

```
## The security database must reside in the InterBase home directory.
```

```
#ANY_EVENT_MEM_SIZE      32768
```

```
## Bytes of shared memory allocated for event manager.
```

```
#ANY_LOCK_MEM_SIZE       98304
```

```
## Bytes of shared memory allocated for lock manager.
```

```
## Default is 98304 on Linux and Solaris,™ 256K on Windows.
```

#ANY_LOCK_SEM_COUNT 32

Number of semaphores for interprocess communication

(Classic architecture only).

#ANY_LOCK_SIGNAL 16

UNIX signal to use for interprocess communication

(Classic architecture only).

#CPU_AFFINITY

(Windows only) In an SMP system, sets which processors can be used

by the InterBase server. The value is taken from a bit vector in

which each bit represents a machine. Thus, to use only the first

processor, the value is 1. To use both the first and second processors,

the value is 3. To use the second and third, the value is 6.

Default is all processors (when entry is commented out).

#CONNECTION_TIMEOUT 180

Seconds to wait before concluding an attempt to connect has failed.

#DATABASE_CACHE_PAGES 2048

Server-wide default for the number of database pages

to allocate in memory per database.

Can be overridden by clients.

#DEADLOCK_TIMEOUT 10

Seconds before an ungranted lock causes a scan to check for deadlocks.

#DUMMY_PACKET_INTERVAL60

Seconds to wait on a silent client connection before the

server sends dummy packets to request acknowledgment.

#ENABLE_HYPERTHREADING 1

Specifies whether Intel Hyper-threading technology enabled logical

processors should be enabled

Valid values are: 1 (enable), 0 (disable)

Default: 0 (disable)

#EXTERNAL_FILE_DIRECTORY

If your external files are not in <interbase_home>/ext,
specify their location here. For security reasons, do not
put other files in this directory.

#EXTERNAL_FUNCTION_DIRECTORY

If your UDF library is not in <interbase_home>/UDF, then specify
the location of the library here. For security reasons, do not
put other files in this directory.

#LOCK_ACQUIRE_SPINS 0

Number of spins during a busy wait on the lock table mutex.
Relevant only on SMP machines.

#LOCK_HASH_SLOTS 101

Tune lock hash list; more hash slots mean shorter hash chains.
Not necessary except under very high load.
Prime number values are recommended.

#MAX_THREADS 1000000

Controls the maximum number of threads that can be active

at one time within the InterBase engine. The listed value

applies to a system with multiple licensed CPUs. For a

single CPU system, the value defaults to 1 which eliminates

the synchronization overhead required by multiple CPUs.

#SERVER_CLIENT_MAPPING 4096

Size in bytes of one client's portion of the memory mapped

file used for interprocess communication.

#SERVER_PRIORITY_CLASS 1

Priority of the InterBase service on Windows NT,® XP,® or Windows® 2000, .

The value 1 is low priority, 2 is high priority.

Relevant only on Windows NT/2000/XP.

#SERVER_WORKING_SIZE_MAX 0

Threshold above which the OS is requested to swap out all memory.

Relevant only on Windows NT, XP, and 2000

Default is system-determined

#SERVER_WORKING_SIZE_MIN 0

Threshold below which the OS is requested to swap out no memory

Relevant only on Windows NT, XP, and 2000

Default is system-determined

#SWEEP_QUANTUM 250

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.

#SWEEP_YIELD_TIME 1

Specifies the time, in milliseconds, the sweeper or

garbage collector thread sleeps.

#TCP_REMOTE_BUFFER 8192

TCP/IP buffer size for send and receive buffers. This applies to

both client and server programs.

Default is 8192 bytes. Valid range is 1448 to 32768

#TMP_DIRECTORY

Directory to use for storing temporary files.

Specify directory path and number of bytes available in the directory.

List multiple entries one per line; directories are

used in the order specified.

Default is the value of the INTERBASE_TMP environment

variable; otherwise /tmp on UNIX or C:\temp on Windows NT/2000/XP.

#USER_QUANTUM 250

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.

#V4_EVENT_MEMSIZE 32768

Bytes of shared memory allocated for event manager.

Overridden by ANY_EVENT_MEMSIZE.

#V4_LOCK_GRANT_ORDER 1

1 means locks are granted first come, first served.

0 means emulate InterBase V3.3 behavior, where locks are granted

as soon as they are available; can result in lock request starvation.

#V4_LOCK_MEM_SIZE 98304

Bytes of shared memory allocated for lock manager.

Default is 98304 on Linux and Solaris, 256K on Windows.

Overridden by ANY_LOCK_MEM_SIZE.

#V4_LOCK_SEM_COUNT 32

Number of semaphores for interprocess communication

(Classic architecture only).

Overridden by ANY_LOCK_SEM_COUNT.

#V4_LOCK_SIGNAL 16

UNIX signal to use for interprocess communication

(Classic architecture only).

Overridden by ANY_LOCK_SIGNAL.

#V4_SOLARIS_STALL_VALUE 60

Number of seconds a server process waits before retrying

for the lock table mutex.

Relevant on Solaris only.

Made in Borland® Copyright © 2003 Borland Software Corporation. All rights reserved. All Borland brand and product names are trademarks or registered trademarks of Borland Software Corporation in the United States and other countries. Microsoft, Windows, and other Microsoft product names are trademarks or registered trademarks of Microsoft Corporation in the U.S. and other countries. Linux is a registered trademark of Linus Torvalds. All other marks are the property of their respective owners. Corporate Headquarters: 100 Enterprise Way, Scotts Valley, CA 95066-3249 • 831-431-1000 • www.borland.com • Offices in: Australia, Brazil, Canada, China, Czech Republic, Finland, France, Germany, Hong Kong, Hungary, India, Ireland, Italy, Japan, Korea, Mexico, the Netherlands, New Zealand, Russia, Singapore, Spain, Sweden, Taiwan, the United Kingdom, and the United States. • 20947