Firebird's nbackup tool

Paul Vinkenoog

19 Sep 2011 — Document version 1.2

Table of Contents

Introduction

Nbackup is a backup utility included in the Firebird download packages since version 2.0. It offers possibilities not present in *gbak* – Firebird's pre-existing backup tool – but doesn't replace the latter. Both programmes have their strengths and weaknesses; they will likely coexist for some time to come.

Nbackup features – an overview

With nbackup, you can perform two different kinds of tasks:

1. Making and restoring of both full and *incremental* backups. An *incremental backup* only contains the mutations since some specific previous backup.

2. Locking the main database file so you can subsequently back it up yourself with copying or backup tools of your own choice. In this mode, nbackup doesn't back up anything; it just creates the conditions under which you can safely make the backup yourself. There's a provision for restoring here, too.

Both modes can operate on an active database, without hindering connected users. The backup created will always reflect the state of the database *at the beginning of the operation*. Furthermore, only SYSDBA and the database owner (and sometimes OS admins) can make a backup, but every user can restore a backup to a new database. In these respects nbackup doesn't differ from gbak.

Advantages of nbackup

- *Both modes:* high speed (as high as hardware and OS will allow), because nbackup doesn't look at the actual data. In backup mode the contents are written more or less blindly to the backup file.

- *Backup/restore mode:* time and disk space savings, because you don't need to make a full backup every time. This can make a huge difference with databases in the gigabyte range.

- *Lock/unlock mode:* total freedom in your choice of backup, copy, and/or compression tools.

Limitations of nbackup

- Nbackup will not sweep and compact your database the way gbak does.

- You can't change the database owner with an nbackup backup/restore cycle, like you can with gbak.

- Nbackup can't make *transportable backups*, that is: backups you can restore on an incompatible platform or under another server version.

- At this moment, nbackup should not be used on multi-file databases.

- Nbackup itself can only back up local databases. However, in Firebird 2.5 and above its backup and restore tasks can also be performed remotely through the Services Manager.

- Except when the Services Manager is used (in Firebird 2.5+) backing up with nbackup requires direct access to the database file. With gbak this is not the case.

Functions and parameters

The following table gives an overview of nbackup's parameters. If the field “Added” is empty, the parameter has existed since nbackup's introduction in Firebird 2.0.

Table 1. Nbackup parameters

Parameter	Function	Added
-B *n* *<database>* [*<filename>*]	Make level-*n* backup of database to file	
-R *<database>* [*<filename>* ...]	Restore database from backup file(s)	
-L *<database>*	Lock database	
-N *<database>*	Unlock locked database	
-F *<database>*	Unlock self-restored database	
-S	Give size in database pages (with -L)	2.1
-T	Suppress database triggers (with -B, -L, -N)	2.1
-D on|off	Direct I/O on/off (with -B)	2.1.4
-U *<username>*	Supply user name (with -B, -L, -N)	
-P *<password>*	Supply password (with -B, -L, -N)	
-FE *<filename>*	Fetch password from file (with -B, -L, -N)	2.5
-Z	Version info (by itself or with -B, -R, -L, -N, -F)	2.5
-?	Help (switches off all other parameters)	2.5

Depending on the chosen main function (*-B*, *-R*, *-L*, *-N* or *-F*), nbackup may require different types of access to the database: a Firebird server connection, direct file access, or both. The following table gives the details:

Table 2. Access required

Parameter	Function	Access
-B	Backup	server + file
-R	Restore	file
-L	Lock	server
-N	Unlock (undo -L)	server
-F	Unlock after self-restore	file

Where server access is required (with `-B`, `-L` and `-N`), the user must either provide a Firebird username and password (with `-U` and `-P`/`-FE` or through the environment variables `ISC_USER` and `ISC_PASSWORD`), or be admitted by the server on other grounds (e.g. as root under Posix or by trusted authentication under Windows).

Where filesystem access is required (with `-B`, `-R` and `-F`), the user must have sufficient read and/or write privileges to the database file.

Where filesystem access is required exclusively (with `-R` and `-F`), the user need not have a Firebird login and a running Firebird server need not be present.

Please notice: The above table and text concern access to the *database*. Access to the backup file is – obviously – always on the filesystem level.

Making and restoring backups

To begin with: `nbackup.exe` is located in the `bin` subdirectory of your Firebird folder. Typical locations are e.g. `C:\Program Files\Firebird\Firebird_2_0\bin` (Windows) or `/opt/firebird/bin` (Linux). Just like most of the tools that come with Firebird, nbackup has no graphical interface; you launch it from the command prompt or call it from within a batch file or application.

Warning

Under heavy-load circumstances in some environments, nbackup 2.0.3 and below may cause problems that will lead to deadlocks or even corrupted databases. While these problems aren't common, they are serious enough to warrant upgrading to Firebird 2.0.4 or higher if you want to use nbackup comfortably. If it concerns large databases under Posix, the use of direct I/O may also make a difference. More about this in the section *Direct I/O*.

Full backups

Making full backups

To make a full database backup, the command syntax is:

```
nbackup [-U <user> -P <password>] -B 0 <database> [<backupfile>]
```

For instance:

```
C:\Data> nbackup -B 0 inventory.fdb inventory_1-Mar-2006.nbk
```

Comments:

- The parameter `-B` stands for backup (gee!). The *backup level* 0 indicates a full backup. Backup levels greater than 0 are used for incremental backups; we'll discuss those later on.
- Instead of a database filename you may also use an alias.

- Instead of a backup filename you may also specify `stdout`. This will send the backup to standard output, from where you can redirect it to e.g. a tape archiver or a compression tool.
- The `-U` (user) and `-P` (password) parameters may be omitted if at least one of the following condtions is met:
  - The environment variables `ISC_USER` and `ISC_PASSWORD` have been set, either to SYSDBA or to the owner of the database.
  - You are logged on as root on a Posix system. This makes you SYSDBA by default.
  - Under Windows: Trusted authentication is enabled in `firebird.conf`, and you are logged on to the Windows account that owns the database. This is possible in Firebird 2.1 and above.
  - Under Windows: Trusted authentication is enabled in `firebird.conf`, and you are logged on as a Windows administrator. In Firebird 2.1, this automatically gives you SYSDBA rights. In Firebird 2.5 and above, there is the additional condition that AUTO ADMIN MAPPING has been set in the database.

  For clarity and brevity, the `-U` and `-P` parameters are not used in the examples.
- Starting with Firebird 2.5, instead of `-P <password>` you may also use `-FE <filename>`. This will cause nbackup to fetch the password from the given file. With `-FE`, the password itself doesn't appear in the command and will thus be better shielded against people who might otherwise pick it up via the command history, the `w` command on Unix or from a script or batchfile.
- In Firebird 2.1 and up, the firing of database triggers can be prevented by specifying the `-T` option. For more information, see *Suppressing database triggers*.
- Starting with Firebird 2.1.4, it is possible to force direct I/O on or off by specifying `-D on` or `-D off`. For details and background see *Direct I/O*, elsewhere in this manual.
- The different parameters (`-B`, `-U` etc.) may occur in any order. Of course each parameter should be immediately followed by its own argument(s). In the case of `-B` there are three of them: backup level, database, and backup file - in that order!
- If the `-B` parameter comes last, you *may* leave out the name of the backup file. In that case nbackup will compose a filename based on the database name, the backup level, and the current date and time. This can lead to a name clash (and a failed backup) if two backup commands of the same level are issued in the same minute.

> **Warning**
>
> Do *not* use nbackup for multi-file databases. This can lead to corruption and loss of data, despite the fact that nbackup will not complain about such a command.

A word on the inner workings

Note: What follows here is not necessary knowledge to use nbackup. It just gives a rough (and incomplete) impression of what happens under the hood during execution of nbackup `-B`:

1. First of all, the main database file is locked by changing an internal state flag. From this moment on, any and all mutations in the database are written to a temporary file - the difference file or *delta file*.
2. Then the actual backup is made. This isn't a straight file copy; restoring must be done by nbackup as well.
3. Upon completion of the backup, the contents of the delta file are integrated with the main database file. After that, the database is unlocked (flag goes back to “normal”) and the delta is removed.

The functionality of steps 1 and 3 is provided by two new SQL statements: ALTER DATABASE BEGIN BACKUP and ALTER DATABASE END BACKUP. Contrary to what the names suggest, these statements do *not* take care of making the actual backup; rather, they create the conditions under which the main database file can be safely backed up. And to be clear: you don't need to issue these commands yourself; nbackup will do that for you, at the right moments.

Restoring a full backup

A full backup is restored as follows:

```
nbackup -R <database> [<backupfile>]
```

For instance:

```
C:\Data> nbackup -R inventory.fdb inventory_1-Mar-2006.nbk
```

Comments:

- You don't specify a level for a restore.
- When restoring, the `-R` parameter *must* come last, for reasons that will become clear later.
- Instead of a database filename you may also use an alias.
- If the specified database file already exists, the restore fails and you get an error message.
- Here too, you may omit the name of the backup file. If you do, nbackup will prompt you for it. *(Attention! In Firebird 2.0 this “interactive restore” feature is broken, leaving you with an error message and a failed restore. Fixed in 2.0.1.)*
- Restoring works purely on the filesystem level and can even be done without a Firebird server running. Any credentials supplied via the `-U` and `-P` parameters are ignored. The same goes for passwords read from a file. However, nbackup *does* try to read the password from the file if the `-FE` parameter is present, and if an error occurs, the entire operation is abandoned.

Incremental backups

Warning

The incremental backup facility was entirely broken in Firebird 2.1, and fixed again in 2.1.1.

Making incremental backups

To make an incremental (“differential”) backup we specify a backup level greater than 0. An incremental backup of level `N` always contains the database mutations since the most recent level `N-1` backup.

Examples:

One day after the full backup (level 0), you make one with level 1:

```
C:\Data> nbackup -B 1 inventory.fdb inventory_2-Mar-2006.nbk
```

This backup will only contain the mutations of the last day.

One day later again, you make another one with level 1:

```
C:\Data> nbackup -B 1 inventory.fdb inventory_3-Mar-2006.nbk
```

This one contains the mutations of the last *two* days, since the full backup, not only those since the previous level-1 backup.

A couple of hours on we go for a level-2 backup:

```
C:\Data> nbackup -B 2 inventory.fdb inventory_3-Mar-2006_2.nbk
```

This youngest backup only contains the mutations since the most recent level-1 backup, that is: of the last few hours.

Note

All the comments that have been made about full backups also apply to incremental backups.

Warning

Again: do not use nbackup for multi-file databases.

Restoring incremental backups

When restoring incremental backups you must specify the entire chain of backup files, from level 0 through the one you wish to restore. The database is always built up from the ground, step by step. (It is this stepwise adding until the database is restored that gave rise to the term *incremental backup*.)

The formal syntax is:

```
nbackup -R <database> [<backup0> [<backup1> [...] ] ]
```

So restoring the level-2 backup from the previous example goes as follows:

```
C:\Data> nbackup -R inventory.fdb inventory_1-Mar-2006.nbk
           inventory_3-Mar-2006.nbk inventory_3-Mar-2006_2.nbk
```

Of course the line has been split here for layout reasons only - in reality you type the entire command and only hit **Enter** at the end.

Comments (in addition to the comments with restoring a full backup):

- Because it is not known beforehand how many filenames will follow the *-R* switch (as we don't specify a level when restoring), nbackup considers all arguments after the *-R* to be names of backup files. It is for this reason that no other parameter may follow the list of filenames.
- There is no formal limit to the number of backup levels, but in practice it will rarely make sense to go beyond 3 or 4.

Non-connecting links

What happens if you accidentally leave out a file, or specify a series of files that don't all belong together? You could imagine that you specify `inventory_2-Mar-2006.nbk` by mistake instead of `inventory_3-Mar-2006.nbk` in the above example. Both are level-1 backup files, so in both cases we get a nice “0, 1, 2” level series. But our level-2 file is incremental to the level-1 backup of 3 March, not to the one of 2 March.

Fortunately such a mistake can never lead to an incorrectly restored database. Each backup file has its own unique ID. Furthermore, each backup file of level 1 or above contains the ID of the backup on which it is based. When restoring, nbackup checks these IDs; if somewhere in the chain the links don't connect, the operation is cancelled and you get an error message.

Backing up raw-device databases

Firebird databases need not be files; they can also be placed on a so-called *raw device*, for instance a disk partition without a file system. The question where the delta has to be placed in such cases was at first overlooked during the development of nbackup. On Posix systems, if the database was located at e.g. `/dev/hdb5`, it could happen that the delta was created as `/dev/hdb5.delta`. In light of the nature and purpose of the `/dev` directory and its often limited available space, this is undesirable.

As of Firebird 2.1, nbackup refuses to operate on raw-device databases unless an explicit location for the delta file has been set. The way to do this is discussed in *Setting the delta file*, later on in this manual.

Suppressing database triggers (Firebird 2.1+)

Firebird 2.1 introduced the concept of *database triggers*. Certain types of these triggers can fire upon making or breaking a database connection. As part of the backup process, nbackup opens a regular connection to the database (in some versions even more than once). To prevent database triggers from firing inadvertently, the new *`-T`* switch can be used. Notice that the corresponding switches in gbak and isql are called *`-nodbtriggers`* (we love diversity, here at Firebird).

Direct I/O (Firebird 2.1.4+)

Originally, nbackup used direct I/O only when making a backup under Windows NT (and successors like 2000, 2003 etc.) On all other OS'es, direct I/O was off. This caused problems on some Linux systems, so in versions 2.0.6 and 2.1.3 direct I/O was switched on under Linux as well. However, this turned out to be problematic for certain other Linux configurations. In 2.1.4 and 2.5 the original behaviour was restored, but this time as a default that was overridable by a newly added parameter: *`-D`*. Its use is as follows:

```
nbackup -B 0 cups.fdb cups.nbk -D on    -- direct I/O on
nbackup -B 0 mugs.fdb mugs.nbk -D off   -- direct I/O off
```

Just like the option letters themselves, the arguments ON and OFF are case-insensitive.

Direct I/O is only applied when making a backup, not during a restore. Under Windows it is realized by setting `FILE_FLAG_NO_BUFFERING`. On other systems, `O_DIRECT` and `POSIX_FADV_NOREUSE` are used. The latter two are sometimes unavailable; in such cases, they are (or one of them is) silently left out. Even if the user specified `-D on` explicitly, this doesn't lead to a warning or error message.

Informational options (Firebird 2.5+)

Apart from the already mentioned `-FE` and `-D` parameters, Firebird 2.5 also saw the introduction of the following two:

`-Z`
: Shows single-line version information. This option can be used independently, but also in combination with other parameters, such as `-B`, `-R`, `-L` etc.

`-?`
: Shows a summary of nbackup's usage and command-line parameters. Attention: If this option is present, all the other parameters are ignored!

Backups on remote machines (Firebird 2.5+)

Nbackup itself only operates on local databases. But in Firebird 2.5 and up, nbackup-type backups and restores can also be performed remotely via the Services Manager. For this, the program `fbsvcmgr.exe` on the local machine is used; it is located in the same folder as `nbackup.exe` and the other Firebird command-line tools. The first argument is always “*hostname*:`service_mgr`”, with *hostname* being the name of the remote server. Other available parameters are:

```
-user username
-password password
-action_nbak
-action_nrest
-nbk_level n
-dbname database
-nbk_file filename
-nbk_no_triggers
-nbk_direct on|off
```

Making a full backup on the remote machine `frodo` goes like this:

```
fbsvcmgr frodo:service_mgr -user sysdba -password masterke
  -action_nbak -nbk_level 0
  -dbname C:\databases\countries.fdb -nbk_file C:\databases\countries.nbk
```

And a subsequent incremental backup:

```
fbsvcmgr frodo:service_mgr -user sysdba -password masterke
  -action_nbak -nbk_level 1
  -dbname C:\databases\countries.fdb -nbk_file C:\databases\countries_1.nbk
```

To restore the whole shebang:

```
fbsvcmgr frodo:service_mgr -user sysdba -password masterke
  -action_nrest -dbname C:\databases\countries_restored.fdb
  -nbk_file C:\databases\countries.nbk -nbk_file C:\databases\countries_1.nbk
```

Notice: Each of the above commands should be typed as a single sentence, without line breaks. The hyphens before the parameter names may be omitted, but especially with long commands like these it may be helpful to leave them in, so you can easily identify the individual parameters (the arguments don't get a hyphen).

Comments:

- The Services Manager always requires authentication, be it automatic (root under Posix, trusted under Windows) or explicit through the parameters *`-user`* and *`-password`*. The environment variables `ISC_USER` and `ISC_PASSWORD` are not used. AUTO ADMIN MAPPING in the database has no effect when connecting remotely (though this may also depend on the configuration of the network).

  Note: When Windows trusted authentication is in effect, the account name of the user on the local machine is passed to the Services Manager on the remote machine. If the owner of the remote database is a Windows account (e.g. `FRODO\PAUL`) rather than a Firebird account, *and* the Windows account name on the local machine is the same as the owner account name on the remote machine, the caller is acknowledged as the database owner and allowed to make a backup. This could pose a security risk, because even on local networks user `PAUL` on one machine is not necessarily the same person as user `PAUL` on another machine.

- Restoring (*`-action_nrest`*) also requires authentication, but once verified the credentials are not used in any way. Hence, the user need not be the database owner, SYSDBA or superuser. In the case of Windows trusted authentication, the user need not exist at all on the remote machine (where the database is located).

  This weak authentication implies another potential security risk. Suppose a sensitive database is nbackupped, and the backups are well protected on the filesystem level. An average user can't restore the database with nbackup then, because nbackup runs in the user process space. But that same user, if he knows name and location of the backup, or can guess them by analogy, might be able to get hold of the database by having fbsvcmgr restore it to a public folder. After all, fbsvcmgr calls the Firebird server, which may have file-level access to the backup. Of course there are solutions to this, but it's important to be aware of the risk.

- The Services Manager can also be used locally; in that case the first argument becomes *`service_mgr`*, without hostname. When used locally, AUTO ADMIN MAPPING has the intended effect; this is still true if you prepend `localhost:` or the name of the local machine. Local use of the Services Manager can be beneficial if you don't have filesystem access to the database and/or backup files, but the Firebird server process does. If you do have sufficient rights, then it's more practical to use nbackup itself, with its much shorter commands.

- Specifying *`-nbk_no_triggers`* or *`-nbk_direct`* with *`-action_nrest`* leads to an error message. Nbackup itself is more lenient here: it simply ignores the *`-T`* and *`-D`* parameters if they are used in the wrong context.

- Instead of a database filename you may also use an alias.

A practical application

An nbackup-based incremental backup scheme could look like this:

- Each month a full backup (level 0) is made;
- Each week a level-1;
- A level-2 backup daily;
- A level-3 backup hourly.

As long as all backups are preserved, you can restore the database to its state at any hour in the past. For each restore action, a maximum of four backup files is used. Of course you schedule things in such a way that the bigger, time-consuming backups are made during off-peak hours. In this case the levels 0 and 1 could be made at weekends, and level 2 at night.

If you don't want to keep everything for eternity, you can add a deletion schedule:

- Level-3 backups are deleted after 8 days;
- Level-2s after a month;
- Level-1s after six months;
- Full backups after two years, but the first one of each year is kept.

This is only an example of course. What's useful in an individual case depends on the application, the size of the database, its activity, etc.

Read on?

At this point you know everything you need in order to make and restore full and/or incremental backups with nbackup. You only need to read any further if you want to use backup tools of your own choice for your Firebird databases (see *Locking and unlocking*), or if you want to override the default name or location of the delta file (see *Setting the delta file*).

If you have no craving for any of that: good luck in your work with nbackup!

Locking and unlocking

If you prefer to use your own backup tools or just make a file copy, nbackup's lock-unlock mode comes into view. “Locking” means here that the main database file is frozen temporarily, not that no changes can be made to the database. Just like in backup mode, mutations are directed to a temporary delta file; upon unlocking, the delta file is merged with the main file.

As a reminder: `nbackup.exe` lives in the `bin` subdir of your Firebird folder. Typical locations are e.g. `C:\Program Files\Firebird\Firebird_2_0\bin` (Windows) or `/opt/firebird/bin` (Linux). There's no GUI; you launch it from the command prompt or call it from within a batch file or application.

Locking the database and backing up yourself

A typical session in which you make your own backup goes as follows:

1. Lock the database with the `-L` (lock) switch:

   ```
   nbackup [-U <user> -P <password>] -L <database>
   ```

2. Now copy/backup/zip the database file to your heart's content, with your own choice of tools. A simple file copy is also possible.

3. Unlock the database with `-N` (uNlock):

   ```
   nbackup [-U <user> -P <password>] -N <database>
   ```

The last command will also cause any mutations - which have been written to the delta file - to be merged into the main file.

The backup you made contains the data as they were at the moment the database was locked, regardless how long the locked state has lasted, and regardless how long you may have waited before making the actual backup.

Comments:

- Instead of a database filename you may also specify an alias.
- The *-U* and *-P* parameters may be omitted if the envars `ISC_USER` and `ISC_PASSWORD` are set, if you are root on a Posix system, or if trusted authentication under Windows permits it. For a detailed description see the comments under *Making full backups*.
- Starting with Firebird 2.5, instead of *-P <password>* you may also use *-FE <filename>*.
- Both *-L* and *-N* make a regular connection to the database, so in Firebird 2.1 and above it may be wise to add the *-T* parameter (see *Suppressing database triggers*).
- If you're locking a raw-device database with Firebird 2.1 or above, the *-S* option can be very helpful; see *Locking raw-device databases*.
- You can optionally add *-Z* to have version information printed on the first line of the output.

> **Warning**
>
> What goes for backup/restore also applies to the lock/unlock switches: do not use them on multi-file databases. Until things have changed, don't let nbackup loose on multi-file databases at all!

Restoring a backup made after “nbackup -L”

An copy of a locked database is itself a locked database too, so you can't just copy it back and start using it. Should your original database get lost or damaged and the self-made copy needs to be restored (or should you wish to install the copy on another machine), proceed like this:

1. Copy/restore/unzip the backed-up database file yourself with the necessary tools.
2. Now unlock the database, *not* with the *-N* switch, but with *-F* (fixup):

   ```
   nbackup -F <database>
   ```

   Here too, you can optionally use an alias instead of a filename, and add *-Z* for version info. Other options make no sense.

Why are there two unlock switches, *-N* and *-F*?

- *-N* first sees that any changes made since the locking by *-L* are merged into the main database file. After that, the database goes back into normal read/write mode and the temporary file is deleted.
- *-F* only changes the state flag of the self-restored database to “normal”.

So you use:

- *-N* after having *made* a copy/backup yourself (to reverse the *-L* issued earlier);
- *-F* after having *restored* such a backup yourself.

Note

It is a bit unfortunate that the last switch should be called `-F` for Fixup. After all, it doesn't fix anything; it only *unlocks* the database. The `-N` (uNlock) flag on the other hand performs not only an unlock, but also a fixup (integration of mutations into the main file). But we'll have to live with that. Come to think of it: you *can* read `-F` as *Flag-only*.

Under the hood

Note: This section doesn't contain any necessary knowledge, but provides some extra information which could deepen your understanding of the various switches.

nbackup `-L` does the following:

1. Connect to the database;
2. Start a transaction;
3. Call ALTER DATABASE BEGIN BACKUP (this statement has been discussed in the extra information on nbackup -B);
4. Commit the transaction;
5. Disconnect from the database.

nbackup `-N` follows the same steps, but with "...END BACKUP" in step 3.

nbackup `-F` works as follows:

1. The restored database file is opened;
2. Within the file, the state flag is changed from locked (`nbak_state_stalled`) to normal (`nbak_state_normal`);
3. The file is closed again.

Note

nbackup `-F` operates purely on file level and can therefore also be performed without a Firebird server running. Any credentials supplied via the `-U`, `-P` or `-FE` parameters are ignored, just as with nbackup `-R`.

Locking raw-device databases

As discussed in *Backing up raw-device databases*, problems can arise if a delta has to be created for a database located on a raw device. Therefore, in Firebird 2.1 and up, nbackup refuses to operate on raw-device databases unless an explicit location for the delta file has been set previously. For the procedure, see *Setting the delta file*, a little further down.

There's also another problem if you lock and copy a raw device: you don't know the actual size of the database! The raw device may be 10 GB, but the database might only take up 200 MB of that space. To prevent having to copy the entire device just to be on the safe side – possibly wasting huge amounts of time and space – Firebird 2.1 has introduced a new parameter for nbackup: `-S`. This parameter is only valid in combination with `-L` and when it is present, nbackup writes the database size in pages to `stdout` after locking the database. Because the size is given in pages, it has to be multiplied by the database page size in order to get the actual number of bytes to be copied. Or, if you use the `dd` copy utility, you could specify the page size as *`(i)bs`* and the output of `nbackup -L -S` as *`count`*.

Setting the delta file

By default, the delta file lives in the same directory as the database itself. The file name is also the same, but with `.delta` appended. This is usually not a problem, but sometimes it is desirable or even necessary to change the location, e.g. when the database is stored on a raw device. Nbackup itself has no provision for setting the location; this must be done through SQL.

Make a connection to the database with any client that allows you to enter your own SQL statements and give the command:

```
alter database add difference file 'path-and-filename'
```

The custom delta file specification is persistent in the database; it is stored in the system table RDB$FILES. To revert to the default behaviour, issue the following statement:

```
alter database drop difference file
```

You can also specify a custom delta location while creating a new database:

```
create database 'path-and-dbname' difference file 'path-and-deltaname'
```

Notes

- If you specify a bare file name with [ADD] DIFFERENCE FILE, the delta will likely *not* be created in the same directory as the database, but in the current directory as seen from the server. On Windows this may e.g. be the system directory. The same logic applies to relative paths.
- The entire directory path must already exist. Firebird doesn't attempt to create any missing directories.
- If you want to change your custom delta specification, you must first DROP the old one and then ADD the new one.

Appendix A: Document history

The exact file history is recorded in the `manual` module in our CVS tree; see http://firebird.cvs.sourceforge.net/viewvc/firebird/manual/src/docs/firebirddocs/nbackup.xml?view=log

Revision History

0.1	21 Oct 2005	PV	First edition.
1.0	1 Dec 2006	PV	Removed “beta” reference in edition info. Changed warning against specifying backup file names interactively with nbackup -R. Removed “(or will be)” from first sentence in Document History. Changed `C:\Databases` to `C:\Data` in the examples, just to keep the lines from running out of the shaded `<screen>` areas in the PDF. Added section *Setting the delta file*, and changed section *Read on?* accordingly.
1.1	5 May 2008	PV	*Making and restoring backups*: Added warning about heavy-load risks with nbackup 2.0.0–2.0.3. *Restoring a full backup*: Corrected wrong statement that nbackup will overwrite an existing database if there are no active connections. Changed italic text about interactive restore failure to a Note and mentioned its fix in 2.0.1. *Incremental backups*: Inserted warning that incremental backups are broken in 2.1. *Suppressing database triggers (Firebird 2.1+)*: New section. *Read on?*: Fixed typo (you -> your).
1.2	19 Sep 2011	PV	Document source formatting: Changed max. line length to 100, without open lines. All sections and subsections now have an `id`. *Introduction*: Edited first sentence. *Nbackup features - an overview*: First sentence: groups -> kinds. Edited last para before first subsection: mentioned that only SYSDBA, owner and sometimes OS admins can make a backup. *Nbackup features - an overview :: Limitations of nbackup*: Edited previously last listitem to mention Services Manager. Added listitem about direct file access. Removed last para. *Functions and parameters*: New section. *Making and restoring backups*: Slightly altered last sentence of first para. Extended warning: added info on the role of direct I/O with large databases under Posix. *Making and restoring backups :: Full backups :: Making full backups*: Corrected and extended listitem on *`-U`* and *`-P`* parameters. Added listitems on *`-FE`* parameter (new in 2.5), *`-T`* parameter (new in 2.1) and *`-D`* parameter (new in 2.5, backport to 2.1.4). In listitem starting with “The different parameters”, the parenthesized text now reads (*`-B`*, *`-U`* etc.), because many new parameters have been added.

Making and restoring backups :: A word on the inner workings: Small edit (image -> impression).
Making and restoring backups :: Full backups :: Restoring a full backup: Removed parameters `-U` and `-P` from specification. Added listitem on aliases. Changed separate Note about interactive restore failure back to italic text inside the listitem itself. Added listitem about non-necessity of running server and ignoring credentials.
Making and restoring backups :: Incremental backups: Edited Warning: mentioned fix in 2.1.1.
Making and restoring backups :: Incremental backups :: Restoring incremental backups: Removed parameters `-U` and `-P` from formal syntax and 1st listitem.
Making and restoring backups :: Backing up raw-device databases: New section.
Making and restoring backups :: Suppressing database triggers: Edited and extended this section, but removed the "SYSDBA and owner only" remark.
Making and restoring backups :: Direct I/O (Firebird 2.1.4+): New section.
Making and restoring backups :: Informational options (Firebird 2.5+): New section.
Making and restoring backups :: Backups on remote machines (Firebird 2.5+): New section.
Locking and unlocking: Slightly altered last sentence of second para.
Locking and unlocking :: Locking the database and backing up yourself: Added Comments (para + itemizedlist).
Locking and unlocking :: Restoring a backup made after "nbackup -L": Added info on use of alias and `-Z` to step 2 of procedure. In next para, translated "en" (leftover from Dutch original) -> "and". Added sentence to Note about reading `-F` as Flag-only.
Locking and unlocking :: Locking raw-device databases: New section.
Locking and unlocking :: Under the hood: Edited Note.
Setting the delta file: 1st para largely rewritten; now refers to raw-device databases. Split off last sentence into a para of its own. Added info (para + programlisting) about setting delta with CREATE DATABASE. 1st listitem in Notes: ADD -> [ADD].
Document history: Changed ulink to CVS (both text and url); now points directly to document.
License notice: End year in copryright mention now 2011.

Appendix B: License notice