Firebird 3 Quick Start Guide

IBPhoenix Editors
Firebird Project members

25 April 2016, document version 5.2 — covers Firebird 3

Table of Contents

About this guide

The *Firebird Quick Start Guide* is an introduction for the complete newcomer to a few essentials for getting off to a quick start with a Firebird binary kit. The guide first saw the light as Chapter 1 of the *Using Firebird* manual, sold on CD by IBPhoenix. Later it was published separately on the Internet. In June 2004, IBPhoenix donated it to the Firebird Project. Since then it is maintained, and regularly updated, by members of the Firebird documentation project.

> **Important**
>
> Before you read on, verify that this guide matches your Firebird version. This document covers Firebird 3. For all other Firebird versions, get the corresponding Quick Start Guide at http://www.firebirdsql.org/en/documentation/.

The Firebird licenses

Firebird is a free, open-source database management system, but “free” does not mean that everything is permitted. The use of Firebird is governed by two licenses: the IPL (InterBase Public License) and the IDPL (Initial Developer's Public License). The first one covers the parts of the source code that were inherited from InterBase; the second applies to the additions and improvements made by the Firebird Project. Both licenses offer similar rights and restrictions. In short:

- Use of the software is free, even for commercial purposes. You may also redistribute the software, separately or with a product of your own, but you may not claim ownership or credit for it. Any license notices included with Firebird must remain intact.
- You may modify and recompile the Firebird source code or parts of it. You may distribute such modified versions, but if you do so, you *must* document your modifications and make them publicly available, at no cost, under the same license as the original code.
- You may include Firebird source code (modified or not) in a larger work and distribute that larger work, in source and/or compiled form, under a license of your own choosing. You need not publicize the source code for the entire larger work, but you *must* fulfill the license conditions for the parts that were taken from Firebird, whether they were modified or not.

Please notice that the above is a simplified overview. Only the original license texts are legally binding. You can find them here:

http://www.firebirdsql.org/ipl/ (IPL)

http://www.firebirdsql.org/idpl/ (IDPL)

Installing Firebird

The instructions given below for the installation of Firebird on Windows and Linux should be sufficient for the vast majority of cases. However, if you experience problems or if you have special needs not covered here, be sure to read the Release Notes. This is especially important if you are upgrading from a previous version or if there are remnants of an old (and maybe long gone) InterBase or Firebird installation floating around your system (DLLs, Registry entries, environment variables...)

Installation kits

At the Firebird website – http://firebirdsql.org – the installation kits have names like:

`Firebird-3.0.0.`*`bbbbb`*`_`*`p`*`_x64.exe` (Windows executable installer)

`Firebird-3.0.0.`*`bbbbb`*`-`*`p`*`_x64.zip` (Windows zip kit for manual installation)

`Firebird-3.0.0.`*`bbbbb`*`_`*`p`*`_Win32.exe` (Windows executable installer, 32 bits)

`Firebird-3.0.0.`*`bbbbb`*`-`*`p`*`.amd64.rpm` (Linux RPM kit)

`Firebird-3.0.0.`*`bbbbb`*`-`*`p`*`.amd64.tar.gz` (Linux compressed tarball)

`Firebird-3.0.0.`*`bbbbb`*`-`*`p`*`.i686.rpm` (Linux RPM kit, 32 bits)

`Firebird-3.0.0.`*`bbbbb`*`-x86_64.pkg` (Mac OS-X 64-bit package)

etc.

...where *bbbbb* is the build number (32483 for the initial 3.0.0 release) and *p* the packaging number (usually 0 or another low one-digit number).

Firebird 3 packages will also undoubtedly wind up in various Linux distributions and their online repositories. These will have their own naming schemes.

Installing the Firebird server

Before installation

It is almost always advisable to uninstall any previous Firebird installations completely (*after* you've read the next paragraph!) and also hunt the Windows system dirs for old copies of `gds32.dll` and `fbclient.dll`. If you're using Linux, the uninstall scripts should have removed any copies and/or symlinks in `/usr/lib[64]`, but it won't hurt to look if anything named `libfbclient.*` or `libgds.*` is still lying around.

Furthermore, you should be aware that Firebird 3 won't open databases that were created by older versions. So before taking down your existing setup, you should back up all your databases in order that you can restore them later under Firebird 3.

You may also want to back up your old security database `security2.fdb`. Firebird 3 comes with an SQL script `security_database.sql` (located in `misc/upgrade/security`) that will upgrade the old security database to Firebird 3, preserving all information *except* SYSDBA's and except any passwords. For more information, see *Compatibility Issues* :: *Upgrading a v.2.x Security Database* in the Firebird 3 Release Notes.

Installation drives

The Firebird server – and any databases you create or connect to – must reside on a hard drive that is physically connected to the host machine. You cannot locate components of the server, or any database, on a mapped drive, a filesystem share or a network filesystem. (Well, you can, but you shouldn't, and this technique isn't covered here.)

> **Note**
>
> You can mount a read-only database on a CD-ROM drive but you cannot run Firebird server from one.

Installation script or program

Although it is possible to install Firebird by a filesystem copying method – such as untarring a snapshot build or decompressing a structured `.zip` archive – it is strongly recommended that you use the distributed release kit (`.exe` for Windows, `.rpm` for Linux), especially if this is the first time you install Firebird. The Windows installation executable, the Linux rpm program and the `install.sh` script in the official `.tar.gz` for various Posix platforms all perform some essential setup tasks. Provided you follow the installation instructions correctly, there should be nothing for you to do upon completion but log in and go!

Server modes

Some installers ask you to choose between Classic, SuperClassic and Superserver mode. What are they?

- Classic mode (aka *MultiProcess*) involves a single listening process that spawns off an additional process for each client connection. Using a locking mechanism, it allows shared connections to database files.
- SuperClassic (*ThreadedShared*) is a single server process. Client connections are handled by separate threads, each having their own database page cache. Other processes (e.g. embedded servers) may open the same database simultaneously (hence the *Shared*).
- Superserver (*ThreadedDedicated*) is also a single server process with threads handling client connections. There is a single, common database page cache. The server requires exclusive access to each database file it opens (hence the *Dedicated*).

Each mode is fully stable and there is no reason to categorically prefer one to the other. Of course you may have your own specific considerations. When in doubt, just follow the installer default for now. Changing the server mode later can be done via the configuration file `firebird.conf` and requires a restart but not reinstallation. The server mode can even be configured per database (consult the Release Notes for details).

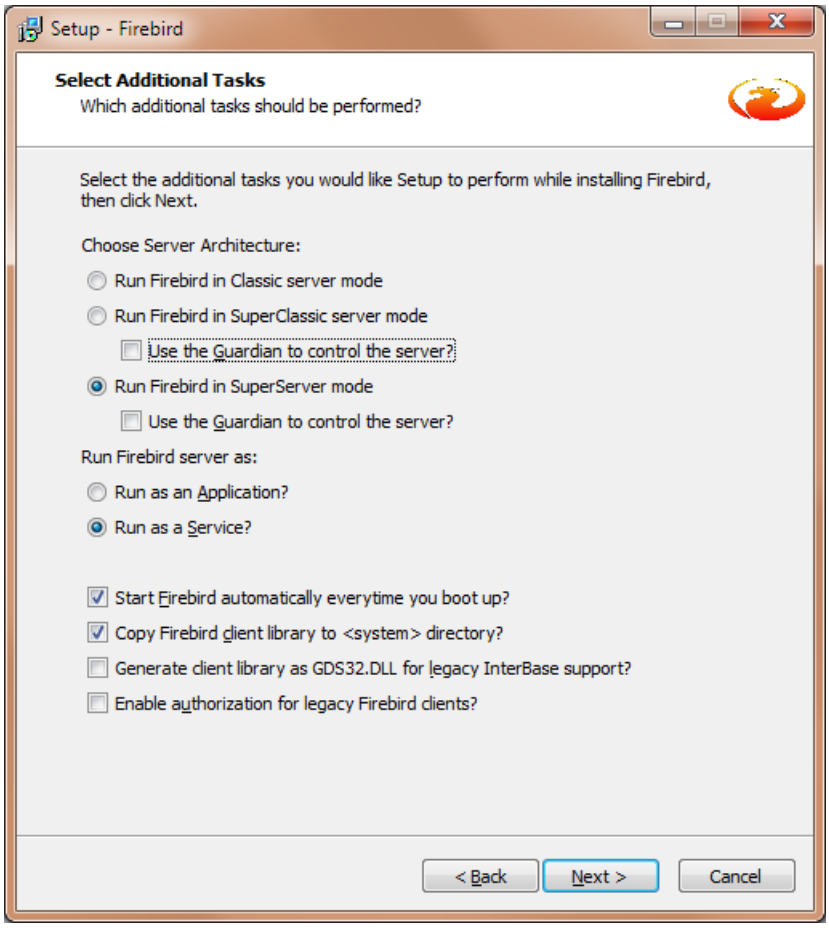
> **Note**
>
> Users of Firebird 2.5 or earlier: please notice that as from Firebird 3, Superserver fully supports the use of multiple processors/cores out of the box, so lack of SMP support is no longer a reason to avoid it.

Installing on Windows

Make sure you run the installer program as Administrator (i.e. right-click on the executable and choose “Run as Administrator”) or you may run into permission problems later!

On Windows server platforms Firebird will run as a system service by default, but during installation you can also choose to let it run as an application. Don't do this unless you have a compelling reason.

The installer will also ask if you want to enable authorization for legacy (i.e. pre-3.0) Firebird clients. If security is a concern (as it should be), don't allow this or allow it only temporarily while you upgrade your existing clients to Firebird 3.0. The legacy connection method sends passwords over the wire unencrypted; it also limits the usable length of the password to 8 characters.



During installation you have the option of providing a password for Firebird's superuser, SYSDBA. Firebird passwords may be up to 255 bytes long, but due to the nature of the hashing algorithm the “effective length” is around 20 bytes, so it's not very useful to enter a password that's much longer than that. Notice however that if you do enter such a password, you must supply it in its full length every time you connect – it won't work if you truncate it to the first 20 characters!

Use the Guardian?

The Firebird Guardian is a utility that monitors the server process and tries to restart it if it terminates abnormally. During a Windows install, you can opt to use the Guardian when running in SuperClassic or Superserver mode. However, since modern Windows systems have the facility to watch and restart services, there is no reason to use the Guardian if Firebird runs as a service (which it should).

The Guardian may be phased out in future versions of Firebird.

Installing on Linux and other Unix-like platforms

In all cases, read the Release Notes for the Firebird version you're going to install. There may be significant variations from release to release of any Posix operating system, especially the open source ones. Where possible, the build engineers for each Firebird version have attempted to document any known issues.

Aside from being packaged with the download kits, Release Notes for all officially released versions of Firebird can also be found at http://www.firebirdsql.org/en/release-notes/.

If you have a Linux distribution that supports rpm installs, consult the appropriate platform documentation for instructions about using RPM Package Manager. In most distributions you will have the choice of performing the install from a command shell or through a GUI interface.

For Linux distributions that cannot process rpm programs, and for Unix flavours for which no `.rpm` kit is provided, use the `.tar.gz` kit. Quite often, installation is just a matter of untarring the archive and running `install.sh`. In some cases, the Release Notes or packed Readmes may instruct you to edit the scripts and make some manual adjustments.

Installing multiple servers

Firebird allows the operation of multiple servers on a single machine. It can also run concurrently with Firebird 1.x or InterBase servers. Setting this up is not a beginner's task though. If you need to run multiple servers on the same machine, the second and subsequent servers must be installed and configured manually. They need to have different service names and should listen on different TCP/IP ports. The file `install_windows_manually.txt` in the `doc` subdir may be of help if you're doing this on Windows, but bear in mind that it was written for Firebird 2.1.

Also read the chapter *Configuring the Port Service on Client and Server* in the Firebird 1.5 (!) Release Notes:

> http://www.firebirdsql.org/file/documentation/release_notes/html/rlsnotes15.html#config-port
>
> http://www.firebirdsql.org/file/documentation/release_notes/Firebird-1.5.6-ReleaseNotes.pdf#page=96

Testing your installation

If you want to connect to your Firebird server across a network, then before testing the Firebird server itself you may want to verify that the server machine is reachable from the client at all. At this point, it is assumed that you will use the recommended TCP/IP network protocol for your Firebird client/server connections. (On Windows networks, NetBEUI is also supported.)

Pinging the server

The **ping** command – available on most systems – is a quick and easy way to see if you can connect to a server machine via the network. For example, if your server's IP address in the domain that is visible to your client is `192.13.14.1`, go to a command shell on the client machine and type the command

ping 192.13.14.1

substituting this example IP address with the IP address that your server is broadcasting. If you are on a managed network and you don't know the server's IP address, ask your system administrator. Of course you can also ping the server by its name, if you know it:

ping vercingetorix

If you are connecting to the server from a local client – that is, a client running on the same machine as the server – you can ping the virtual TCP/IP loopback server:

ping localhost –or– **ping 127.0.0.1**

If you have a simple network of two machines linked by a crossover cable, you can set up your server with any IP address you like except 127.0.0.1 (which is reserved for a local loopback server) and, of course, the IP address which you are using for your client machine. If you know the “native” IP addresses of your network cards, and they are different, you can simply use those.

Once you have verified that the server machine is reachable from the client, you can go on to the next step.

Making sure that the Firebird server is running

Most – but not all – installation packages start up the Firebird server as one of the final steps during installation, and also make sure that Firebird is started at every reboot.

After being launched, the Firebird server should be running:

On Linux or other Unix-like systems:
As a service.

On Windows server systems:
As a service or as an application. Service is default and highly recommended.

The following sections show you how to test the server on each platform.

Server check: Linux and other Unices

Use the **top** command in a command shell to inspect the running processes interactively. If a Firebird 3 server is running, you should see a process named `firebird` and possibly also `fbguard` (the Guardian process).

The following screen shows the output of top, restricted by grep to show only lines containing the string `firebird`:

```
paul@fili ~ $ top -b -n1 | grep [f]irebird
 7169 firebird  20   0   29668    992    560 S   0,0  0,0   0:00.00 fbguard
 7171 firebird  20   0  228160   5876   3048 S   0,0  0,1   0:00.01 firebird
```

As an alternative to top, you can use **ps -ax** or **ps -aux** and pipe the output to grep.

The process name is `firebird` regardless if Firebird is running in Superserver, Classic or SuperClassic mode. However, it is possible to configure a Classic-mode Firebird in such a way that it runs as a service under (x)inetd. In that case, you will only see a `firebird` process if a client connection has been made.

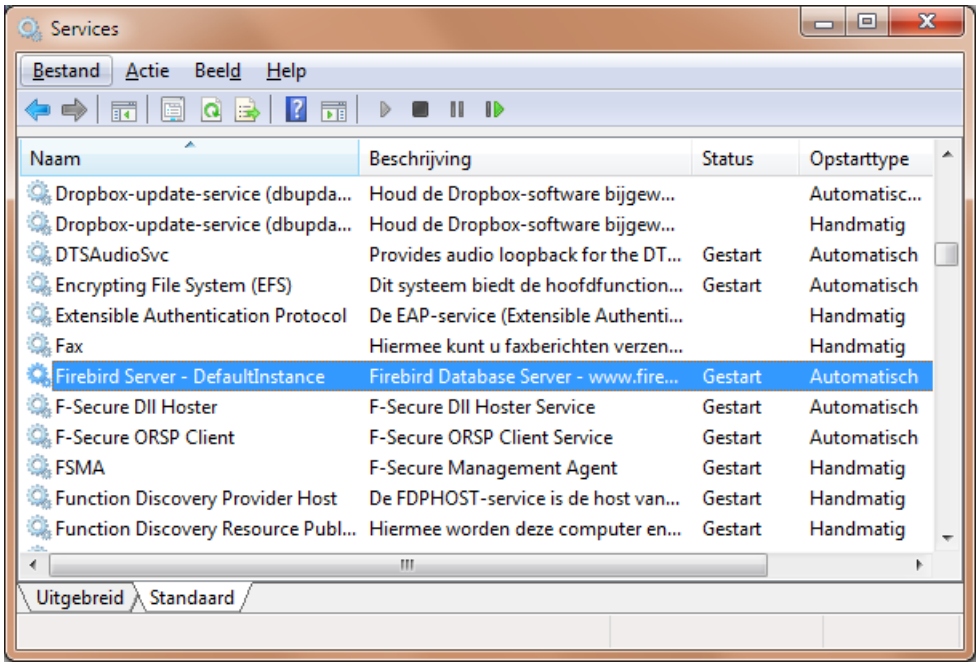
Another way of testing the server after installation is by starting a Firebird client (e.g. `/opt/firebird/bin/isql`) and connecting to a database or creating one. These operations are described later in this guide.

If it turns out that the server hasn't been started after all, you may need to do this manually, e.g. with **/etc/init.d/firebird start** or **systemctl start firebird** and **systemctl enable firebird**, depending on the type of Linux system and your Firebird installation package.

Server check: Windows, running as service

Open Control Panel -> Administrative Tools -> Services.

This illustration shows the Services applet display on Windows 7. The appearance may vary from one Windows server edition to another.



You should at least find the Firebird server in the services listing. The Guardian may or may not be running, depending on the choices you made during installation. If you didn't opt to start the server at the end of the installation process, you may do so now by right-clicking on the Firebird entry (or the Guardian) and choosing Start.

Server check: Windows, running as application

If Firebird is up and running as an application, it is represented by an icon in the system tray:

- A green and grey server symbol if controlled by the Guardian;
- A round yellow and black graphic if running standalone.

A flashing icon indicates that the server is in the process of starting up (or at least trying to do so). A red icon, or an icon with an overlying red stop sign, indicates that startup has failed.

One way to make 100% sure if the server is running or not is to press Ctrl-Alt-Del and look for the `firebird` process (and possibly `fbguard`) in the task list. You may need to check the box “Show processes of all users” for these processes to become visible.

On some occasions, you may need to start the Guardian or server once explicitly via the Start menu even if you opted for “Start Firebird now” at the end of the installation process. Sometimes a reboot is necessary.

You can shut the server down via the menu that appears if you right-click on the tray icon. Notice that this also makes the icon disappear; you can restart Firebird via the Start menu.

> **Note**
>
> In Classic mode (but not SuperClassic!) a new process is launched for every connection, so the number of `firebird` processes will always equal the number of client connections plus one. Shutdown via the tray icon menu only terminates the first process (the *listener*). Other processes, if present, will continue to function normally, each terminating when the client disconnects from the database. Of course, once the listener has been shut down, new connections can't be made.

Performing a client-only install

Each remote client machine needs to have the client library – `libfbclient.so` on Posix clients, `fbclient.dll` on Windows clients – that matches the release version of the Firebird server.

Firebird can install symlinks or copies named after the 1.0 libs (with the “old” InterBase names), to maintain compatibility with third-party products which need these files.

Some extra pieces are also needed for the client-only install.

Windows

At present, no separate installation program is available to install only the client pieces on a Windows machine. If you are in the common situation of running Windows clients to a Linux or other Unix-like Firebird server (or another Windows machine), you need to download the full Windows installation kit that corresponds to the version of Firebird server you install on your server machine.

Fortunately, once you have the kit, the Windows client-only install is a breeze. Just run the installation program and when you arrive at the “Select Components” screen, choose one of the client-only options from the drop-down list or uncheck the “Server Components” checkbox.

Linux and some other Posix clients

A small-footprint client install program for Linux clients is not available either. Additionally, some Posix flavours – even within the Linux constellation – have somewhat idiosyncratic requirements for filesystem locations. For these reasons, not all *x distributions for Firebird even contain a client-only install option.

For most Linux flavours, the following procedure is suggested for a manual Firebird client-only install. Log in as `root` for this.

1. Look for `libfbclient.so.3.0.`*n* (*n* being the patch version number) in `/opt/firebird/lib` on the machine where the Firebird server is installed. Copy it to `/usr/lib` on the client (or `/usr/lib64` if both server and client are 64-bits).

2. Create chained symlinks using the following commands:

   ln -s /usr/lib/libfbclient.so.3.0.*n* /usr/lib/libfbclient.so.2

   ln -s /usr/lib/libfbclient.so.2 /usr/lib/libfbclient.so

   ...replacing *3.0.n* with your version number, e.g. `3.0.0` or `3.0.4`

   If you're running applications that expect the legacy libraries to be present, also create the following symlinks:

   ln -s /usr/lib/libfbclient.so /usr/lib/libgds.so.0

   ln -s /usr/lib/libfbclient.so /usr/lib/libgds.so

3. Copy `firebird.msg` to the client machine, preferably into the `/opt/firebird` directory. If you place it somewhere else, create a system-wide permanent `FIREBIRD` environment variable pointing to the right directory, so that the API routines can locate the messages.

4. Optionally copy some of the Firebird command-line tools – e.g. isql – to the client machine.

Instead of copying the files from a server, you can also pull them out of a Firebird `tar.gz` kit. Everything you need is located in the `/opt/firebird` tree within the `buildroot.tar.gz` archive that's packed inside the kit.

Default disk locations

The tables below show you where you'll find the Firebird files and directories after a standard installation. Please notice that the listings are not exhaustive.

Linux

The following table shows the default component locations of a Firebird installation on Linux. Some of the locations may be different on other Unix-like systems, or on certain Linux distributions.

Table 1. Firebird 3.0 component locations on Linux

Component	File Name	Default Location
Installation directory (referred to hereafter as *$(install)*)	—	`/opt/firebird` (may vary per distribution)
Configuration files	`firebird.conf`, `databases.conf`, etc.	`$(install)`
Release Notes and other documentation	Various files	`$(install)/doc`
Firebird server	`firebird`	`$(install)/bin`
Command-line tools	`isql`, `gbak`, `nbackup`, `gfix`, `gstat`, etc.	`$(install)/bin`
Plugins (new in Firebird 3)	`libEngine12.so`, `libSrp.so`, `libudr_engine.so`, etc.	`$(install)/plugins`
Sample database	`employee.fdb`	`$(install)/examples/empbuild`
UDF libraries	`ib_udf.so`, `fbudf.so`	`$(install)/UDF`
Additional server-side libraries	`libib_util.so`	`$(install)/lib`
Client libraries	`libfbclient.so.3.0.`*n* The usual symlinks (`*.so.2`, `*.so`) are created. Legacy `libgds.*` symlinks are also installed.	`/usr/lib[64]` (actually, the real stuff is in `$(install)/lib`, but you should use the links in `/usr/lib[64]`)

Windows

In the table below, `<ProgramDir>` refers to the Windows programs folder. This is usually "`C:\Program Files`" but may also be a different path, e.g. "`D:\Programmi`". Likewise, `<SystemDir>` refers to the Windows system directory. Be sure to read the notes below the table, especially if you're running Firebird on a 64-bit Windows system.

Table 2. Firebird 3.0 component locations on Windows

Component	File Name	Default Location
Installation directory (referred to hereafter as *$(install)*)	—	`<ProgramDir>\Firebird\Firebird_3_0`
Configuration files	`firebird.conf`, `databases.conf`, etc.	`$(install)`
Release Notes and other documentation	Various files	`$(install)\doc`
Firebird server	`firebird.exe`	`$(install)`
Command-line tools	`isql.exe`, `gbak.exe`, `nbackup.exe`, `gfix.exe`, `gstat.exe`, etc.	`$(install)`
Plugins (new in Firebird 3)	`engine12.dll`, `srp.dll`, `udr_engine.dll`, etc.	`$(install)\plugins`
Sample database	`employee.fdb`	`$(install)\examples\empbuild`
Internationalisation	`fbintl.conf`, `fbintl.dll`	`$(install)\intl`
User-defined function (UDF) libraries	`ib_udf.dll`, `fbudf.dll`	`$(install)\UDF`
Additional server-side libraries	`icu*.dll`, `ib_util.dll`	`$(install)`
Client connection libraries	`fbclient.dll` (with an optional `gds32.dll`, to support legacy apps)	`$(install)` (with an optional copy in `<SystemDir>` – see note below table)
Some necessary Microsoft runtime libs	`msvcp100.dll`, `msvcr100.dll`	`$(install)`
32-bit library versions for use with 64-bit Firebird	`fbclient.dll`, `msvcp100.dll`, `msvcr100.dll`	`$(install)\WOW64` (with an optional copy in `SysWOW64` – see second note below table)

The Windows system directory

A typical location for the Windows system directory – on both 32-bit and 64-bit systems – is `C:\Windows\System32`

If you run Firebird on a 64-bit system, make sure to also read the next note.

Important notice for 64-bit Windows users

On 64-bit Windows systems, the "`Program Files`" directory is reserved for 64-bit programs. If you try to install a 32-bit application into that folder, it will be auto-redirected to a directory which – in English versions – is called "`Program Files (x86)`". In other language versions the name may be different.

In the same vein, the `System32` directory is reserved for 64-bit libraries. 32-bit libraries go into `SysWOW64`. That's right: 64-bit libraries are in `System32`, 32-bit libraries in `SysWOW64`.

If you're not aware of this, you may have a hard time locating your 32-bit Firebird components on a 64-bit Windows system.

(Incidentally, *WOW* stands for *Windows on Windows*. Now you can work out for yourself what *LOL* means.)

Server configuration and management

There are several things you should be aware of – and take care of – before you start using your freshly installed Firebird server. This part of the manual introduces you to some useful tools and shows you how to protect your server and databases.

User management

In Firebird 3, user management is done entirely through SQL commands. Users of previous versions are probably familiar with the gsec utility for this task. It is still present, but deprecated and it won't be discussed here.

Changing the SYSDBA password

One Firebird account is created automatically as part of the installation process: SYSDBA. This account has all the privileges on the server and cannot be deleted. Depending on version, OS, and architecture, the installation program will either

- install the SYSDBA user with the password `masterkey`, or
- ask you to enter a password during installation, or
- generate a random password and store that in the file `SYSDBA.password` within your Firebird installation directory.

If the password is `masterkey` and your server is exposed to the Internet at all – or even to a local network, unless you trust every user with the SYSDBA password – you should change it immediately. Fire up isql or another Firebird client and connect to a database. In this example, the "employee" example database is used, because its alias is always present in a freshly installed Firebird setup:

```
connect localhost:employee user sysdba password masterkey;
```

If you do this in isql, it should respond with:

```
Database: localhost:employee, User: SYSDBA
```

Now alter the sysdba password:

alter user sysdba set password 'Zis4_viZuna83YoYo';

The SET keyword is optional, and instead of USER SYSDBA you can also use CURRENT USER, which always refers to the user you are logged in as.

If the command succeeds, you won't get any feedback. Instead, isql will just print the next "SQL>"-prompt, thus indicating that all is well and your further input is awaited.

Please notice that unlike "regular" user names, Firebird passwords are always case sensitive.

Adding Firebird user accounts

Firebird allows the creation of many different user accounts. Each of them can own databases and also have various types of access to databases and database objects it doesn't own.

Assuming you are connected to a database as SYSDBA, you can add a user account as follows:

create user billyboy password 'TooLongFor8099Comfort'

The full range of user management commands is:

```
CREATE USER name PASSWORD 'password' [<options>] [<tags>];
[CREATE OR] ALTER USER name [SET] [PASSWORD 'password'] [<options>] [<tags>];
ALTER CURRENT USER [SET] [PASSWORD 'password'] [<options>] [<tags>];
DROP USER name;

<options>  ::=  <option> [, <option> ...]
<option>   ::=  {FIRSTNAME | MIDDLENAME | LASTNAME} 'stringvalue'
                   | ACTIVE
                   | INACTIVE

<tags>     ::=  TAGS (<tag> [, <tag> ...])
<tag>      ::=  tagname = 'stringvalue'
                   | DROP tagname
```

Tags are optional key-value pairs that can be freely defined by the user. The key (tag name) must be a valid SQL identifier, the value a non-NULL string of at most 255 bytes.

Only SYSDBA and co-admins can use all these commands. Ordinary users can change their own parameters (such as password, name parts and tags, but not active/inactive) using ALTER USER *name* or ALTER CURRENT USER. It is not possible to change an account name.

Examples:

```
create user dubya password 'Xwha007_noma' firstname 'GW' lastname 'Shrubbery';
create user lorna password 'Mayday_domaka'
    tags (Street = 'Main Street', Number = '888');
alter user benny tags (shoesize = '8', hair = 'blond', drop weight);
alter current user set password 'SomethingEvenMoreSecretThanThis';
alter user dubya set inactive;
drop user ted;
```

The security database

Firebird user accounts are kept in a *security database*, which normally resides in the installation directory and is called `security3.fdb` (alias: `security.db`). Except in the case of so-called embedded connections (more about those later in this guide), connecting to a database always involves the security database, against which

the user credentials are verified. Of course this is done transparently; the user doesn't have to make an explicit connection to the security database.

However, in Firebird 3 this is not the end of the story. Firebird now allows the use of multiple security databases on a system, each security database governing a specific set of databases. A database can even act as its own security database.

Showing how to set this up is outside the scope of this Quick Start Guide. You can find full details in the Release Notes, chapter *Security*. But it is important to realise that *if* a system has multiple security databases, managing user accounts while connected to a database will always affect the accounts in the security database that governs *that* specific database. To be on the safe side, you may want to connect to the security database itself before issuing your user management commands. Connecting to the security database used to be forbidden in recent versions of Firebird, but is now once again possible, albeit by default only locally (which means that even the `localhost` route is blocked).

Appointing co-administrators

Note: What follows here is not essential knowledge for beginners. You can skip it if you like and go on to the Security section.

In Firebird 2.5 and up, SYSDBA (and others with administrator rights) can appoint co-administrators. This is done with the GRANT ADMIN ROLE directive:

```
create user bigbill password 'bigsekrit7foryou' grant admin role;
alter user littlejohn grant admin role;
```

The first command creates user `bigbill` as a Firebird administrator, who can add, alter and drop users. The second command grants administrator privileges to the existing user `littlejohn`.

To revoke administrator privileges from an account, use ALTER USER ... REVOKE ADMIN ROLE.

Notes

- GRANT ADMIN ROLE and REVOKE ADMIN ROLE are not GRANT and REVOKE statements, although they look that way. They are parameters to the CREATE and ALTER USER statements. The actual role name involved here is RDB$ADMIN. This role also exists in regular databases; more about that in a minute.
- Every user who has received administrator rights can pass them on to others. Therefore, there is no explicit WITH ADMIN OPTION.
- Just for completeness, administrators can also grant admin rights to an existing user by connecting to the security database and issuing a regular GRANT statement:

  ```
  grant rdb$admin to littlejohn
  ```

Differences between co-administrators and SYSDBA

- Co-admins can create, alter and drop users, but they have no automatic privileges in regular databases, like SYSDBA has.
- Unlike SYSDBA, co-admins must specify the *`RDB$ADMIN`* role explicitly if they want to exert their rights as system administrator:

  ```
  connect security.db user bigbill password bigsekrit7foryou role rdb$admin
  ```

For reasons explained elsewhere in this guide, connecting to the security database like this may fail if a Superserver is running. On Windows, you may circumvent this by prepending `xnet://` to the database path or alias, but on Posix, you're stuck. The only solution there is to grant the co-admin the RDB$ADMIN role in at least one regular database as well. (A database that uses the security database in question, of course.) This is done in the usual way that roles are granted:

```
grant rdb$admin to bigbill
```

Grantors can be the database owner, SYSDBA, and every other user who has the RDB$ADMIN role in that database and has specified it while connecting. Every RDB$ADMIN member in a database can pass the role on to others, so again there is no WITH ADMIN OPTION. Once the co-admin has obtained the role, he can connect to the (regular) database with it and use the SQL user management commands. It's not the most elegant of solutions, but it works.

Please remember:

The RDB$ADMIN role in a database gives the grantee SYSDBA rights *in that database only*!

- If it is the security database, the grantee can manage user accounts, but has no special privileges in other databases.
- If it is a regular database, the grantee can control that database like he was SYSDBA, but again has no special privileges in other databases, and has no user administration privileges.

Of course it is possible to grant a user the RDB$ADMIN role in several databases, including the security database.

Security

Firebird 3 offers a number of security options, designed to make unauthorised access as difficult as possible. Be warned however that some configurable security features default to the old, “insecure” behaviour inherited from InterBase and earlier Firebird versions, in order not to break existing applications.

It pays to familiarise yourself with Firebird's security-related configuration parameters. You can significantly enhance your system's security if you raise the protection level wherever possible. This is not only a matter of setting parameters, by the way: other measures involve tuning filesystem access permissions, an intelligent user accounts policy, etc.

Below are some guidelines for protecting your Firebird server and databases.

Run Firebird as non-system user
: On Unix-like systems, Firebird already runs as user `firebird` by default, not as `root`. On Windows server platforms, you can also run the Firebird service under a designated user account (e.g. `Firebird`). The default practice – running the service as the `LocalSystem` user – poses a security risk if your system is connected to the Internet. Consult `README.instsvc.txt` in the `doc` subdir to learn more about this.

Change SYSDBA's password
: As discussed before, if your Firebird server is reachable from the network and the system password is `masterkey`, change it.

Don't create user databases as SYSDBA
: SYSDBA is a very powerful account, with full (destructive) access rights to all your Firebird databases. Its password should be known to a few trusted database administrators only. Therefore, you shouldn't use this super-account to create and populate regular databases. Instead, generate normal user accounts, and provide

their account names and passwords to your users as needed. You can do this with the SQL user management commands as shown above, or with any decent third-party Firebird administration tool.

Protect databases on the filesystem level

Anybody who has filesystem-level read access to a database file can copy it, install it on a system under his or her own control, and extract all data from it – including possibly sensitive information. Anybody who has filesystem-level write access to a database file can corrupt it or totally destroy it.

Also, anybody with filesystem-level access to a database can make an embedded connection to it posing as *any* Firebird user (including SYSDBA) without having his credentials checked. This can be especially disastrous if it concerns the security database!

As a rule, only the Firebird server process should have access to the database files. Users don't need, and should not have, access to the files – not even read-only. They query databases via the server, and the server makes sure that users only get the allowed type of access (if at all) to any objects within the database.

As a relaxation of this rule, most Firebird configurations allow users to create and use databases in their own filesystem space and make embedded connections to them. Since these are *their* files and *their* data, one may argue that unrestricted and possibly destructive access should be their own concern, not yours.

If you don't want or need this relaxation, follow the instructions in the next item.

Disable embedded connections

If you don't want any type of direct access, you may disable embedded mode (= direct filesystem-level access) altogether by opening `firebird.conf` and locating the `Providers` entry. The default (which is probably commented out) is:

```
#Providers = Remote,Engine12,Loopback
```

Now, either remove the hash mark and the `Engine12` provider (this is the one that makes the embedded connections), or – better – add an uncommented line:

```
Providers = Remote,Loopback
```

The `Remote` provider takes care of remote connections; the `Loopback` provider is responsible for TCP/IP connections via `localhost`, as well as (on Windows) WNET/NetBEUI and XNET connections to databases on the local machine. All these connection types require full authentication and have the server process, not the user process, open the database file.

Please notice that you can also set the *`Providers`* parameter on a per-database basis. You can set a default in `firebird.conf` as shown above, and then override it for individual databases in `databases.conf` like this:

```
bigbase = C:\Databases\Accounting\Biggus.fdb
{
  Providers = Engine12,Loopback
}
```

The first line defines the *alias* (see next item), and everything between the curly brackets are parameters for that specific database. You'll find `databases.conf` in the same directory as `firebird.conf`. Refer to the Release Notes, chapter *Configuration Additions and Changes*, section *Per-database Configuration*, for more information about the various parameters.

Use database aliases

Database aliases hide physical database locations from the client. Using aliases, a client can e.g. connect to "`frodo:zappa`" without having to know that the real location is `frodo:/var/firebird/music/underground/mothers_of_invention.fdb`. Aliases also allow you to relocate databases while the clients keep using their existing connection strings.

Aliases are listed in the file `databases.conf`, in this format on Windows machines:

```
poker = E:\Games\Data\PokerBase.fdb
blackjack.fdb = C:\Firebird\Databases\cardgames\blkjk_2.fdb
```

And on Linux:

```
books = /home/bookworm/database/books.fdb
zappa = /var/firebird/music/underground/mothers_of_invention.fdb
```

Giving the alias an `.fdb` (or any other) extension is fully optional. Of course if you do include it, you must also specify it when you use the alias to connect to the database.

Aliases, once entered and saved, take effect immediately. There is no need to restart the server.

Restrict database access

The *`DatabaseAccess`* parameter in `firebird.conf` can be set to `Restrict` to limit access to explicitly listed filesystem trees, or even to `None` to allow access to aliased databases only. Default is `Full`, i.e. no restrictions.

Note that this is not the same thing as the filesystem-level access protection discussed earlier: when *`DatabaseAccess`* is anything other than `Full`, the server will refuse to open any databases outside the defined scope even if it has sufficient rights on the database files.

Choose your authentication method(s)

Firebird supports three authentication methods when connecting to databases:

1. *Srp (Secure Remote Password):* The user must identify him/herself with a Firebird username and password, which the server checks against the security database. The maximum effective password length is around 20 bytes, although you may specify longer passwords. Wire encryption is used.

2. *Win_Sspi (Windows Security Support Provider Interface):* The user is logged in automatically with his Windows account name.

3. *Legacy_Auth:* Insecure method used in previous Firebird versions. Passwords have a maximum length of 8 bytes and are sent unencrypted across the wire. Avoid this method if possible.

Two configuration parameters control Firebird's authentication behaviour:

- *`AuthServer`* determines how a user can connect to the local server. It is usually "`Srp`" or, on Windows machines, "`Srp`, `Win_Sspi`". In the latter case, the user will be authenticated with his Windows login if he fails to supply user credentials (causing the `Srp` method, which is tried first, to fail).

- *`AuthClient`* defines how the local client tries to authenticate the user when making a connection. It is usually "`Srp`, `Win_Sspi`, `Legacy_Auth`", allowing the user to connect to pre-Firebird-3 servers on remote machines.

If `Win_Sspi` and/or `Legacy_Auth` are allowed on the server side, you must also set the *`WireCrypt`* parameter to `Enabled` or `Disabled`, but not `Required`.

Likewise, if a server (not a client!) supports `Legacy_Auth`, the *`UserManager`* parameter must be set to `Legacy_UserManager` instead of `Srp`. (The default `Srp` user manager can still be addressed by adding USING PLUGIN SRP to your user management commands.)

The *`AuthServer`*, *`AuthClient`*, *`WireCrypt`* and *`UserManager`* parameters are all set in `firebird.conf` en can be overridden per database in `databases.conf`.

Please notice: enabling `Win_Sspi` on the server activates the plugin but doesn't grant Windows accounts any type of access to databases yet. Logging in to, say, the `employee` database without credentials (and making sure no embedded connection is made) will result in this error message:

```
SQL> connect xnet://employee;
Statement failed, SQLSTATE = 28000
Missing security context for employee
```

In other words: “We know who you are (because the `Win_Sspi` plugin identified you) but you can't come in.”

The solution is to create, as SYSDBA, a global mapping that gives any Windows account access to databases – but no special privileges – under the same name. This is done with the following command:

```
create global mapping trusted_auth
using plugin win_sspi
from any user to user
```

`Trusted_auth` is just a chosen name for the mapping. You may use another identifier. `From any user` means that the mapping is valid for any user authenticated by the `Win_Sspi` plugin. `To user` indicates that every user will be made known under his own Windows account name in each database he connects to. If instead we had specified `to user bob`, then every Windows user authenticated by the `Win_Sspi` plugin would be `bob` in every database.

With the mapping in effect, the “Windows trusted” connection succeeds:

```
SQL> connect xnet://employee;
Database: xnet://employee, User: SOFA\PAUL
SQL> select current_user from rdb$database;

USER
==============================
SOFA\PAUL
```

Note

With embedded connections, i.e. serverless connections handled by `Engine12`, where the client process directly opens the database file, the user is also logged in under his Windows account name if he doesn't provide a user name when connecting. However, this doesn't require `Win_Sspi` to be enabled, nor does it need any explicit mapping:

```
SQL> connect employee;
Database: employee, User: PAUL
SQL> select current_user from rdb$database;

USER
==============================
PAUL
```

Consider whether Windows administrators should have SYSDBA rights

In Firebird 2.1, if the (now defunct) configuration parameter *`Authentication`* was *trusted* or *mixed*, Windows administrators would automatically receive SYSDBA privileges in all databases, including the security database. In Firebird 2.5 and later, this is no longer the case. This reduces the risk that administrators with little or no Firebird knowledge mess up databases or user accounts.

If you still want to apply the automatic SYSDBA mapping as it was in Firebird 2.1, login as SYSDBA and give the command:

```
create global mapping win_admin_sysdba
using plugin win_sspi
from predefined_group domain_any_rid_admins
to user sysdba
```

This grants all Windows administrators automatic SYSDBA rights in every database (including the security database, so they can manage user accounts), provided that they are authenticated by the `Win_Sspi` plugin. To achieve this, they must connect

- without supplying any user credentials, and
- making sure that the `Engine12` provider doesn't kick in. This is easily achieved with a connection string like `xnet://local-path-or-alias`.

To give just one administrator – or indeed any user – full SYSDBA power, use this command:

```
create global mapping frank_sysdba
using plugin win_sspi
from user "sofa\frank"
to user sysdba
```

The double quotes are necessary because of the backslash in the user name. (Specifying just `frank` will be accepted by Firebird, but won't result in a working mapping on most, if not all, Windows systems.)

You can drop any mapping with the command:

```
DROP [GLOBAL] MAPPING mapping_name
```

E.g.:

```
drop global mapping win_admin_sysdba;
drop global mapping frank_sysdba;
```

The GLOBAL keyword is necessary if it concerns a global mapping and you're not directly connected to the security database where the mapping is registered.

Administration tools

The Firebird kit does not come with a GUI admin tool. It does have a set of command-line tools – executable programs which are located in the `bin` subdirectory of your Firebird installation (on Windows, they are in the installation directory itself). One of them, isql, has already been introduced to you.

The range of excellent GUI tools available for use with a Windows client machine is too numerous to describe here. At least one of them, *FlameRobin*, is also available for Linux.

Explore the Download > Tools > Administration page at http://www.ibphoenix.com for all of the options.

> **Note**
>
> Remember: you can use a Windows client to access a Linux server and vice-versa.

Working with databases

In this part of the manual you will learn:

- how to connect to an existing database,
- how to create a database,
- and some things you should know about Firebird SQL.

In as much as remote connections are involved, we will use the recommended TCP/IP protocol.

Connection strings

If you want to connect to a database or create one you have to supply, amongst other things, a *connection string* to the client application (or, if you are a programmer, to the routines you are calling). A connection string uniquely identifies the location of the database on your computer, local network, or even the Internet.

Local connection strings

An explicit local connection string consists of the path + filename specification in the native format of the filesystem used on the server machine, for example

- on a Linux or other Unix-like server:

```
/opt/firebird/examples/empbuild/employee.fdb
```

- on a Windows server:

```
C:\Biology\Data\Primates\Apes\populations.fdb
```

Many clients also allow relative path strings (e.g. "`..\examples\empbuild\employee.fdb`") but you should use these with caution, as it's not always obvious how they will be expanded. Getting an error message is annoying enough, but applying changes to another database than you thought you were connected to may be disastrous.

Instead of a file path, the local connection string may also be a *database alias* that is defined in `aliases.conf`, as mentioned earlier. The format of the alias depends only on how it's defined in the aliases file, not on the server filesystem. Examples are:

- `zappa`
- `blackjack.fdb`
- `poker`

Upon receiving a local connection string, the Firebird client will first attempt to make a direct, embedded connection to the database file, bypassing authentication but respecting the SQL privileges and restrictions of the supplied user and/or role name. That is, if the `Engine12` provider is enabled in `firebird.conf` or `databases.conf` – which it is by default. If the database file exists but the connection fails because the client process doesn't have the required access privileges to the file, a client-server connection is attempted (by the `Loopback` provider), in this order:

1. Using TCP/IP via `localhost`;
2. On Windows: using WNET (NetBEUI), aka Named Pipes, on the local machine;
3. On Windows: using XNET (shared memory) on the local machine.

You can force Firebird to use a certain protocol (and skip the embedded connection attempt) by prepending the protocol in URL style:

- `inet://zappa` (TCP/IP connection using an alias on the local machine)
- `inet:///opt/firebird/examples/citylife.fdb` (TCP/IP connection using an absolute path on the local Posix machine – notice the extra slash for the root dir)
- `inet://C:\Work\Databases\Drills.fdb` (TCP/IP connection using an absolute path on the local Windows machine)
- `wnet://doggybase` (NetBEUI – named pipes – connection using an alias on the local Windows machine)
- `wnet://D:\Fun\Games.fdb` (NetBEUI – named pipes – connection using an absolute path on the local Windows machine)
- `xnet://security.db` (XNET connection using an alias on the local Windows machine)
- `xnet://C:\Programmas\Firebird\Firebird_3_0\security3.fdb` (XNET connection using the full path on the local Windows machine)

Tip

If your XNET connections fail, it may be because the local protocol isn't working properly on your machine. If you're running Windows Vista, 2003 or XP with terminal services enabled, this can often be fixed by setting *IpcName* to `Global\FIREBIRD` in the configuration file `firebird.conf` (don't forget to uncomment the parameter and restart the server).

If setting *IpcName* doesn't help and you don't get the local protocol enabled, you can usually work around the problem by using `inet://`, `wnet://`, or putting "`localhost:`" before your database paths or aliases, thus turning them into TCP/IP connection strings (discussed below).

TCP/IP connection strings

A TCP/IP connection string consists of:

1. a server name or IP address
2. an optional slash ("`/`") plus port number or service name
3. a colon ("`:`")
4. either the absolute path + filename on the server machine, or an alias defined on the server machine.

Examples:

- On Linux/Unix:

```
pongo:/opt/firebird/examples/empbuild/employee.fdb

bongo/3052:fury

112.179.0.1:/var/Firebird/databases/butterflies.fdb

localhost:blackjack.fdb
```

- On Windows:

```
siamang:C:\Biology\Data\Primates\Apes\populations.fdb

sofa:D:\Misc\Friends\Rich\Lenders.fdb

inca/fb_db:D:\Traffic\Roads.fdb

127.0.0.1:Borrowers
```

Notice how the aliased connection strings don't give any clue about the server OS. And they don't have to, either: you talk to a Linux Firebird server just like you talk to a Windows Firebird server. In fact, specifying an explicit database path is one of the rare occasions where you have to be aware of the difference.

NetBEUI connection strings

A NetBEUI (named pipes) connection string consists of:

1. two backslashes ("`\\`")
2. a server name or IP address
3. an optional at sign ("`@`") plus port number or service name
4. another backslash ("`\`")

5. either the absolute path + filename on the server machine, or an alias defined on the server machine.

Examples:

- On Windows, the exact same databases as in the TCP/IP examples:

```
\\siamang\C:\Biology\Data\Primates\Apes\populations.fdb

\\sofa\D:\Misc\Friends\Rich\Lenders.fdb

\\inca@fb_db\D:\Traffic\Roads.fdb

\\127.0.0.1\Borrowers
```

URL-style connection strings

Local URL-style connection strings have already been introduced.

A remote URL-style connection string consists of:

1. a protocol name (`inet` or `wnet`) followed by a colon and two slashes (“`://`”)
2. a server name or IP address
3. an optional colon (“`:`”) plus port number or service name
4. a slash (“`/`”)
5. either the absolute path + filename on the server machine, or an alias defined on the server machine.

Examples:

- On Linux/Unix:

```
inet://pongo//opt/firebird/examples/empbuild/employee.fdb

inet://bongo:3052/fury

inet://112.179.0.1//var/Firebird/databases/butterflies.fdb

inet://localhost/blackjack.fdb
```

- On Windows:

```
inet://siamang/C:\Biology\Data\Primates\Apes\populations.fdb

inet://sofa:4044/D:\Misc\Friends\Rich\Lenders.fdb

wnet://inca:fb_db/D:\Traffic\Roads.fdb

wnet://127.0.0.1/Borrowers
```

Since XNET is a purely local protocol, you can't have remote connection strings starting with `xnet://`.

Third-party programs

Please be aware that some third-party client programs may have different requirements for the composition of connection strings. Refer to their documentation or online help to find out.

Connecting to an existing database

A sample database named `employee.fdb` is located in the `examples/empbuild` subdirectory of your Firebird installation. It is also reachable under its alias `employee`. You can use this database to “try your wings”.

If you move or copy the sample database, be sure to place it on a hard disk that is physically attached to your server machine. Shares, mapped drives or (on Unix) mounted SMB (Samba) file systems will not work. The same rule applies to any databases that you create or use.

Connecting to a Firebird database requires – implicit or explicit – authentication. In order to work with objects inside the database, such as tables, views and functions, you (i.e. the Firebird user you're logged in as) need explicit permissions on those objects, unless you own them (you own an object if you have created it) or if you're connected as user SYSDBA or with the role RDB$ADMIN. In the example database `employee.fdb`, sufficient permissions have been granted to PUBLIC (i.e. anybody who cares to connect) to enable you to view and modify data to your heart's content.

For simplicity here, we will look at authenticating as SYSDBA using the password `masterkey`. Also, to keep the lines in the examples from running off the right edge, we will work with local databases and use aliases wherever possible. Of course everything you'll learn in these sections can also be applied to remote databases, simply by supplying a full TCP/IP connection string.

Connecting with isql

Firebird ships with a text-mode client named *isql* (Interactive SQL utility). You can use it in several ways to connect to a database. One of them, shown below, is to start it in interactive mode. Go to the directory where the Firebird tools reside (see Default disk locations if necessary) and type **isql** (Windows) or **./isql** (Linux) at the command prompt.

[In the following examples, # means “hit **Enter**”]

```
C:\Programmas\Firebird\Firebird_3_0>isql#
Use CONNECT or CREATE DATABASE to specify a database
SQL>connect xnet://employee user sysdba password masterkey;#
```

Important

- In isql, every SQL statement must end with a semicolon. If you hit **Enter** and the line doesn't end with a semicolon, isql assumes that the statement continues on the next line and the prompt will change from `SQL>` to `CON>`. This enables you to split long statements over multiple lines. If you hit **Enter** after your statement and you've forgotten the semicolon, just type it after the `CON>` prompt on the next line and press **Enter** again.
- If the connection string doesn't start with a host or protocol name, a direct serverless connection to the database is attempted. This may fail if your OS login doesn't have sufficient access rights to the database file. In that case, connect to `localhost:`*`path-or-alias`* or specify a protocol like `xnet://` (Windows only) or `inet://`. Then the server process (usually running as user `firebird` on Posix or `LocalSystem` on Windows) will open the file. On the other hand, network-style connections may fail if a user created the database in direct-access mode and the server doesn't have enough access rights.

> **Note**
>
> You can optionally enclose the path, the user name and/or the password in single (') or double (") quotes. If the path contains spaces, quoting is mandatory. Case-sensitive user names (created like this: `create user "Jantje" password ...`) and user names with spaces, international characters or other "funny stuff" also need to be double-quoted.

At this point, isql will inform you that you are connected:

```
Database: xnet://employee, User: SYSDBA
SQL>
```

You can now continue to play about with the `employee` database. With isql you can query data, get information about the metadata, create database objects, run data definition scripts and much more.

To get back to the OS command prompt, type:

```
SQL>quit;#
```

You can also type `EXIT` instead of `QUIT`, the difference being that `EXIT` will first commit any open transactions, making your modifications permanent.

Connecting with a GUI client

Some GUI client tools take charge of composing the CONNECT string for you, using server, path (or alias), user name and password information that you type into prompting fields. Supply the various elements as described in the preceding topic.

> **Notes**
>
> - It is also quite common for such tools to expect the entire server + path/alias as a single connection string – just like isql does.
> - Remember that file names and commands on Linux and other Unix-like platforms are case-sensitive.

Creating a database using isql

There is more than one way to create a database with isql. Here, we will look at one simple way to create a database interactively – although, for your serious database definition work, you should create and maintain your metadata objects using data definition scripts.

Starting isql

To create a database interactively using the isql command shell, type **isql** (Windows) or **./isql** (Linux) at the command prompt in the directory where the Firebird tools are.

[In the following examples, # means "hit **Enter**"]

```
C:\Programmas\Firebird\Firebird_3_0>isql#
Use CONNECT or CREATE DATABASE to specify a database
```

The CREATE DATABASE statement

Now you can create your new database interactively. Let's suppose that you want to create a database named `test.fdb` and store it in a directory named `data` on your `D` drive:

```
SQL>create database 'D:\data\test.fdb' page_size 8192#
CON>user 'sysdba' password 'masterkey';#
```

> **Important**
>
> - In the CREATE DATABASE statement it is *mandatory* to place quote characters (single or double) around path and password. This is different from the CONNECT statement. Quoting the user name is optional, unless it is case-sensitive or contains spaces, international characters or any other character that is not allowed in an unquoted identifier.
> - If the connection string doesn't start with a host or protocol name, creation of the database file is attempted with your OS login as the owner. This may or may not be what you want (think of access rights if you want others to be able to connect). If you prepend `localhost:` or a protocol to the path or alias, the server process will create and own the file.

The database will be created and, after a few moments, the SQL prompt will reappear. You are now connected to the new database and can proceed to create some test objects in it.

But to verify that there really is a database there, let's first type in this query:

```
SQL>select * from rdb$relations;#
```

Although you haven't created any tables yet, the screen will fill up with a large amount of data! This query selects all of the rows in the system table RDB$RELATIONS, where Firebird stores the metadata for tables. An “empty” database is not really empty: it contains a number of system tables and other objects. The system tables will grow as you add more user objects to your database.

To get back to the command prompt type `QUIT` or `EXIT`, as explained in the section on connecting.

Creating a database as a non-privileged user

In Firebird 3, if you try to create a database other than in embedded mode as someone who is not a Firebird admin (i.e. SYSDBA or an account with equal rights), you may be in for a surprise:

```
SQL>create database 'xnet://D:\data\mydb.fdb' user 'john' password 'lennon';#
Statement failed, SQLSTATE = 28000
no permission for CREATE access to DATABASE D:\DATA\MYDB.FDB
```

Non-admin users must explicitly be granted the right to create databases by a Firebird admin:

```
SQL>grant create database to user john;#
```

After that, they can create databases.

Notice that with a serverless connection, i.e. without specifying a host name or protocol before the database name (and `Engine12` enabled!), Firebird won't deny any CREATE DATABASE statement. It will only fail if the client process doesn't have sufficient rights in the directory where the database is to be created.

Firebird SQL

Every database management system has its own idiosyncrasies in the ways it implements SQL. Firebird adheres to the SQL standard more rigorously than most other RDBMSes. Developers migrating from products that are less standards-compliant often wrongly suppose that Firebird is quirky, whereas many of its apparent quirks are not quirky at all.

Division of an integer by an integer

Firebird accords with the SQL standard by truncating the result (quotient) of an integer/integer calculation to the next lower integer. This can have bizarre results unless you are aware of it.

For example, this calculation is correct in SQL:

```
1 / 3 = 0
```

If you are upgrading from an RDBMS which resolves integer/integer division to a float quotient, you will need to alter any affected expressions to use a float or scaled numeric type for either dividend, divisor, or both.

For example, the calculation above could be modified thus in order to produce a non-zero result:

```
1.000 / 3 = 0.333
```

Things to know about strings

String delimiter symbol

Strings in Firebird are delimited by a pair of single quote (apostrophe) symbols: `'I am a string'` (ASCII code 39, *not* 96). If you used earlier versions of Firebird's relative, InterBase®, you might recall that double and single quotes were interchangeable as string delimiters. Double quotes cannot be used as string delimiters in Firebird SQL statements.

Apostrophes in strings

If you need to use an apostrophe inside a Firebird string, you can “escape” the apostrophe character by preceding it with another apostrophe.

For example, this string will give an error:

```
'Joe's Emporium'
```

because the parser encounters the apostrophe and interprets the string as `'Joe'` followed by some unknown keywords. To make it a legal string, double the apostrophe character:

```
'Joe''s Emporium'
```

Notice that this is TWO single quotes, not one double-quote.

Concatenation of strings

The concatenation symbol in SQL is two “pipe” symbols (ASCII 124, in a pair with no space between). In SQL, the “+” symbol is an arithmetic operator and it will cause an error if you attempt to use it for concatenating strings. The following expression prefixes a character column value with the string “`Reported by: `”:

```
'Reported by: ' || LastName
```

Firebird will raise an error if the result of a string concatenation exceeds the maximum (var)char size of 32 Kb. If only the *potential* result – based on variable or field size – is too long you'll get a warning, but the operation will be completed successfully. (In pre-2.0 Firebird, this too would cause an error and halt execution.)

See also the section below, Expressions involving NULL, about concatenating in expressions involving `NULL`.

Double-quoted identifiers

Before the SQL-92 standard, it was not legal to have object names (identifiers) in a database that duplicated keywords in the language, were case-sensitive or contained spaces. SQL-92 introduced a single new standard to make any of them legal, provided that the identifiers were defined within pairs of double-quote symbols (ASCII 34) and were always referred to using double-quote delimiters.

The purpose of this “gift” was to make it easier to migrate metadata from non-standard RDBMSes to standards-compliant ones. The down-side is that, if you choose to define an identifier in double quotes, its case-sensitivity and the enforced double-quoting will remain mandatory.

Firebird does permit a slight relaxation under a very limited set of conditions. If the identifier which was defined in double-quotes:

1. was defined as all upper-case,
2. is not a keyword, and
3. does not contain any spaces,

...then it can be used in SQL unquoted and case-insensitively. (But as soon as you put double-quotes around it, you must match the case again!)

> **Warning**
>
> Don't get too smart with this! For instance, if you have tables "TESTTABLE" and "TestTable", both defined within double-quotes, and you issue the command:
>
> ```
> SQL>select * from TestTable;
> ```
>
> ...you will get the records from "TESTTABLE", not "TestTable"!

Unless you have a compelling reason to define quoted identifiers, it is recommended that you avoid them. Firebird happily accepts a mix of quoted and unquoted identifiers – so there is no problem including that keyword which you inherited from a legacy database, if you need to.

> **Warning**
>
> Some database admin tools enforce double-quoting of *all* identifiers by default. Try to choose a tool which makes double-quoting optional.

Expressions involving NULL

In SQL, NULL is not a value. It is a condition, or *state*, of a data item, in which its value is unknown. Because it is unknown, NULL cannot behave like a value. When you try to perform arithmetic on NULL, or involve it with values in other expressions, the result of the operation will almost always be NULL. It is not zero or blank or an "empty string" and it does not behave like any of these values.

Below are some examples of the types of surprises you will get if you try to perform calculations and comparisons with NULL.

The following expressions all return NULL:

- `1 + 2 + 3 + NULL`
- `not (NULL)`
- `'Home ' || 'sweet ' || NULL`

You might have expected 6 from the first expression and "`Home sweet `" from the third, but as we just said, NULL is not like the number 0 or an empty string – it's far more destructive!

The following expression:

- `FirstName || ' ' || LastName`

will return NULL if either `FirstName` or `LastName` is NULL. Otherwise it will nicely concatenate the two names with a space in between – even if any one of the variables is an empty string.

> **Tip**
>
> Think of NULL as UNKNOWN and these strange results suddenly start to make sense! If the value of `Number` is unknown, the outcome of '`1 + 2 + 3 + Number`' is also unknown (and therefore NULL). If the content of `MyString` is unknown, then so is '`MyString || YourString`' (even if `YourString` is non-NULL). Etcetera.

Now let's examine some PSQL (Procedural SQL) examples with `if`-constructs:

- ```
  if (a = b) then
    MyVariable = 'Equal';
  else
    MyVariable = 'Not equal';
  ```

  After executing this code, `MyVariable` will be `'Not equal'` if both `a` and `b` are NULL. The reason is that `'a = b'` yields NULL if at least one of them is NULL. If the test expression of an "`if`" statement is NULL, it behaves like `false`: the '`then`' block is skipped, and the '`else`' block executed.
  ```

> **Warning**
>
> Although the expression may *behave* like `false` in this case, it's still `NULL`. If you try to invert it using `not()`, what you get is another `NULL` – not "`true`".

- ```
  if (a <> b) then
    MyVariable = 'Not equal';
  else
    MyVariable = 'Equal';
  ```

  Here, `MyVariable` will be `'Equal'` if `a` is `NULL` and `b` isn't, or vice versa. The explanation is analogous to that of the previous example.

The DISTINCT keyword comes to the rescue!

Firebird 2 and above implement a new use of the DISTINCT keyword allowing you to perform (in)equality tests that take `NULL` into account. The semantics are as follows:

- Two expressions are DISTINCT if they have different values or if one is `NULL` and the other isn't;
- They are NOT DISTINCT if they have the same value or if they are both `NULL`.

Notice that if neither operand is `NULL`, DISTINCT works exactly like the "<>" operator, and NOT DISTINCT like the "=" operator.

DISTINCT and NOT DISTINCT always return `true` or `false`, never `NULL`.

Using DISTINCT, you can rewrite the first PSQL example as follows:

```
if (a is not distinct from b) then
  MyVariable = 'Equal';
else
  MyVariable = 'Not equal';
```

And the second as:

```
if (a is distinct from b) then
  MyVariable = 'Not equal';
else
  MyVariable = 'Equal';
```

These versions will give you the results that a normal (i.e. not SQL-brainwashed) human being would expect, whether there are `NULL`s involved or not.

More about NULLs

A lot more information about `NULL` behaviour can be found in the *Firebird Null Guide*, at these locations:

http://www.firebirdsql.org/manual/nullguide.html (HTML)

http://www.firebirdsql.org/pdfmanual/Firebird-Null-Guide.pdf (PDF)
```

Protecting your data

Backup

Firebird comes with two utilities for backing up and restoring your databases: *gbak* and *nbackup*. Both can be found in the `bin` subdirectory of your Firebird installation. Firebird databases can be backed up while users are connected to the system and going about their normal work. The backup will be taken from a snapshot of the database at the time the backup began.

Regular backups and occasional restores should be a scheduled part of your database management activity.

> **Warning**
>
> Except in nbackup's lock mode, do not use external proprietary backup utilities or file-copying tools such as WinZip, tar, copy, xcopy, etc., on a database which is running. Not only will the backup be unreliable, but the disk-level blocking used by these tools can corrupt a running database.

> **Important**
>
> Study the warnings in the next section about database activity during restores!

More information about gbak can be found here (HTML and PDF version, same content):

http://www.firebirdsql.org/manual/gbak.html

http://www.firebirdsql.org/pdfmanual/Firebird-gbak.pdf

The nbackup manual is here (again same content in HTML and PDF):

http://www.firebirdsql.org/manual/nbackup.html

http://www.firebirdsql.org/pdfmanual/Firebird-nbackup.pdf

How to corrupt a database

The following sections constitute a summary of things *not* to do if you want to keep your Firebird databases in good health.

Disabling forced writes

Firebird is installed with forced writes (synchronous writes) enabled by default. Modifications are written to disk immediately upon posting.

It is possible to configure a database to use asynchronous data writes – whereby modified or new data are held in the memory cache for periodic flushing to disk by the operating system's I/O subsystem. The common term for this configuration is *forced writes off* (or *disabled*). It is sometimes resorted to in order to improve performance during large batch operations.

Disabling forced writes on Windows

The big warning here is: do *not* disable forced writes on a Windows server. It has been observed that the Windows server platforms do not flush the write cache until the Firebird service is shut down. Apart from power interruptions, there is just too much that can go wrong on a Windows server. If it should hang, the I/O system goes out of reach and your users' work will be lost in the process of rebooting.

Disabling forced writes on Linux

Linux servers are safer for running an operation with forced writes disabled temporarily. Still, do not leave it disabled once your large batch task is completed, unless you have a very robust fall-back power system.

Restoring a backup to a running database

One of the restore options in the gbak utility (`gbak -rep[lace_database]`) allows you to restore a gbak file over the top of an existing database. It is possible for this style of restore to proceed without warning while users are logged in to the database. Database corruption is almost certain to be the result.

> **Note**
>
> Notice that the shortest form of this command is `gbak -rep`, not `gbak -r` as it used to be in previous Firebird versions. What happened to `gbak -r`? It is now short for `gbak -recreate_database`, which functions the same as `gbak -c[reate]` and throws an error if the specified database already exists. You can force overwriting of the existing database by adding the `o[verwrite]` flag though. This flag is only supported with `gbak -r`, not with `gbak -c`.
>
> These changes have been made because many users thought that the `-r` switch meant *restore* instead of replace – and only found out otherwise when it was too late.

> **Warning**
>
> Be aware that you will need to design your admin tools and procedures to prevent any possibility for any user (including SYSDBA) to restore to your active database if any users are logged in.

If is practicable to do so, it is recommended to restore to spare disk space using the `gbak -c[reate]` option and test the restored database using isql or your preferred admin tool. If the restored database is good, shut down the old database (you can use the gfix command-line tool for this; see http://www.firebirdsql.org/manual/gfix.html or http://www.firebirdsql.org/pdfmanual/Firebird-gfix.pdf). Make a filesystem copy of the old database just in case and then copy the restored database file(s) over their existing counterparts.

Allowing users to log in during a restore

If you do not block access to users while performing a restore using `gbak -rep[lace_database]` then users may be able to log in and attempt to do operations on data. Corrupted structures will result.

How to get help

The community of willing helpers around Firebird goes a long way back, to many years before the source code for its ancestor, InterBase® 6, was made open source. Collectively, the Firebird community does have all the answers! It even includes some people who have been involved with it since it was a design on a drawing board in a bathroom in Boston.

- Visit the official Firebird Project site at http://www.firebirdsql.org and join the user support lists, in particular `firebird-support`. Look at http://www.firebirdsql.org/en/mailing-lists/ for instructions.
- Use the Firebird documentation index at http://www.firebirdsql.org/en/documentation/.
- Visit the Firebird knowledge site at http://www.ibphoenix.com to look up a vast collection of information about developing with and using Firebird. IBPhoenix also sells a Developer CD with the Firebird binaries and lots of documentation.
- Order the official three-volume *Firebird Book, Second Edition* at http://www.ibphoenix.com/products/books/firebird_book, for more than 1200 pages jam-packed with Firebird information. (*Notice:* at the time of this writing, the *Firebird Book* is not yet up-to-date with Firebird 3.)
- Read the Release Notes for your Firebird version!

How to give help

Firebird exists, and continues to be improved, thanks to a community of volunteers who donate their time and skills to bring you this wonderful piece of software. But volunteer work alone is not enough to keep an enterprise-level RDBMS such as Firebird up-to-date. The *Firebird Foundation* supports Firebird development financially by issuing grants to designers and developers. If Firebird is useful to you and you'd like to give something back, please visit the Foundation's pages and consider making a donation or becoming a member or sponsor.

The Firebird Project

The developers, designers and testers who gave you Firebird and several of the drivers are members of the Firebird open source project at SourceForge, that amazing virtual community that is home to thousands of open source software teams. The Firebird project's address there is http://sourceforge.net/projects/firebird/. At that site are the source code tree, the download packages and a number of technical files related to the development and testing of the code bases.

The Firebird Project developers and testers use an email list forum – `firebird-devel@lists.sourceforge.net` – as their “virtual laboratory” for communicating with one another about their work on enhancements, bug-fixing and producing new versions of Firebird.

Anyone who is interested in watching their progress can join this forum. However, user support questions are a distraction which they do not welcome. Please do not try to post your user support questions there! These belong in the `firebird-support` group.

Happy Firebirding!

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/quickstartguide-2.5.xml?view=log

Revision History

0.0	2002	IBP	Published as Chapter One of *Using Firebird*.
1.0	2003	IBP	Published separately as a free Quick Start Guide.
1.x	June 2004	IBP	Donated to Firebird Project by IBPhoenix.
2.0	27 Aug 2004	PV	Upgraded to Firebird 1.5 Added Classic vs. Superserver section. Reorganised and corrected Disk Locations Table. Added (new) screenshots. Added section on security. Updated and completed information on Control Panel applets. Added more examples to “Expressions involving `NULL`”. Various other corrections and additions.
2.1	20 Feb 2005	PV	Enhanced GSEC section. Added more info to CONNECT and CREATE DATABASE sections. Added version number and document history.
2.1.1	1 Mar 2005	PV	Changed `gbak r[estore]` to `r[eplace]` in two places.
2.1.2	8 Apr 2005	PV	Reordered Firebird SQL subsections. Added links to Firebird Null Guide.
2.2	2 Dec 2005	PV	Removed "Using the books by IBPhoenix" as it doesn't make sense in the QSG. Promoted "How to get help" to 1st-level section and removed "Where to next" shell. Removed link to UFB and RefGuide; added a note instead explaining their current status. Updated/corrected classic-super comparison table. Moved a number of sections on installing, working with databases, and (un)safety into newly created top-level sections.
2.2.1	22 Dec 2005	PV	Corrected statement on SS thread usage in Classic-vs-Superserver table. Fixed broken link.
3.0	21 May 2006	PV	Creation of 2.0 Quick Start Guide, still equal to previous revision except for some version numbers, XML ids etc.
3.2	10 Aug 2006	PV	Promoted “Firebird Project members” to co-authors in articleinfo.

			Updated references to website (`firebird.sourceforge.net` -> `www.firebirdsql.org`). Removed “maturity” and “Service Manager” rows from Classic-vs-Super table; these things are no longer different in Firebird 2. Also changed the row on local connections: CS and SS now both allow safe, reliable local connections on Windows. Added row on Guardian. Prepended a column with feature names. Removed any and all remarks about Classic not having a (full) Service Manager. Removed 2nd paragraph of “Default disk locations” section. Removed notes stating that Classic/Win connections will fail without a host name. Updated location table and inserted rows for documentation. Edited the Installation sections; added sections on Guardian and installing multiple servers. Removed “if-you-do-not-find-the-release-notes” tip. Heavily edited and extended the “Testing your installation” sections. The “Other things you need” section is now gone and its contents distributed across other sections. Added a section on gsec (consisting partly of existing material). Greatly enhanced and extended the *Security* section, and moved it to another location. Extended and improved the “Windows Control Panel applets” section. Edited “Working with databases”. Added a special section on connection strings. Added information on access to database objects, the `EXIT` statement, and local vs. remote connections. Made some paths in the examples relative, to keep the lines short. Extended paragraph on metadata. Weakened the claim that Firebird is more SQL-compliant than any other RDBMS. Changed the “Expressions involving `NULL`” section. Added a subsection on DISTINCT. Changed “More about `NULL`s” subsection somewhat. Renamed “Safety measures” to “Preventing data loss”. The Security subsection has been moved elsewhere. Extended *Backup* section to include nbackup information. Added links to other documentation. In the “How to corrupt...” part, changed gbak -r syntax to -rep and added explanatory note. Added the “IB6 plus rlsnotes” as last-resort option to *How to get help*. Also mentioned firebird-support explicitly. Corrected more version numbers, paths, and stuff. Many sections have been reshuffled, moved up or down the hierarchy, etc. Many smaller modifications are not listed here. Added “Happy Firebirding!” to conclude the last section.
3.3	15 Oct 2006	PV	Default disk locations table: added isql to command line tools; added row for additional server-side libs. Added introductory paragraph to “Installing Firebird”. Changed first sentence of “Installing on Linux...” Changed and extended “Server check: Linux and other Unices”. Corrected and extended the section on Linux client-only installs.

			Security section: moved last paragraph of the “Protect databases...” list-titem into a new item on Classic local mode. Connection strings: improved and extended introductory paragraph; added a subsection on third party program requirements. Changed 3rd and 4th paragraph of “Connecting to an existing database”. Used relative paths in connection examples. Updated/corrected note on the use of quote characters. Edited first “Important” item in “The CREATE DATABASE statement”. Updated the warning about concatenation of long strings. Extended the note in “Restoring a backup to a running database”. Updated last sentence of first paragraph in “The Firebird Project”.
3.4	25 Jan 2007	PV	*About this guide*: Changed note about versions and replaced HTML and PDF links with single link to new doc index page. *Classic or Superserver?*: Replaced note on Embedded Server with a proper subsection, containing more info and links to UFB. *Default disk locations*: Created two subsections (for Linux and Windows); also split table in two and removed first column. Introduced placeholders `<ProgramDir>` and `<SystemDir>`. Changed text around tables, changed existing note, and added note for Win64 users. *Security*: Removed statement that 1.5 Release Notes are included with 2.x packages. *More about `NULL`s*: Replaced note about the Null Guide being updated with a para announcing the availability of the new version. *Backup*: Updated information on UFB. *How to get help*: Updated documentation links and changed text here and there.
3.5	14 Mar 2007	PV	*About this guide* and *Important notice for 64-bit Windows users*: Minor rewordings. *User management: gsec* and *Connection strings*: Added information on enabling local protocol with *`IpcName=Global\FIREBIRD`*. *Security :: Use database aliases*: Changed type from `<database>` to `<literal>` to improve output.
3.6	21 Sep 2007	PV	*About this guide*: Mentioned 2.0.3. Warned against 2.0.2. *Expressions involving `NULL`*: Space added to expected concatenation result: “`Home sweet `”.
3.7	8 Apr 2008	PV	*About this guide*: Added 2.0.4 and 2.1 to covered versions. Mentioned forced writes bug. *Installing the Firebird server :: Use the Guardian?*: Added warning about Win installer not detecting existing server. *How to corrupt a database?*: Gave subsections `id` attributes. *Disabling forced writes on Windows*: Created new parent section *Disabling forced writes*, with the Windows and Linux cases as subsections. Warned against Linux forced writes bug. *License notice*: Copyright end year now 2008.
3.8	18 Jan 2009	PV	*About this guide*: Added 2.0.5 and 2.1.2 to covered versions. *Preventing data loss :: Backup*: Mentioned nbackup's brokenness in 2.1.

			How to corrupt a database :: Restoring a backup to a running database: Changed and extended instructions re. safe restoring. *How to give help*: New section directing reader to the Foundation. *License notice*: Copyright end year now 2009.
3.9	4 Sep 2010	PV	*About this guide*: Added 2.0.6 and 2.1.3 to covered versions. *Working with databases* :: *Creating a database using isql* :: *Starting isql*: Added "# means Enter" note, like in *Connecting with isql*. *Preventing data loss :: Backup*: Mentioned nbackup's still-brokenness in 2.1.1 and complete fix in 2.1.2. *How to get help*: Last list item: changed version ref. `2.0` to `2.x`. *License notice*: Copyright end year now 2010.
4.0	5 Sep 2010	PV	Creation of 2.5 Quick Start Guide, still equal to previous revision except for some version numbers, XML ids etc. Also removed erroneous id from primary index term in *Document History* title.
4.1	15 Sep 2010	PV	*About this guide*: Removed 2.0-specific warnings. Upgraded and added information for version 2.5. [Details still to be filled in]
4.2	21 Sep 2010	PV	*Classic, SuperClassic or Superserver*: Moved the table into an appendix and left only a concise overview here so first-time users can make a reasonable choice. Moved the paragraph about installation packages into a separate subsection. *User management: gsec :: Trouble running gsec*: Improved/corrected first listitem. *Firebird server architectures*: New appendix, containing the (slightly edited) table that used to be in the *Classic, SuperClassic or Superserver* section.
4.3	8 Jul 2011	PV	General: Added IDs to all sections that lacked one. *About this guide*: Changed ulink to Firebird Doc Index. New top-level section: *The Firebird licenses*. *Classic, SuperClassic or Superserver? :: Multiprocessing*: Changed wording. *Classic, SuperClassic or Superserver?*: After differences listing, changed advice from flat-out SuperClassic to SuperClassic on 64-bit and Superserver/Classic on 32-bit systems under high load. Split paragraph into two. *Classic, SuperClassic or Superserver? :: Installation packages*: Added instruction on switching to SuperClassic on Linux. *What is in the kit?* now comes after the *Classic, SuperClassic or Superserver* section. Itemized list now has compact spacing; words "essential reading" now wrapped in `emphasis` element instead of written in all-caps. *Default disk locations :: Linux*: Added that default install dir may vary per distribution. *Installing Firebird :: Testing your installation :: Note*: Updated ulink to IB OpGuide (text and url). *Server configuration and management :: User management: gsec :: Appointing co-administrators :: Differences between co-administrators and SYSDBA*: Changed wording of 2nd listitem. Completely rewrote

			and extended 3rd listitem. In Note, changed wording of 2nd listitem slightly. *Server configuration and management :: Security :: Disable Classic local mode on Linux*: Took out reference to `libfbembed` library. *Server configuration and management :: Windows Control Panel applets :: Firebird Server Manager*: Edited last para. *Server configuration and management :: Windows Control Panel applets :: Firebird Control Center*: Changed almost entire text. *Server configuration and management :: Administration tools*: Updated ulink text and url. *Preventing data loss*: Renamed *Protecting your data*. *Protecting your data :: How to corrupt a database :: Disabling forced writes on Linux*: Removed obsolete indexterm to Linux Forced Writes bug. *How to get help*: Updated ulink to mailing lists page (text and url). Updated ulink to Firebird Doc Index (text and url). Updated link to Firebird Book and added notice about first edition being out of print. Removed Note about *Using Firebird* and *Firebird Reference Guide*. *How to give help*: Updated Firebird Foundation url. *The Firebird Project*: Added terminating slash to ulink (text and url). *Firebird Server Architectures*: In comparison table, added para in Multiprocessor / multicore row for Classic/SuperClassic. Edited final two paragraphs. *Document history*: Link to CVS changed, points directly to document log view in manual module now. *License notice*: Copyright end year now 2011.
4.4	26 Sep 2011	PV	`articleinfo`, *About this guide*: Added 2.5.1 to covered versions. *Classic, SuperClassic or Superserver?*: Fixed typo in first para: Firebid -> Firebird. *Server configuration and management :: User management: gsec :: Appointing co-administrators :: Differences between co-administrators and SYSDBA*: Slightly changed wording of 2nd listitem. *Server configuration and management :: Administration tools*: Inspect -> Explore. In ulink text: Downloads -> Download. *Working with databases :: Firebird SQL :: Expressions involving `NULL` :: The DISTINCT keyword comes to the rescue!*: Wrapped "DISTINCT" in title in database element. Slightly altered wording in 2nd listitem.
5.0	11 Apr 2016	PV	Top-level sections up to and including *Working with databases* extensively reworked to bring them up to date with Firebird 3. Some sections have been moved or completely removed. Mild reworking in other sections.
5.1	14 Apr 2016	PV	*Installing Firebird :: Installing the Firebird server :: Installing on Windows*: corrected erroneous figure dash. *Server configuration and management :: User management :: Adding Firebird user accounts :: Examples*: removed "=" from "`set password =`". *Working with databases :: Connection strings :: URL-style connection strings*: changed indexterm "URL-style" to "URL-style connection strings". Added same indexterm before the listing of local URL-style connection strings in the earlier subsection *Local connection strings*.

5.2	25 Apr 2016	PV	*Installing Firebird :: Installation kits*: changed/corrected the package names and added a para after the blockquote explaining about build and packaging number. *Installing Firebird :: Installing the Firebird server :: Installing on Windows*: updated image link (now points to installer screenshot as it is in final 3.0.0 release). *Installing Firebird :: Installing the Firebird server :: Installing on Linux and other Unix-like platforms*: Removed Note about Firebird 3 still being in RC2 stage. *Server configuration and management :: Security :: Disable embedded connections*: mash mark -> hash mark. *Working with databases :: Creating a database using isql :: Creating a database as a non-privileged user*: Added “other than in embedded mode” to first para. Prepended “`xnet://`” to database path in example creation attempt. Added “(and `Engine12` enabled!)” to last para.

Appendix B: License notice

Alphabetical index

R

S

T

U