

This is Google's cache of <http://homepage.mac.com/david.pugh/FirebirdOSX/FirebirdDatabaseonOSX.html> as retrieved on Jul 31, 2006 15:50:07 GMT.

Google's cache is the snapshot that we took of the page as we crawled the web.

The page may have changed since that time. Click here for the [current page](#) without highlighting.

This cached page may reference images which are no longer available. Click here for the [cached text](#) only.

To link to or bookmark this page, use the following url: <http://www.google.com/search?q=cache:OksyPyhanqMJ:homepage.mac.com/david.pugh/FirebirdOSX/FirebirdDatabaseonOSX.html+%22Firebird+Relational+Database+on+MacOSX%22&hl=en&gl=us&ct=clnk&cd=1&client=sv>

Google is neither affiliated with the authors of this page nor responsible for its content.

These search terms have been highlighted: **firebird relational database on macosx**

Firebird Relational Database on MacOSX

Written by [David Pugh](#) between Oct - Dec, 2004

Table of Contents

- [Firebird Relational Database on MacOSX](#)
 - [1. Introduction](#)
 - [1.1 Why Am I Writing This ?](#)
 - [1.2 Copyright and Caveats](#)
 - [1.3 Why Should I Use Firebird ?](#)
 - [2. Installing Firebird](#)
 - [2.1 Where can I get Firebird ?](#)
 - [2.2 Which version should I get ?](#)
 - [2.3 System Requirements](#)
 - [2.4 Installing Firebird](#)
 - [3. Getting Started With Firebird](#)
 - [3.1 Administration Tools](#)
 - [3.1.1 gsec Security Administrator](#)
 - [3.1.2 isql Interactive SQL Processor](#)
 - [3.2 Creating Your First Database](#)
 - [3.3 Adding Users and Roles](#)
 - [3.4 Testing the Database](#)
 - [4. Advanced Topics](#)
 - [4.1 User Defined Functions \(UDFs\)](#)
 - [4.2 Database Aliases](#)
 - [4.3 Building Firebird Classic 1.5.1 from Source on MacOSX](#)
 - [4.4 Updated - Building Firebird Classic 1.5.2 from Source on MacOSX](#)
 - [4.5 Building Firebird SuperServer 1.5.2 from Source on MacOSX](#)
 - [A. Additional References](#)

1. Introduction

In July 2000, Borland (then Inprise) released the source code of their database product, Interbase, under the Interbase Public License. Firebird is the Open Source direct descendent of that database. For more information about Interbase and Firebird, there is an excellent description of the history of events leading up to the release of Interbase to the Open Source community, and the subsequent development of Firebird.

1.1 Why Am I Writing This ?

Until now, my database of choice has been PostgreSQL running on Linux, and more recently on OSX. Our company was commissioned to build a Java application targeted at the Windows platform. Firebird was chosen as the database because of its wide platform availability (Windows, Linux, BSD etc) . In particular, Firebird is regarded as mature and stable on Windows, the main target market for the application.

To perform my QA tasks for this project, I elected to install Firebird on my Powerbook. While downloading and installing the MacOSX Firebird package was relatively straightforward, I found myself scrabbling around the net looking for information on how to put my installed Firebird to use under MacOSX. It is my hope that this document will give others a headstart with Firebird on OSX.

1.2 Copyright and Caveats

The structure and content of this document is based on the document entitled [Firebird Database on Linux](#) Prepared by Pascal Chong on March 9 2003. The following copyright notice is reproduced verbatim from the original document.

"This document is free documentation; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software

Foundation; either version 2 of the License, or (at your option) any later version. This document is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details."

This document **Firebird Database on OSX** remains subject to the terms of the GNU GPL.

Thank you Pascal .. for your original document, even though it was targetted at linux, helped me up and running with Firebird under OSX !!!

1.3 Why Should I Use Firebird ?

Open Source databases generally suffer from 2 common deficiencies : either they are large, such as PostgreSQL and MySQL, or they lack features and documentation, such as HypersonicSQL or McKoi.

Firebird has a relatively tiny footprint. At about 10MB for the MacOSX 'packaged' version. If your requirement is for an "embedded database", bundled with an application server and/or an application, then it is possible to slim down the required files by approx 5Mb. Firebird has all the common features of more mature databases, such as support for transactions, stored procedures, SQL-compliance, etc. If your background is in DB2 and PostgreSQL, the syntax is very similar, and the data types and data handling may seem very familiar, such as the mandatory "single-quotes" for strings.

The design emphasis for Firebird seems to be on small, fast and minimum management. This is ideal for developers who need a database for storage, but do not want to spend too much time on tuning the database for performance. In many situations you may not even need stored procedures, or do complex table joins. In such cases, Firebird is the ideal compromise between size and functionality.

2. Installing Firebird

2.1 Where can I get Firebird ?

The Firebird website can be reached at <http://firebird.sourceforge.net/index.php>, or alternatively at <http://www.firebirdsql.org/>.

At the time this document was written, the download site for Firebird was at http://sourceforge.net/project/showfiles.php?group_id=9028. Please note that this may have already changed. It is advisable to go to the Firebird website and click on the "Download" link.

2.2 Which version should I get ?

For MacOSX, the download site only has a pre-built binary for the Firebird Classic Server, version 1.5.1. There is no binary available for the Firebird Super Server, however, a method for building and installing the SuperServer version is included in section 4 of this document.

As I was completing this document, a point release 1.5.2 was released. There are no pre-built binaries for OSX yet, but the procedure for building 1.5.2 from source is in section 4 of this document.

The difference between the Classic Server and the Super Server lies in the design approach. The Super Server uses threads to service multiple clients connecting to it at the same time, while the Classic Server was the original approach used in Interbase, which spawns a separate server process for every connection. Under MacOSX, `xinetd` is used to spawn a new process for each client connection to a database. A White Paper comparing the two approaches can be found [here](#).

On MacOSX, the question of which server version to use does not arise as there is only one version available (unless you compile firebird yourself from source). For "embedded databases" with few concurrent users, it should not make much of a difference which one you choose. Theoretically, for applications with a wider audience, the Super Server should perform better by reducing process startup times and better sharing of resources. I have done some limited testing with our application and there is a noticeable increase in performance when the SuperServer version of Firebird is used. This most likely due to the overhead of `xinetd` having to start new processes when using the Classic version.

2.3 System Requirements

The default 1.5.1 package when installed, occupies just over 10 MB of disk space, and should run well on any system capable of running OSX.

The only requirements are :

- OSX 10.2 or above

These instructions were tested on OSX 10.3.4, 10.3.5 and 10.3.6 (panther). It should work on any OSX version 10.2 or above, though, as always, your mileage may vary.

2.4 Installing Firebird

In this example, I will assume you have created a directory called `temp` located in your home directory and all work will be performed in there.

After you have successfully downloaded the zip file from the website, save it into your `temp` directory.

Open a Terminal window, change into the `temp` directory and execute the following command :

```
$ cd temp
~/temp $ unzip Firebird-CS-1.5.1-MacOS.zip
```

This will extract the Firebird insaller package into a directory named `Firebird-CS-1.5.pkg` inside your temp directory.

To launch the MacOSX Installer program and process this package, you can use the `open` command in your Terminal window as follows:

```
~/temp $ open Firebird-CS-1.5.pkg
```

Alternatively, you can use the Finder to navigate to your temp directory. Double-click on the `Firebird-CS-1.5.pkg` directory to launch the MacOSX Installer.

Follow the installer dialogs to complete installation of the Firebird software. This will install Firebird into the directory `/Library/Frameworks/Firebird.framework`

In order to use the command-line Firebird Administration programs such as `gsec` and `isql`, you must run them from the Terminal. You will need to make these binary files available to our Terminal sessions by adding them to our `PATH`. I did this by editing the `.bash_profile` file in my home directory, appending the following lines:

```
export FIREBIRD_HOME=/Library/Frameworks/Firebird.framework/Resources
export PATH=$PATH:$FIREBIRD_HOME/bin
```

Unix people will feel comfortable using an editor such as `vim` or `pico` to make these changes. Alternatively, if you would prefer to use `TextEdit`, you can just type `open ~/.bash_profile` at the command prompt and the file will appear in a `TextEdit` window. Add the above `export` statements to the end of the file, then save your changes.

To verify that your changes have worked, open a new Terminal window, type `isql` then press enter. You should see the following.

```
~/temp $ isql
Use CONNECT or CREATE DATABASE to specify a database
SQL> quit;
Type quit; (dont forget the semi-colon!) to exit the isql program and return to your command prompt.
```

Congratulations. If you have gotten this far, you now have a working Firebird installation running on OSX.

3. Getting Started With Firebird

We will now walk through setting up a sample database and familiarizing ourselves with the operations and administration tools of this database software.

3.1 Administration Tools

The default system administration account has the username `SYSDBA` (this username does not appear to be case-sensitive, when I tested it) and the password `masterkey`. For users of previous versions of Interbase (and people who worked through the Fish Catalog tutorial for Delphi), this will seem very familiar. You will use this account to create another user and the sample database initially.

The administrative tools that are available with the software are :

- `gsec` - This is the security administrator. You will use this command-line tool for creating, modifying and deleting database users, changing passwords, etc.
- `isql` - This is the interactive SQL tool, similar to Oracle's `SQL*Plus` and PostgreSQL's `psql` command. You can use this to test or run SQL queries.

3.1.1 gsec Security Administrator

You will need to run `gsec` as `SYSDBA`. To invoke it, execute the following in your Terminal session:

```
~/temp $ gsec -user sysdba -password masterkey
```

This will bring up the `GSEC>` prompt. You can display current users by typing `display` at the prompt, like so :

```
GSEC> display
```

It is a good idea to change the `SYSDBA` password, because the default is so well-known. To change it, we modify the `SYSDBA` account using the following command :

```
GSEC> modify SYSDBA -pw newpassword
```

Ok, `newpassword` is not exactly a strong password. You should generate your own, which should contain both numbers and letters, and they should be changed frequently. But we will not go into that here.

3.1.2 isql Interactive SQL Processor

As mentioned previously, `isql` is analogous to `psql` for PostgreSQL and `SQL*Plus` for Oracle. You can type in an SQL command and get the query results from the database. Firebird comes with an example `EMPLOYEE` database, and we will use it to test our SQL commands. To begin, execute the following command :

```
~/temp $ isql localhost:/Library/Frameworks/Firebird.framework/Resources/examples/employee.fdb
```

This will connect you to the sample `EMPLOYEE` database and display an `SQL>` prompt. You can type in your SQL commands at the prompt. Remember to put a semicolon (;) at the end of every statement to terminate it, before pressing `<ENTER>` to execute it.

To test, type the following SQL command and press `<ENTER>`

```
SQL> SELECT emp_no, full_name, job_code, job_country FROM employee;
```

This should give you output similar to the following :

EMP_NO	FULL_NAME	JOB_CODE	JOB_COUNTRY
2	Nelson, Robert	VP	USA
4	Young, Bruce	Eng	USA
5	Lambert, Kim	Eng	USA
8	Johnson, Leslie	Mktg	USA
9	Forest, Phil	Mngr	USA
11	Weston, K. J.	SRep	USA
12	Lee, Terri	Admin	USA
14	Hall, Stewart	Finan	USA
15	Young, Katherine	Mngr	USA
20	Papadopoulos, Chris	Mngr	USA
24	Fisher, Pete	Eng	USA
28	Bennet, Ann	Admin	England
29	De Souza, Roger	Eng	USA
34	Baldwin, Janet	Sales	USA

If you wish to see all the tables in the database, type the following :

```
SQL> SHOW TABLES;
```

This will give you all the tables in that database.

COUNTRY	CUSTOMER
DEPARTMENT	EMPLOYEE
EMPLOYEE_PROJECT	JOB
PHONE_LIST	PROJECT
PROJ_DEPT_BUDGET	SALARY_HISTORY
SALES	

To exit from `isql`, simply type `quit`; and press `<ENTER>` .

3.2 Creating Your First Database

So far, we have executed our commands as `sysdba`, and used the default examples provided with the software. Now, we are going to create a database of our own, create a user that will have rights to view and modify the database, and try operating on the database.

To create our database, we will need to use the `isql` tool. Firebird saves its databases under discrete files, and, by convention, the extension is `.fdb`. Note that this is just a convention, and that you can save the database as any extension you wish. For this demonstration, we will first create a database using the `sysdba` user .

We first make sure we are in our `temp` directory, then run the `isql` tool as follows:

```
$ cd ~/temp
~/temp $ isql
```

Then we execute the `CREATE DATABASE` command

```
SQL> CREATE DATABASE 'firstdb.fdb' USER 'sysdba' PASSWORD 'masterkey';
```

This creates a file called `firstdb.fdb` inside the current directory (our `temp` directory). The database is owned by `sysdba`. We will now create a very rudimentary Sales catalog and fill it with data. If you are already familiar with SQL, the following commands should be easily understood. If not, you should probably read up on the ANSI SQL-92 standard.

```
SQL> CREATE TABLE sales_catalog (
CON> item_id varchar(10) not null primary key,
CON> item_name varchar(40) not null,
CON> item_desc varchar(50)
CON> );

SQL> INSERT INTO sales_catalog VALUES('001', 'Aluminium Wok', 'Chinese wok used for stir fry dishes');
SQL> INSERT INTO sales_catalog VALUES('002', 'Chopsticks extra-long', '60-cm chopsticks');
SQL> INSERT INTO sales_catalog VALUES('003', 'Claypot', 'Pot for stews');
SQL> INSERT INTO sales_catalog VALUES('004', 'Charcoal Stove', 'For claypot dishes');
SQL> SELECT * FROM sales_catalog;
```

ITEM_ID	ITEM_NAME	ITEM_DESC
001	Aluminium Wok	Chinese wok used for stir fry dishes
002	Chopsticks extra-long	60-cm chopsticks
003	Claypot	Pot for stews
004	Charcoal Stove For claypot dishes	

To exit `isql`, simply type `quit`; and press `<ENTER>`.

3.3 Adding Users and Roles

We now have a database, but it may not be a good idea to create and administer all databases using the `sysdba` account. In some cases, for example, if I am running multiple databases belonging to different people or groups, I may want each user or group to own their respective database, with no rights to view other databases. Another scenario may be a requirement to create a proxy user that will execute all database operations, but which may not have all the superuser rights of `sysdba`.

In this section we will create a database user, and assign the account viewing and updating rights.

We will need to use the `gsec` utility for this operation. So, supposing we want to create a user called `TestAdmin` with password `testadmin` (I know, I know, another weak password) and give him viewing, modification and deletion rights to `firstdb.fdb`, we will execute the following commands. Note that only the first 8 characters are used for the password.

```
$ gsec -user SYSDBA -password masterkey
GSEC> add TestAdmin -pw testadmin -fname FirstDB -lname Administrator
Warning - maximum 8 significant bytes of password used
GSEC> quit
```

Next, we open the database, create a `firstdbadmin` ROLE for the database, assign the appropriate rights to that role, then add `TestAdmin` to the role.

```
$ isql firstdb.fdb -user SYSDBA -password masterkey
Database: firstdb.fdb, User: SYSDBA
SQL> CREATE ROLE firstdbadmin;
SQL> GRANT SELECT, UPDATE, INSERT, DELETE ON sales_catalog TO ROLE firstdbadmin;
SQL> GRANT firstdbadmin TO TestAdmin;
SQL> quit;
```

Now, we are ready to test our database.

3.4 Testing the Database

First, exit `gsec` and `isql`, if you have not already done so.

We will login to `firstdb.fdb` as user `TestAdmin` with the role `firstdbadmin`, run some queries, then exit. The commands, and the results are shown below :

```
$ isql firstdb.fdb -user TestAdmin -password testadmin -role firstdbadmin
SQL> DELETE FROM sales_catalog;
SQL> INSERT INTO sales_catalog VALUES('001', 'Aluminum Wok', 'Chinese wok');
SQL> INSERT INTO sales_catalog VALUES('002', 'Microwave Oven', '300W Microwave oven');
SQL> INSERT INTO sales_catalog VALUES('003', 'Chopsticks extra-long', '60cm chopsticks');
SQL> SELECT * FROM sales_catalog;
```

ITEM_ID	ITEM_NAME	ITEM_DESC
001	Aluminum Wok	Chinese wok
002	Microwave Oven	300W Microwave oven
003	Chopsticks extra-long	60cm chopsticks

If you encounter any SQL errors at any point, you will need to check with the additional references section at the end of this doc for sources of Firebird info. If everything worked, congratulations ! Your Firebird is now ready to fly !

4 Advanced Topics

These are my observations/fixes/kludges that address some more advanced areas of Firebird on OSX. I'm no OSX or Firebird guru so while these worked for me... your mileage may vary ... if you have a more elegant or even the correct solution, please share it so this document can be improved.

4.1 User Defined Functions (UDFs)

Our development project uses a UDF library called `uudlib` to generate `UUID` and `GUID` unique id strings. This UDF library can be [downloaded here](#) and comes with source code, plus out-of-the-box binaries for Windows and Linux. In order for me to develop/test our project under OSX, I would need to build the `uudlib` library from source, and then configure Firebird-OSX to use my library. This led me on a voyage of discovery which ultimately ended in success, but first some background info

In [this article](#), Roy Nelson describes UDFs as follows

A User Defined Function (UDF) is a mechanism provided to extend the built-in functions InterBase provide. A UDF is written in a "host language" i.e. a language which compiles to libraries usable by InterBase on the host platform. UDF's can be written to provide custom statistical, string, date or performance monitoring functions. Once a UDF is created, it can be used in a database application anywhere that a built-in SQL function can be used. On the NT and Windows 95 platforms native libraries normally take the form of Dynamic Link Libraries or simply DLL's these libraries are loaded by the operating system on a "as needed" basis.

In summary, a UDF library is a shared library, dynamically loaded by Firebird at runtime. Under Windows, shared libraries have a `.DLL` extension. Under Linux, shared libraries have a `.so` extension. However, under OSX, the type of shared library we want has a `.dylib` extension.

The MacOSX Firebird package ships two UDF libraries installed in the directory
`/Library/Frameworks/Firebird.framework/Versions/Current/Resources/English.lproj/var/UDF`

These libraries are contained in the files `fbudf.dylib` and `ib_udf.dylib`

Based on documentation for other platforms, I should be able to load and use the `rpadd()` function from the supplied `ib_udf` library by entering the following SQL fragment into ISQL ..

```
DECLARE EXTERNAL FUNCTION rpadd
CSTRING(80), INTEGER, CSTRING(1)
```

```
RETURNS CSTRING(80) FREE_IT
ENTRY_POINT 'IB_UDF_rpad' MODULE_NAME 'ib_udf';
```

This command appears to succeed and no error messages are printed, however, when trying to use the newly installed `rpad()` function, the following output is produced:

```
SQL> select rpad('test',10, '.') from RDB$DATABASE;
Statement failed, SQLCODE = -902

Access to UDF library "ib_udf.so" is denied by server administrator
SQL>
```

Not the result we expected! This error message is a bit misleading and seems to be a generic message produced whenever there is a problem with a UDF. It suggests a permission problem but in our case, it is because Firebird couldn't load the UDF library because it simply does not exist. Recall that our library file is called `ib_udf.dylib`. Firebird is looking for a file called `ib_udf.so` which it cannot find.

OK, so the first quick fix we try is to copy or rename the `ib_udf.dylib` file to `ib_udf.so` and then try again... Strangely, we get exactly the same error message ...

```
Access to UDF library "ib_udf.so" is denied by server administrator
```

This had me puzzled ... There does now exist a file called `ib_udf.so` in the UDF directory yet still Firebird insists that it cannot be loaded.. To cut a long story short, the solution was found in a configuration file called `firebird.conf` located in the directory `/Library/Frameworks/Firebird.framework/Versions/Current/Resources/English.lproj/var`

Within this file, there is a section that defines the location and access level of UDF libraries. The access level can be one of `'None, Restrict or Full'`. In the case of `Restrict`, a list of paths, separated by semi-colons is required to indicate the search locations for UDF libraries. In the MacOSX Firebird install, the default setting is ...

```
UdfAccess = Restrict UDF
```

which tells Firebird that access to UDF functions is to be Restricted to the directory UDF.

The intention seems to be that this path is relative to the Firebird Install directory, however, under OSX, this does not seem to work. So, I tried adding the full path of the UDF directory as follows:

```
UdfAccess = Restrict UDF;/Library/Frameworks/Firebird.framework/Resources/English.lproj/var/UDF;
```

Now, when we try and use the `rpad()` function we get the following results:

```
SQL> select rpad('test',10, '.') from RDB$DATABASE;

RPAD
=====
test.....
```

SUCCESS !! We can now use UDF's under MacOSX.

Some further information on UDF's can be found at these places ...

- A PDF document on [Extending Interbase with User Defined Functions](#).
- A page with links to a number of UDF Libraries [here](#) however, the pre-built binaries are generally for Windows and/or Linux only..

4.2 Database Aliases

Whenever you connect to a Firebird database, the full path and filename of the database file must be specified. Apart from being inconvenient and non-intuitive, when accessing a database on a remote server, it presents a bit of a security risk. You can make life a lot easier if you use the 'Alias' capability of Firebird as follows:

Use your editor to create the plain text file `/Library/Frameworks/Firebird.framework/Resources/English.lproj/var/aliases.conf`

Then add lines of the format

```
alias = full_path_to_database_file
```

For example ...

```
testdb=/Users/dwp/fbdata/testdb.fdb
```

Now, when using `isql` to access the `testdb` database, instead of ...

```
SQL> connect '/Users/dwp/fbdata/testdb.fdb'
```

you can use ...

```
SQL> connect 'testdb'
```

or to connect via the network socket ...

```
SQL> connect 'localhost:testdb'
```

4.3 Building Firebird Classic 1.5.1 from Source on MacOSX

To build from source, you will have to have the XCode Tools (which includes GCC) installed on your Mac. These are freely available from the [Apple Developer Connection](#) site. Create a free membership account in order to download the developer tools.

My current build environment is:

- Mac OSX Panther (10.3.7)
- XCode 1.5
- gcc (GCC) 3.3 20030304 (Apple Computer, Inc. build 1671)

In order to build Firebird Classic1.5.1 from Source Code on MacOSX, the following steps are required:

1. Download the latest source code package from the [SourceForge Firebird page](#)
In this example, we have downloaded `firebird-1.5.1.4481.tar.bz2`

2. Create a work directory and copy the source package into it.
Our work directory is called `firebird151`

3. Open a Terminal window and change into our work directory

4. Unpack the source code archive using the following command

```
$ tar -jxvf firebird-1.5.1.4481.tar.bz2
```

This will unpack the contents into a subdirectory called `firebird-1.5.1.4481`

Change into the source directory using `cd firebird-1.5.1.4481` before continuing.

5. You need to set a couple of environment variables before proceeding as the default build process looks for the programs `libtoolize` and `libtool`. Under OSX, the `gnu` versions of these programs, `glibtoolize` and `glibtool` must be used instead.:

```
$ export LIBTOOLIZE=glibtoolize
$ export LIBTOOL=glibtool
```

6. Now run the `autogen.sh` shell script which automagically creates and runs the `configure` script to generate the required Makefile.

```
$ ./autogen.sh
```

This will take a wee while to complete. Once it is finished, you should see something like this at the end of the output:

```
The Firebird2 package has been configured with the following options:
```

```
Architecture : ClassicServer
  Debug       : disabled
  64 bit I/O  : enabled
  Raw devices : disabled
  Lock manager : enabled
  Service name : gds_db
  Service port : 3050
  GPRE modules : c_cxx.cpp
```

```
Install Dir : /usr/local/firebird
```

```
Now type `make' to compile Firebird2
```

7. As the last line above says, run `make` to build the software

```
$ make
```

The build takes a while to run ... make a cup of coffee. At the completion of the build process, you should see ...

```
*****
```

```
Build Successful!!
```

```
You can find the installer packages in gen/firebird, and the raw
frameworks in gen/firebird/frameworks
```

```
Run "make install" as root (or via sudo) to install your binary.
```

```
Enjoy
```

8. If you look in the `gen/firebird` directory, you should see something like the following:

```
ahost:~/projects/Firebird/firebird-1.5.1.4481 $ ls -l gen/firebird/
total 2880
drwxr-xr-x  20 dwp  dwp    680 28 Dec 17:27 .
drwxr-xr-x  60 dwp  dwp   2040 28 Dec 17:13 ..
drwxr-xr-x   3 dwp  dwp    102 28 Dec 17:27 Firebird-CS-1.5.pkg
```

```

drwxr-xr-x  6 dwp dwp      204 28 Dec 17:21 Firebird.framework
drwxr-xr-x  4 dwp dwp      136 28 Dec 17:27 UDF
drwxr-xr-x 28 dwp dwp      952 28 Dec 17:26 bin
-rw-rw-rw-  1 dwp dwp    145272 28 Dec 17:21 de_DE.msg
drwxr-xr-x  3 dwp dwp      102 28 Dec 17:12 examples
-rw-rw-rw-  1 dwp dwp    132796 28 Dec 17:21 firebird.msg
-rw-rw-rw-  1 dwp dwp    144216 28 Dec 17:21 fr_FR.msg
drwxr-xr-x  3 dwp dwp      102 28 Dec 17:27 frameworks
drwxr-xr-x  3 dwp dwp      102 28 Dec 17:20 help
drwxr-xr-x  8 dwp dwp      272 28 Dec 17:21 include
drwxr-xr-x  3 dwp dwp      102 28 Dec 17:21 intl
-rw-rw-rw-  1 dwp dwp         0 28 Dec 17:20 isc_init1.PBBear.local
-rw-rw-rw-  1 dwp dwp    262144 28 Dec 17:20 isc_lock1.PBBear.local
-rw-rw-rw-  1 dwp dwp    135656 28 Dec 17:21 ja_JP.msg
drwxr-xr-x 12 dwp dwp       408 28 Dec 17:27 lib
drwxr-xr-x  3 dwp dwp      102 28 Dec 17:21 misc
-rw-rw-rw-  1 dwp dwp    643072 28 Dec 17:27 security.fdb

```

The first entry shows that we have successfully built a MacOSX Installer Package called `Firebird-CS-1.5.pkg`. You can install this package either by double-clicking it in the Finder, or from the commandline by typing

```
open gen/firebird/Firebird-CS-1.5.pkg
```

4.4 Updated - Building Firebird Classic 1.5.2 from Source on MacOSX

My initial attempts to build FB 1.5.2 failed with a variety of error messages. After hacking around a bit, the following steps worked for me.

1. Unpack the source tarball and then change into the firebird source directory.

```
$ tar -jxvf firebird-1.5.2.4731.tar.bz2
$ cd firebird-1.5.2.4731
```

2. Run the clean script to ensure we start from the base build tree.

```
$ yes | ./clean.sh
```

3. Export these environment variables so that the build uses the gnu libtools installed on OSX

```
$ export LIBTOOLIZE=glibtoolize
$ export LIBTOOL=glibtool
```

4. Now, edit the `builds/posix/make.defaults` and `configure.in` as described below:

The Darwin Linker does not recognize the `--version-script` flag for attaching symbols to generated libs. To prevent errors, edit the `builds/posix/prefix.darwin` file and add the following lines to the end of the file.

```
LIB_LINK_MAPFILE=
LINK_FIREBIRD_SYMBOLS=
LINK_FBINTL_SYMBOLS=
```

5. Next, some os-specific directories are not created by the supplied `configure` file. This will cause the build to fail.

You should edit the `configure` file and add the following lines at approximately line number 21277 in the appropriate sections ...

```
mkdir -p temp/client.qli/jrd/os/darwin
mkdir -p temp/client.util/jrd/os/darwin
mkdir -p temp/client.gdef/jrd/os/darwin
mkdir -p temp/embed.util/jrd/os/darwin
mkdir -p temp/embed.gdef/jrd/os/darwin
mkdir -p temp/embed.qli/jrd/os/darwin
```

Note, the `configure` file is a generated file. If you run `autogen.sh` again, this file will be overwritten. to make this change permanent, add the lines to the `configure.in` file before running `autogen.sh`.

6. Now run the `configure` script to produce the platform-specific `Makefile(s)`

```
$ ./configure
```

7. When the script completes, you will see the following output:

```
The Firebird2 package has been configured with the following options:
```

```
Architecture : ClassicServer
Debug : disabled
64 bit I/O : enabled
Raw devices : disabled
Lock manager : enabled
Service name : gds_db
Service port : 3050
GPRES modules : c_cxx.cpp
```

```
Install Dir : /usr/local/firebird
```

Now type `make` to compile Firebird2

8. So, now do as it says and run make

```
$ make
```

The build will chug away for a while before displaying the completion message :

```
*****
Build Successful!!

You can find the installer packages in gen/firebird, and the raw
frameworks in gen/firebird/frameworks

Run "make install" as root (or via sudo) to install your binary.

Enjoy
```

9. You will find the Firebird-CS-1.5.pkg OSX Installer Package in the gen/firebird directory.

Note

On a couple of occasions, the build failed with an error relating to ranlib and the libeditline.a being out of date. If you receive this error, you need to run the ranlib program against the libeditline.a file, then continue the make process as follows:

```
$ ranlib gen/firebird/lib/libeditline.a
$ make
```

4.5 Building Firebird SuperServer 1.5.2 from Source on MacOSX

To build the SuperServer Installer Package, you should first successfully build the CS version as described above. Then,

1. In the firebird source directory, run the configure script with the superserver switch as follows:

```
$ ./configure --enable-superserver
```

At the end of the script execution, you will see the following output:

The Firebird2 package has been configured with the following options:

```
Architecture : SuperServer
  Debug       : disabled
  64 bit I/O  : enabled
  Raw devices : disabled
Service name  : gds_db
Service port  : 3050
GPRE modules : c_cxx.cpp

Install Dir  : /usr/local/firebird
```

2. Now run the make command to build the package

```
$ make
```

<>When the build is complete, you should have a new Installer Package in the gen/firebird directory called Firebird-SS-1.5.pkg

```
$ ls gen/firebird/F*
```

```
Firebird-CS-1.5.pkg  Firebird-SS-1.5.pkg  Firebird.framework
```

3. Now, run the installer package, either by locating it in the Finder and double-clicking, or by using the open utility from the commandline as follows:

```
$ open gen/firebird/Firebird-SS-1.5.pkg
```

4. Upon completion of the installation, a StartupItem for the Firebird SuperServer daemon has been installed in /System/Library/StartupItems/Firebird.

Start the Firebird SuperServer using the following commandline:

```
$ sudo SystemStarter start "Firebird Server"
```

You can check that it is running using the ps command as follows :

```
$ ps aux | grep fb
root  1054  0.0  0.2  28484  2392  ??  S   5:03PM  0:00.01
/Library/Frameworks/Firebird.framework/Resources/English.lproj/var/bin/fbguard -f
root  1055  0.0  0.2  31316  3696  ??  S   5:03PM  0:00.12
/Library/Frameworks/Firebird.framework/Resources/English.lproj/var/bin/fbserver
```

You should see running processes called `fbguard` and `fbserver`

5. From now on, whenever you restart your machine, the Firebird SuperServer will be started automatically.

You can manually stop, start and restart the Firebird SuperServer using the `systemstarter` command in a Terminal as follows:

```
$ sudo SystemStarter stop "Firebird Server"
$ sudo SystemStarter start "Firebird Server"
$ sudo SystemStarter restart "Firebird Server"
```

6. Notes

- o If you have already installed either the Classic or Superserver package, running the installer again will 'Upgrade' it. Whilst this should be safe, I recommend that you backup your existing installation by copying the `/Library/Frameworks/Firebird.framework` directory tree to another location before proceeding with installation. I have safely flipped between Classic/Superserver operation just by 'upgrading' with the required package. Note that during an upgrade, the Firebird Security database is preserved, so any users/password changes you have made remain in effect.
- o Installation of both the Classic and SuperServer packages is not a problem, however, operationally, only the **last installed** version will be used at runtime. For instance, when you install Classic, Firebird is configured to listen for database connections on port 3050 via the `xinetd` daemon. If you later install SuperServer, Firebird is removed from the `xinetd` configuration. You must use `systemstarter` (or restart your machine) to start the Firebird SuperServer daemon which will then listen for database connections on port 3050.

- o To cleanup before a totally fresh install (ie. during development or testing), the following directories must be removed:

```
/Library/Frameworks/Firebird.framework
/System/Library/StartupItems/Firebird
/Library/Receipts/Firebird-XX-1.5.pkg
```

The next installation of the package will then ask you to `Install` instead of `Upgrade`

- o In order for other machines to connect to your Firebird databases, in your `System Preferences`, go to the `Sharing/Firewall` page and open port 3050 to inbound connections. Assess your own security policy before doing this.
- o The SuperServer installation installs a `SystemStarter` script in the location `/System/Library/StartupItems/Firebird/`. According to the docs I have read, the `/System/Library/StartupItems` directory is reserved for use by Mac OSX startup processing. User daemons should instead be installed in `/Library/StartupItems` directory.
- o The SuperServer startup script `/System/Library/StartupItems/Firebird/Firebird` contains the `SYSDBA` password hard-coded in it to enable the Firebird shutdown command to succeed.
 - This may be regarded by some as a security hole
 - If you change the default `SYSDBA` password (masterkey) as recommended in the install docs, then you will also have to edit the startup script and replace the default password with your new `SYSDBA` password
- o The SuperServer startup script runs the Firebird SuperServer processes under the `root` profile.
 - This may be regarded by some as a security risk
 - During installation of the Firebird package, a user called `firebird` was created, however, this user profile is not used.
 - You can modify the startup script to run the SuperServer under the `firebird` profile as follows:

Edit the script `/System/Library/StartupItems/Firebird/Firebird`

Change the `startService` code to use the `su -c` command as follows:

```
su firebird -c "/Library/Frameworks/Firebird.framework/Resources/bin/fbmgr.bin -start"
```

Change `stopService` as so ..

```
su firebird -c "/Library/Frameworks/Firebird.framework/Resources/bin/fbmgr.bin -shut -password masterkey"
```

and `RestartService` as so ...

```
su firebird -c "/Library/Frameworks/Firebird.framework/Resources/bin/fbmgr.bin -shut -password masterkey"
su firebird -c "/Library/Frameworks/Firebird.framework/Resources/bin/fbmgr.bin -start"
```

Save your changes.

- Then you must ensure that the contents of `/Library/Frameworks/Firebird.framework` are owned by the user `firebird` ...


```
$ sudo chown -R firebird:firebird /Library/Frameworks/Firebird.framework/
```
- Finally, you must ensure that any database you wish to create and/or access must be in a location to which the `firebird` user has read/write privileges. The easiest way to do this is to create a special directory to hold all your Firebird databases and assign it's ownership to the `firebird` user.

```
$ cd ~
$ mkdir fbdata
$ sudo chown firebird:firebird fbdata

$ isql -u SYSDBA -p masterkey
Use CONNECT or CREATE DATABASE to specify a database
SQL> CREATE DATABASE '/Users/dwp/fbdata/testdb.fdb';
SQL> quit;
```

If we now look in the `fbdata` directory, we will see our new database has been created by the Firebird SuperServer process and that it is owned by the `firebird` user (ie. the user who owns the SuperServer process)

```
$ ls -l fbdata
total 1168
drwxr-xr-x  3 firebird  firebird    102  2 Jan 20:26 .
drwxr-xr-x 77 dwp      dwp      2618  2 Jan 20:25 ..
-rw-----  1 firebird  firebird 598016  2 Jan 20:28 testdb.fdb
```

A. Additional References

For more information about Firebird operations or SQL commands that it accepts, you can refer to the Interbase v6.0 manuals which are available under [here](#).

- API Guide
- Data Definition Guide
- Developers Guide
- Embedded SQL Guide
- Operations Guide
- Language Reference
- Getting Started

There are altogether 7 manuals and the information seems to have been quite reliable for Firebird 1.0.

The **Operations Guide** and the **Language Reference** provided information for the original version of this HowTo.

The [Firebird website](#) contains many pointers to interesting articles related to the history of Firebird as well as several White Papers which may be interesting to technology managers.

The awesome [IBPhoenix](#) website has a large amount of reference information and links relating to Interbase and Firebird.

A detailed description of the Firebird User and Role based Security Model can be found in [this document](#).

Helen Borrie, a member of the the Firebird Project Team (and fellow Kiwi ... *kia ora Helen* ;-) has written [The Firebird Book](#), at 1100+ pages, this hefty tome contains a huge amount of detail on the Firebird database and is an essential resource for anyone using Firebird in the real world. I don't have a copy of this book, but I did manage to thumb through a copy at Foyles bookshop in London, and I will definitely purchase it on my next UK trip!



This document is licensed under the [CC-GNU GPL](#).
This work was written by [David Pugh](#)