



Getting and building the Firebird manual module

Paul Vinkenoog

24 Oct 2009 – Document version 1.6

Table of Contents

Introduction: Purpose of this Howto – Intended audience	3
What is this "manual module" anyway?	3
Apart from the manual module, is there any other Firebird documentation?	3
Do I really have to build the docs myself? Isn't there an easier way?	3
Getting the manual module from SourceForge	4
What is CVS?	4
CVS clients	4
Checking out the manual module	5
Building the Firebird docs	8
Where to get Java 2	8
Where to get the tools	9
How to set up the environment for the build	9
Building the HTML and PDF docs	10
Keeping your manual module up to date	12
If things go wrong	13
Advanced topic: Improving the PDF	13
How the PDF is built	13
General repair scheme	14
Common problems and their solutions	15
XSL-FO references	19
Appendix A: Document History	20
Appendix B: License notice	23

Introduction: Purpose of this Howto – Intended audience

This Howto explains, step by step, how you can download and build the manual module from the Firebird Project at SourceForge.

What is this "manual module" anyway?

The manual module is part of the Firebird Project at SourceForge. It is a collective effort, aimed at producing comprehensive and accurate documentation on the Firebird RDBMS (Relational Database Management System).

It is important to understand that the manual module contains the documentation in *source form* – to be more precise: in DocBook XML format. These sources need to be processed (*built*) to obtain easily readable docs, which can then be published on the Internet.

Apart from the manual module, is there any other Firebird documentation?

Yes! At the time of this writing, most useful Firebird documentation has been produced outside the Firebird manual module. We still have a long way to go before the documentation in the manual module will be anywhere near complete. In fact, one of the reasons this Howto is written is that it can help would-be docwriters to overcome their first hurdles.

If you are looking for ready Firebird documentation and lots of it, your best starting places are:

- <http://www.firebirdsql.org/?op=doc> – the Firebird Documentation Index.
- <http://www.firebirdsql.org/?op=doc&id=othersites> – a listing of other sites with valuable Firebird documentation.

Do I really have to build the docs myself? Isn't there an easier way?

Sure there's an easier way. As soon as a piece of documentation reaches a certain level of maturity, it is published – in PDF and HTML – on the Firebird website. You can find all the docs we have published via the Firebird Documentation Index (see link above).

You should download the manual module and build the docs yourself if and only if:

- You want to check on the absolutely latest state of the docs. (Be aware though that one of the reasons a version has not yet been published may be that it contains errors.)
- You want to help write documentation yourself.
- You're interested in learning how this doc building stuff works, and/or you think DIY is more fun than an easy file download.

If one of the above applies to your situation, this Howto is for you.

Getting the manual module from SourceForge

The manual module is part of the Firebird CVS repository at SourceForge. In order to download it, you need a piece of software called a *CVS client*. This section describes the necessary steps to get the software and the manual module. The actual doc building will be discussed in the next main section: [Building the Firebird docs](#).

What is CVS?

CVS means Concurrent Versions System. It is a tool to manage software development, useful both for single developers and for teamwork. Practically all projects at SourceForge use CVS to store and develop their code base. The Firebird CVS tree is divided into several so-called *modules*, the manual module being one of them.

CVS clients

Downloading a module from a CVS repository is called *checking out* in CVS lingo. To do that, you need a CVS client; they exist for practically every operating system. Here's a list of CVS clients for some popular OSes:

- Linux, BSD and other Unices
 - Command-line CVS is often pre-installed. If it isn't, use the admin tools of your distribution to install it – you'll typically find it in the development category. If that doesn't work for you, get it at <http://www.cvshome.org>
 - gCvs – a graphical CVS interface, at <http://cvsgui.sourceforge.net/download.html>
- Windows
 - Command-line CVS at <http://www.cvshome.org>
 - WinCVS: Very powerful, but maybe a bit overwhelming if you are new to it. At <http://cvsgui.sourceforge.net/download.html>
 - TortoiseCVS: A Windows Explorer plugin. Brilliantly designed, very intuitive. Lacks some of the more advanced CVS functions, but has everything you need. Get it at <http://tortoisecvs.sourceforge.net/download.shtml>

- Mac OS X
 - Mac OS X comes with command-line CVS already included.
 - Fink (Unix tools for OS X) contains a newer version of CVS. <http://fink.sourceforge.net/download/index.php>
 - MacCvsX at <http://cvsgui.sourceforge.net/download.html> – same project as gCvs and WinCvs. Note: OS X version still in beta.
 - MacCVS Pro at <http://sourceforge.net/projects/maccvspro/>
- Mac Classic
 - MacCvs at <http://cvsgui.sourceforge.net/download.html> – same project as gCvs and WinCvs
 - MacCVS Pro at <http://sourceforge.net/projects/maccvspro/>
- Others
 - Try your luck at <http://www.cvshome.org>, [google](#) for it, or ask in the gnu.cvs.help newsgroup or on the firebird-docs mailing list.

Get one or more of the abovementioned clients and install according to the instructions that come with it. After that, you are ready to check out the manual module.

Checking out the manual module

There are two ways you can check out the manual module: as an anonymous user, or with your SourceForge login name. The last method is often termed SSH checkout (because it uses a Secure SHell) or EXT checkout (because it uses SSH as an EXTERNAL protocol).

Everybody can check out anonymously. Members of the Firebird project can also check out with their SourceForge login. (If you have an SF account but you aren't a Firebird project member, you can only check out anonymously.)

If you are a project member, you'll usually want to check out with your SF login because

- a. An anonymous checkout is not always up to date (delay may be several hours to several days);
- b. If you are going to contribute to the manual module yourself, you *must* checkout with your SF login or you won't be able to commit (= upload) your changes back to the server.

We'll describe both methods in the following subsections, starting with anon checkout. [Click here](#) to jump to the subsection on SSH checkouts (after you have read the warnings below).

Warnings

- If you are behind a firewall, make sure that TCP port 2401 is open. If not, you won't be able to make a cvs connection to the repository. Your session will simply “timeout” trying to connect.
- If at all possible, check out to a local directory path that doesn't contain spaces or other characters that are susceptible to URL-encoding. Otherwise, problems may arise when you build certain types of document. These problems can be overcome, but it's better not to have them at all.

Anonymous checkout

The checkout procedure depends on which client you use. We'll give you the exact instructions for command line CVS, and generic info to be used with other clients.

Anonymous checkout using command line CVS

1. If you are in a graphical environment, open a command window.
2. First, you have to verify the password with the server and store it locally. Give the command:

```
cvs -d:pserver:anonymous@firebird.cvs.sourceforge.net:/cvsroot/firebird login
```

When prompted for a password, just hit Enter, because the password for user "anonymous" is empty. Your CVS client will store the password for later reuse, so you can skip this step in the future, as long as you don't login with other passwords in between.

3. Now you can get the manual module. Give the command (on one line):

```
cvs -z6 -d:pserver:anonymous@firebird.cvs.sourceforge.net:/cvsroot/firebird checkout  
manual
```

If all goes well, a directory called `manual` will be created below your current dir, and the manual source tree will be downloaded into it. You can now build the docs – read further under *Building the Firebird docs*.

Anonymous checkout using other clients

There's too much variation to give exact instructions for all CVS clients here. Consult the help or documentation for your own client. Elements you may need to fill in at some point are:

protocol

this must be set to pserver (password server)

authentication method

same as protocol

server

firebird.cvs.sourceforge.net

repository

/cvsroot/firebird

user name

anonymous

module name

manual

cvsroot

usually a combination of server and repository, sometimes even with user name prepended (e.g. in WinCvs): `firebird.cvs.sourceforge.net:/cvsroot/firebird` or `anonymous@firebird.cvs.sourceforge.net:/cvsroot/firebird`

Fill in the necessary data and give your client's checkout command. If the checkout succeeds, you now have a local copy of the manual module on your computer and are ready to build the docs. Read further under *Building the Firebird docs*.

SSH checkout (with your SF login name)

To be able to perform an SSH checkout, a secure shell client must be present on your system. On most Unix systems, as well as on Mac OS X, OpenSSH is installed by default. If not, check the distribution CDs or get it from <http://www.openssh.com>. This site also contains pointers to other SSH products for OSES not supported by OpenSSH. Note however that installing and configuring SSH on some platforms (including Windows) is not a trivial task. You can make life easier for yourself by choosing a CVS client that has SSH built in, like TortoiseCVS for Windows.

As for the checking out itself: just as with anon checkouts, the procedure depends on which client you use. Again, we'll give exact instructions for command line CVS, and generic info for other clients.

Important

Whatever client you use, do not attempt to check out over an existing local copy, e.g. from a previous anonymous checkout. Always check out into a new or at least empty directory.

Warning

SourceForge may expire your password from time to time. When this happens to you, you will receive a 'Permission denied, please try again' error message. If this does happen, and you are sure that the password is correct, go to the SourceForge.org web page – <http://sourceforge.net/index.php> – and try to login using your account name and the password you are using to connect to the cvs repository. If your password has expired, you will be asked to change it.

After changing your password on SourceForge, you must start a new shell session or you will continue to receive the same error message, even with the new password.

SSH checkout using command line CVS

1. If you are in a graphical environment, open a command window.
2. First, make sure that you have an environment variable called `CVS_RSH` with the value “ssh” (or whatever your SSH client is called). How you must do this depends on your OS. On a DOS/Windows command line, use **set CVS_RSH=ssh**. Under Linux/bash: **export CVS_RSH=ssh**.

You may want to make this envvar permanent so you can skip this step in the future. How to do this depends on your OS. Consult its documentation if necessary.

3. Give the following command to get the manual module (on one line, and substitute *username* with your own SF user name):

```
cvs -z6 -d:ext:username@firebird.cvs.sourceforge.net:/cvsroot/firebird checkout manual
```

Enter your password when prompted for it.

If all goes well, a directory called `manual` will be created below your current dir, and the manual source tree will be downloaded into it. You can now build the docs – read further under *Building the Firebird docs*.

SSH checkout using other clients

There's too much variation to give exact instructions for all CVS clients here. Consult the help or documentation for your own client. Elements you may need to fill in at some point are:

protocol

this must be set to ext (or SSH, or SSL)

authentication method

same as protocol

server

firebird.cvs.sourceforge.net

repository

/cvsroot/firebird

user name

your SourceForge user name

module name

manual

cvsroot

usually a combination of server and repository, sometimes even with user name prepended (e.g. in WinCvs): `firebird.cvs.sourceforge.net:/cvsroot/firebird` or `username@firebird.cvs.sourceforge.net:/cvsroot/firebird`

Fill in the necessary data and give your client's checkout command. If the checkout succeeds, you now have a local copy of the manual module on your computer and are ready to build the docs. You can also commit changes you make locally back to the repository.

Building the Firebird docs

Several Java tools are used to produce the HTML and PDF docs from the DocBook XML source. Therefore, you need a recent version of Java 2 installed on your system.

In the next subsections we will show you:

1. Where to get Java 2
2. Where to get the tools
3. How to set up the environment for the doc build process
4. How to build the HTML and PDF docs

If you already have a recent version of Java 2 installed, you may [skip the first step](#).

Where to get Java 2

Download and install *only one* of the following:

- Java 2 Runtime Environment, Standard Edition – often abbreviated as J2RE SE.

Go to <http://www.java.com/> and follow the links to the download pages. Get the version for your own OS. Clicking on a "Download" or "Get it now" link may lead to windows popping up, asking you if it's OK that Sun Microsystems installs stuff on your computer. If you agree, practically everything will be installed automatically. If that makes you feel uneasy, you can also download the installation program manually and run it yourself.

- Java 2 Software Development Kit, Standard Edition – or J2SDK SE.

This is a much larger package, and it also contains the J2RE SE. If you want the SDK, go to <http://java.sun.com/j2se/> and get the latest stable version. When you have to choose between JRE and SDK, take the SDK. (Yes, you can also get the J2RE from here, but you can get it easier and quicker from the link in the first option.) Download the installation program and run it.

If you don't understand the difference between the two, go for the first option: the Java 2 Runtime Environment. You don't need the SDK to build the Firebird docs.

Where to get the tools

The tools and libraries needed to build the HTML and PDF documents come as Java JARs and ZIP archives, and we used to keep them all in our CVS repository so they landed automatically in the right place when you checked out the manual module. However, it's bad practice to keep large binaries in CVS, especially if their sources are already managed elsewhere (they're all open-source tools). So nowadays we keep these binaries out of CVS and place them on the Firebird website for you to download.

After checking out the manual module, look in the `manual/lib` directory. It contains a file `_readme_libs.txt` with exact instructions on downloading and installing the missing libraries. Likewise, `_readme_tools.txt` in `manual/tools` tells you where to download some required tools.

How to set up the environment for the build

The build scripts need an environment variable `JAVA_HOME` pointing to the Java 2 install directory.

- On Windows, this is typically something of the form `C:\Program Files\Java\j2re1.4.2_01`. To be sure, check if there's a directory called `bin` underneath it, and if this `bin` subdir contains the file `java.exe`
- On Linux, it may be `/usr/lib/java/jre` or `/usr/java/j2sdk`, or... well, it can be a lot of things. The same check applies: it should have a subdir `bin` containing an executable file `java` (without the `.exe` extension here).

If you're lucky, the `JAVA_HOME` envvar is already present and correct. If not, you have to set it yourself, e.g. under Windows with `set JAVA_HOME=C:\Program Files\Java\j2re1.4.2_01` or under Linux/bash with `export JAVA_HOME=/usr/lib/java/jre`. (Note: these paths are just examples; they may or may not be the same as yours.)

Tip: make the `JAVA_HOME` envvar permanent so you won't have to set it again and again. How to do this depends on your OS. Consult its documentation if necessary.

Building the HTML and PDF docs

If you've made it here in one piece, you are finally ready to build the Firebird docs. Here's what to do:

1. If you haven't done so already, this is the moment to read the `ReadMe` file that lives in the `manual` directory. It may contain important information not (yet) included in this Howto.
2. If you are in a graphical environment, open a command window.
3. Unless the `ReadMe` instructs you otherwise, go to the folder `manual/src/build` and give the command

build (in Windows), or

./build.sh (in Linux)

If everything was set up correctly, you now get a number of output lines ending with `BUILD SUCCESSFUL`, and mentioning some *build targets* (things you can build).

4. Now you can build something more substantial, e.g.

build html or

build pdf or

build docs

Whatever you build will wind up in the directory tree under `manual/dist`

Notes

- If you build the PDF target, you will receive tons of error messages. You can safely ignore them, as long as one of the last lines reads `BUILD SUCCESSFUL`.
- Due to limitations in the build software, some PDF files may need manual post-processing before they are presentable. For your own use they're OK though, in the sense that “everything's in there”. If you do want to fix them up, read the topic [Improving the PDF](#) near the end of this guide.

Warning

If your local copy of the manual module is placed in a path that contains spaces or other non-alphanumeric, non-underscore characters, the PDF build may fail because an intermediate file is placed in a newly created path with the same name, except that all the “offending” characters are replaced with their URL-encodings: space becomes `%20`, etc.

The `monohtml` build too (`monohtml` is the same as `html`, but as one big web page) will wind up in the urlencoded equivalent of the path. But the images that go with the page are placed in the original path. All this is caused by a Java class that performs some of the build tasks and converts path names to URLs in the process.

The best way to avoid these problems is to place the manual module in a path that contains only unaccented letters, digits and/or underscores. The second best way is to make the URL-encoded version of the path a symlink to the real path. Once you have set up the symlink, all the future builds will go fine. (This may not work on Windows, however.)

Building non-English sets with `-Dsfx`

To build documentation sets in non-English languages (in so far as they are available) use the `-Dsfx` argument and supply the language code, e.g.:

```
build pdf -Dsfx=es
```

```
build html -Dsfx=fr
```

Non-English output will go into subdirectories like `manual/dist/pdf/ru`, `manual/dist/html/fr`, etc.

If you don't specify `-Dsfx`, the English set will be built.

Warning

Not all language sets contain the same amount of documentation. This depends on docwriters' and translators' activity. Usually, the English set will be the most complete and the most up-to-date.

Building subsets with `-Did`

The examples given so far all produce the entire docset (for one language). Usually, this is not what you want. To build a specific document – e.g. a book or an article – use the `-Did` argument.

With the `-Did` argument, you must supply the ID of the element you want to build, for example:

```
build pdf -Did=fbutils
```

```
build pdf -Dsfx=fr -Did=qsg15-fr
```

How do you know the ID? You can find it in the DocBook XML sources. Look for the `id` attributes on elements such as `book`, `article`, and `chapter`. To find out more about this subject, consult the *Firebird Docwriting Guide*.

As you can see from the last example, command-line arguments can be combined.

Building a different base set with `-Dbase`

Since January 2006, the Firebird Release Notes have been integrated with the manual module, but they constitute a base set of their own, parallel to the default “`firebirddocs`” set. This has given rise to yet another command-line parameter, `-Dbase` (pun intended), whose value should be “`rlsnotes`” to build the Release Notes:

```
build pdf -Dbase=rlsnotes
```

```
build pdf -Dbase=rlsnotes -Did=rlsnotes20
```

```
build pdf -Dbase=rlsnotes -Dsfx=fr
```

Meanwhile, two other base sets have been added: `papers` and `refdocs`.

The output from alternative base sets is written to the same folders as usual, except in one case: the multi-file html target output is placed in `manual/dist/html-<base>`, to avoid mixing files from different base sets and so that the sets' `index.html` files don't overwrite each other. Non-English sets go into `manual/dist/html-<base>/<sfx>`. For instance, the English HTML Release Notes are written to `manual/dist/html-rlsnotes`, the French notes to `manual/dist/html-rlsnotes/fr`.

Rendering `monohtml` and `pdf` into the same folder as the default set causes no problems, as these targets generate single-file outputs, each with a unique filename.

Setting default values in `build.xml`

Do you often find yourself building the same base set, language version, and/or subset? Then you may want to set the corresponding parameter value(s) in the build control file `build.xml`, so you don't have to type them on the command line every time. Instructions can be found near the top of `build.xml`, at the beginning of the `init` target.

If you make use of this feature, you can still build the other stuff by overriding your personal settings at the command line. For instance, if you have set `base` to `rlsnotes` in the build file, you can build the default docset like this:

```
build html -Dbase=firebirddocs
```

Building the docs – conclusion

That's it – you are now a certified Firebird doc builder. Congratulations!

If you want to write or translate docs for the Firebird Project yourself, also read the [Firebird Docwriting Guide](#).

Keeping your manual module up to date

The manual module is a work in progress. Contributors commit changes to it on a regular basis. Some time after your initial checkout, your local copy will be out of sync with the repository at SourceForge. Of course it would be a waste of bandwidth if you had to check out the entire module time and again, only to update those few files that have changed. Moreover, doing so would overwrite any changes you may have made yourself. That's why CVS has an **update** command. With **update**, only the *changes* are downloaded from the server, and your own local changes are preserved. (In the event that another contributor has changed a file in the same spot as you, a conflict is signaled and you must edit the file in question to solve it.)

Updating is dead easy. If you use command line CVS, go to the `manual` directory and type:

```
cvs -z6 update -d
```

This command is the same whether you checked out anonymously or with your SF login and password. CVS knows which server to contact and how to authenticate you because this information is saved in the `manual/` CVS subdirectory, which was created automatically when you first checked out the module. The only difference is that if you checked out with login and pass, you will also be prompted for your password when you run **update**.

If you use another CVS tool, look for its **update** command or menu option.

If things go wrong

If the build process fails, this may be due to a too old Java 2 version. See *Where to get Java 2* for more info on getting the latest version.

Alternatively, if the error message says something about a class – or class definition – not being found, you may not have all the build libraries installed. Consult `manual/lib/_readme_libs.txt` and follow the instructions.

If a PDF build ends with `BUILD SUCCESSFUL` but a couple of lines above it says “No files processed. No files were selected...” and indeed the PDF file isn't there, this may be caused by spaces and/or other “naughty” characters in the file path. See the warning at the end of *Building the HTML and PDF docs*.

If a PDF build succeeds but you find ugly things within the produced document, have a look at the next section: *Improving the PDF*. You may find the solution there.

If anything else goes wrong and you can't get it right, ask for help on the firebird-docs mailing list. Please give a good description of your problem so we can help you better. If you aren't on the firebird-docs mailing list yet, visit <https://lists.sourceforge.net/lists/listinfo/firebird-docs> for information and subscription.

Advanced topic: Improving the PDF

Due to limitations in our build tools, the PDF output may suffer from some irritating defects, such as:

- Widowed headers and titles (appearing at the bottom of the page, with the corresponding text block starting on the next page).
- Page breaks at awkward positions in tables or lists.
- Overly wide horizontal justification spaces.
- Squeezed, truncated or otherwise messed-up page-sized content. This is a new feature, introduced with FOP 0.93.

This part shows you how to deal with these problems, should the need arise.

How the PDF is built

First you have to understand how the PDF is built. Contrary to the HTML generation, this is a two-step process:

1. The DocBook XML source is converted to a Formatting Objects (FO) file. FO – formally called *XSL-FO* – is also an XML format, but unlike DocBook it's presentation-oriented. This step is performed by a so-called *XSL transformer* called Saxon. The output goes into `manual/inter/filename.fo`.
2. Another tool, Apache FOP (*Formatting Objects Processor*), then picks up `filename.fo` and converts it to `filename.pdf`, which is stored in `manual/dist/pdf`.

If you give a **build pdf** command, two consecutive build targets are called internally: **fo** and **fo2pdf**, corresponding to the two steps described above. But you can also call them from the command line. For instance,

build fo -Did=qsg15

...transforms the 1.5 Quick Start Guide source to `manual/inter/qsg15.fo`. And

build fo2pdf -Did=qsg15

...produces the PDF from the FO file (which must of course be present for this step to succeed).

In fact, **build pdf** is just a shortcut for **build fo** followed by **build fo2pdf**.

This setup allows us to edit the FO file manually before generating the final PDF. And that's exactly what we're going to do to fix some of those nasty problems that can spoil our PDFs.

General repair scheme

The general procedure for improving the PDF output by editing the FO file is:

1. Build the PDF once as usual with **build pdf [arguments]**.
2. Start reading the PDF and find the first trouble spot.
3. Open the FO file in an XML or text editor.
4. Find the location in the FO file that corresponds to the trouble spot in the PDF (we'll show you how later).
5. Edit the FO file to fix the problem (we'll show you how later), and save it.
6. Rebuild the PDF, but this time use **build fo2pdf [arguments]**. If you don't, you'll overwrite the changes you've just made to the FO file, get the same PDF as first, and have to start all over again.
7. Check if the problem is really solved and if so, find the next trouble spot in the PDF.
8. Repeat steps 4–7 until you've worked your way through the entire PDF.

Notes

- Although this FO-editing approach suggests that the problem lies in the FO file, this is not the case. The FO file is all right, but Apache FOP doesn't support all the nice features in the XSL-FO specification (yet). With our manual editing, we force the PDF in a certain direction.
- It is important to fix the problems *in document order*. Editing the FO in one spot may lead to vertical adjustments at the corresponding spot in the PDF: more lines, less lines, lines moving to the following page, etc... These adjustments may affect everything that comes after it.

For the same reason, you should always look for the next problem *after* you have fixed the previous one. For instance, don't make a list of all widowed headers in the PDF and then start fixing them all in the FO file. Fixing a widowed header moves all the text below it downward, possibly creating new widowed headers and un-widowing others.

- In general, you can keep the FO file open throughout the process. Just don't forget to save your changes before you rebuild the PDF. You must close the PDF before every rebuild though: once it's opened in Adobe (even in Adobe *Reader*), other processes can't write to it.

- The entire process can be pretty time-consuming, so don't try to fix every tiny little imperfection, especially if you're a beginning FO hacker. In general, only the widowed headers are *really* ugly and make the document look very unprofessional. Fortunately, they have become very rare since we've moved to FOP 0.93.

The next section deals with the various problems and how to solve them.

Common problems and their solutions

- Widowed headers
- Split table rows or list items
- Overly wide horizontal spaces
- Squeezed, truncated or otherwise messed-up page-sized content

Widowed headers

Problem: Headers or titles at the bottom of the page.

Cause: Apache FOP doesn't support the `keep-with-next` attribute everywhere.

Note

Since we've upgraded to Apache FOP 0.93, this problem – which used to be our biggest annoyance – has become **extremely rare**. Yet it may still occur under some circumstances. Or, more in general, there may be a page break you find awkward, e.g. after a line that announces what's to come and ends with a colon. This section helps you solve such cases.

Note that the example used here – a widowed section header – shouldn't occur anymore, but it's still usable to demonstrate the steps you have to take, especially for elements with an `id` attribute.

Solution: Force a page break at the start of the element (often a list, list item or table) that the title or header belongs to.

How: If the element has an `id` attribute (you can see this in the DocBook source), do a search on the `id` in the FO file. For example, suppose that you've just built the *Firebird 2 Quick Start Guide* and you find that the title *Creating a database using isql* is positioned at the bottom of a page. In the DocBook XML source you can see that this is the title of a section whose `id` is `qsg2-databases-creating`. If you search on `qsg2-databases-creating` from the top of the file, your first hit will probably look like this:

```
<fo:bookmark starting-state="hide"
  internal-destination="qsg2-databases-creating">
```

The `fo:bookmark` elements correspond to the links in the navigation frame on the left side of the PDF. So this is not yet the section itself; you'll have to look further. Next find:

```
<fo:block text-align-last="justify" end-indent="24pt"
  last-line-end-indent="-24pt"><fo:inline
  keep-with-next.within-line="always"><fo:basic-link
  internal-destination="qsg2-databases-creating">Creating a database...
```

Here, the `id` is an attribute value in a `fo:basic-link`. We're in the Table of Contents now. Still not there.

The third and fourth finds are often a couple of lines below the second; they serve to create a link from the page number citation in the ToC. But the fifth is usually the one we're looking for (unless there are any more forward links to the section in question):

```
<fo:block id="qsg2-databases-creating">
```

That's it! Most mid- and low-level hierarchical elements in DocBook (preface, section, appendix, para etc.) wind up as a `fo:block` in the FO file. Now we have to tell Apache FOP that it must start this section on a new page. Edit the line like this:

```
<fo:block id="qsg2-databases-creating" break-before="page">
```

Save the change and rebuild the PDF (remember: use **build fo2pdf**, not **build pdf**). The section title will now appear at the top of the following page, and you can move on to the next problem.

When there is no DocBook ID

What if the element has no DocBook id? You'll have to search on (part of) the title/header then. This is a bit trickier, because the title may contain a line break in the FO file, in which case it won't be found. Or the title element has one or more children of its own (e.g. `quote` or `emphasis`). This too will keep you from finding it if you search on the full title. On the other hand: the more you shrink the search term, the higher the probability that you will get a number of unrelated hits. You'll have to use your own judgement here; if there is some characteristic text shortly before or after the title you can also search on that, and try to locate the title in the lines above and below it.

No matter how, once you've found the title, go upward in the FO file until you find the beginning of the section – often identifiable by the auto-generated FO id:

```
</fo:block>
<fo:block id="d0e2340">
  <fo:block>
    <fo:block>
      <fo:block keep-together="always" margin-left="0pc"
        font-family="sans-serif,Symbol,ZapfDingbats">
        <fo:block keep-with-next.within-column="always">
          <fo:block font-family="sans-serif" font-weight="bold"
            keep-with-next.within-column="always"
            space-before.minimum="0.8em" space-before.optimum...
            space-before.maximum="1.2em" color="#404090" hyph...
            text-align="start">
          <fo:block font-size="11pt" font-style="italic"
            space-before.minimum="0.88em" space-before.opti...
            space-before.maximum="1.32em">The DISTINCT keyword
comes to the rescue!</fo:block>
```

As you see, there may be quite a number of lines between the section start and the title text. Notice, by the way, how the title is split over two lines here.

Once you've found the `fo:block` that corresponds to the section start, give it a `break-before="page"` attribute just like we did before.

Why look for the section start and not apply the `break-before` attribute to the `fo:block` immediately enclosing the title? Well, doing the latter will print the title on the next page all right, but links from the Outline and the ToC will point to the previous page, because the “invisible” section start – the block tag bearing the ID – lies before the page break.

As said, the widowed header problem shouldn't occur anymore with sections, but it might still happen to some other objects like tables, figures etc. for which the stylesheets generate ids if you haven't assigned them yourself. In all those cases you can use the approach described above.

There are also numerous DocBook elements – in fact, the majority – for which the stylesheets don't generate ids. Examples are `para`, `informaltable`, the various list types, etc. In those cases, once you have located the text fragment in the FO file, simply apply the `break-before` attribute to the nearest enclosing `fo:block`.

Split table rows or list items

Problem: Table rows or list items split across page boundaries. (DocBook lists are implemented as `fo:tables`.)

Cause: Nothing in particular – there's no rule that forbids page breaks to occur within table rows.

Solution: If you want to keep the row together, insert a hard page break at the start of the row.

How: Find the row by searching on text at the beginning of the row or at the end of the previous row. The element you're looking for is a `fo:table-row`, but don't use that for a search term, because many DocBook elements (not only `<table>`s) are implemented using `fo:tables` and thus contain `fo:table-rows`.

Once the start of the split row is found, add a `break-before` attribute like you did with widowed headers:

```
<fo:table-row break-before="page">
```

Alternatively, you can give the previous row a `break-after` attribute.

Overly wide horizontal spaces

Problem: Very large horizontal justification spaces on lines above a long spaceless string. These large strings are often printed in monospaced (fixed-width) font:

```
WinCvs):          firebird.cvs.sourceforge.net:/cvsroot/firebird          or  
username@firebird.cvs.sourceforge.net:/cvsroot/firebird
```

Cause: Apache FOP often doesn't hyphenate these strings. Therefore, if the string doesn't fit on the line it must be moved to the next line as a whole. This leaves the previous line with “too little” text, making large justification spaces necessary. Note that in the example above, the large spaces on the top line are caused by the string on the line below, not by the one on the line itself.

Solution: You may have good reasons to leave the string unbroken. In that case, accept the wide spaces as a consequence. Otherwise, insert a space (or hyphen-plus-space) at the point where the string should be broken.

How: First find the string in the FO file by searching on (part of) its contents. If it's monospaced in the PDF, you'll almost always find it within a `fo:inline` element. Then look at the PDF and estimate how much of the as yet unbroken string would fit in the large whitespace on the line above. Back in the FO file, insert a space – possibly preceded by a hyphen – in the string at a location where it's acceptable to break it. Rebuild the PDF (**build fo2pdf** !) and check the result. If you've broken the string too far to the right, it will still be entirely on the next line. Too far to the left and the whitespace may still be too wide to your liking. Adjust and rebuild until you're satisfied.

One surprise you may get during this job is that, once you've broken the string in one place, Apache FOP suddenly decides that it's OK to hyphenate the rest of the string. This will leave you with a part of the string on

the first line that contains your own (now erroneous) space but also extends beyond it. You'll now have to delete your space and break the string again at the spot chosen by Apache.

Inserting zero-width spaces

An alternative approach to the wide-spaces problem is to insert zero-width space characters at each and every point where the culprit string may be broken, leaving it to Apache FOP to work out which one is best suited. This is guaranteed to work at the first try, but:

- it's only feasible if you have an editor that lets you insert ZWSPs easily;
- you can only do this in places where it's OK to break the string without a hyphen.

Squeezed, truncated or otherwise messed-up page-sized content

Problem: Tables, figures or other formal objects are truncated or some parts are printed on top of others.

Cause: Formal objects are given a `keep-together.within-page="always"` attribute by the stylesheets. As of FOP 0.93, this attribute is *always* enforced, even if the object is too large to fit on a page. The result: wrecked content that is crammed together on one page.

Solution: There are three alternatives. 1: Use the corresponding *informal* DocBook element instead. 2: Insert a processing instruction in the DocBook source. 3: Remove the attribute from the FO.

How: Two solutions are applied to the DocBook source, the third involves editing the FO file:

- If you don't mind leaving the element titleless, use `informalequation / informalexample / informalfigure / informaltable` instead of their formal counterparts `equation, example, figure` and `table`. These elements don't get the `keep-together` attribute during transformation, so they will be page-broken as necessary.
- If it concerns a table and you want to keep the title, insert a *processing instruction* like this:

```
<table frame="all" id="ufb-about-tbl-features">
  <?dbfo keep-together='auto'?>
  <title>Summary of features</title>
  ...
  (table content...)
  ...
</table>
```

Adding the instruction if you're working in the source text is easy enough. With XMLMind, it's a bit laborious:

1. Place the cursor somewhere in the title or select the entire title element.
2. Choose *Edit -> Processing Instruction -> Insert Processing Instruction Before* from the menu. A green line will appear above the title.
3. Type `keep-together=' auto '` on that line.
4. With the cursor still on the green line, choose *Edit -> Processing Instruction -> Change Processing Instruction Target* from the menu. A dialogue box pops up.
5. In the dialogue box, change `target` to `dbfo` and click OK.

By the way: you can do the opposite with an `informaltable` if you absolutely don't want it broken at page borders. The procedure is the same, except that you must specify `always` instead of `auto`. Be sure that the `informaltable` does fit on one page, though!

We don't have a similar provision for the other formal objects because we probably don't need it. (Things like this require work on our custom stylesheets, so we only implement them if we really feel the need.)

- Ye olde fo-hacking way... open the FO file, locate the element (tip: give it an `id` in the DocBook source so it's easy to find) and remove the `keep-together.within-page="always"` attribute. A disadvantage is that this procedure has to be repeated every time the source changes and a new PDF is built. The other two solutions are persistent.

XSL-FO references

The official XSL-FO (Formatting Objects) page is here: <http://www.w3.org/TR/xsl/>

The Apache FOP homepage is here: <http://xmlgraphics.apache.org/fop/>

The Apache FOP compliance page is here: <http://xmlgraphics.apache.org/fop/compliance.html>. It contains a large object support table where you can look up which XSL-FO objects and attributes (properties) are supported. When consulting the table, please bear in mind that we currently use Apache FOP 0.93 (but with some home-made patches).

Appendix A: Document History

The exact file history is recorded in the `manual` module in our CVS tree; see http://sourceforge.net/cvs/?group_id=9028

Revision History

0.1	2 Nov 2003	PV	First draft published under the title <i>How to get and build the Firebird manual module</i> .
0.2	31 Jan 2004	PV	Entered sources in CVS.
1.0	8 Mar 2004	PV	First official release on Firebird website.
1.1	26 Feb 2005	PV	<i>The following changes have accumulated between March 2004 and Feb. 2005:</i> Added note on error messages during PDF build. Added info on building subsets and non-English sets. Added note on need to post-process PDF builds. Changed title to <i>Getting and building the Firebird manual module</i> . Numerous minor improvements. Added document history and revision number. Licensed this work under the Public Documentation License.
1.1.1	8 April 2005	PV	Added some titleabbrevs for presentational purposes. Contents as such unchanged.
1.2	9 Feb 2006	PV	Removed “Firebird” from title of 2nd section. Added information on where to get the build libraries, since we don't commit those to CVS anymore. Created subsections for build parameters; added information on building other base sets and setting parameters in <code>build.xml</code> . Changed <code>docwritehowto</code> link to <code>ulink</code> , as the articles will be in separate PDFs from now on.
1.2.1	15 May 2006	PV	Replaced <code>cvs.sf.net</code> (3x) and <code>cvs.sourceforge.net</code> (6x) with <code>firebird.cvs.sourceforge.net</code> to reflect new situation at SF. Also added “on one line” above two examples that are now line-wrapped in the PDF.
1.3	17 Jul 2006	PV	Changed all <code>sectN</code> elements to <code>section</code> . Shortened ID of Introduction and assigned IDs to its child sections. Spelling matters: <code>RDMS</code> -> <code>RDBMS</code> , <code>paralell</code> -> <code>parallel</code> , <code>OS'es</code> -> <code>OSes</code> , <code>CD's</code> -> <code>CDs</code> , <code>wil</code> -> <code>will</code> , <code>envvar</code> -> <code>envvar</code> , <code>linewrapped</code> -> <code>line-wrapped</code> . In section <i>SSH checkout using command line CVS</i> , 3rd listitem: converted <code><quote></code> around “username” to <code><replaceable></code> , changed “SF username” to “SF user name”, and also wrapped “username” in the command example in a <code><replaceable></code> .

- In section *SSH checkout using other clients*, item `cvroot`: wrapped “username” in a `<replaceable>`.
- Gave the note in *Building the HTML and PDF docs* a title, and added a sentence to the second listitem.
- Corrected rev. 1.1.1 date in document history: 2004 -> 2005.
- Added large section on improving the PDF.
- 1.4 3 Aug 2006 ND Added warnings about firewalls and port 2401 plus how to cope when SourceForge changes your password.
- 1.4.1 23 Aug 2006 PV Added warning against checking out over a pre-existing local copy.
- 1.5 5 May 2007 PV *Apart from the manual module, is there any other Firebird documentation?* – Changed 2nd paragraph and list with links.
Do I really have to build the docs myself? – Changed 1st paragraph and removed all the links that followed.
Where to get the tools – Changed text throughout this section because we now download stuff to `manual/lib` and `manual/tools`.
How to set up the environment for the build – Change of wording in 2nd list item.
Building the HTML and PDF docs – In the note at the end: changed text and added link in the 2nd list item.
Building subsets with -Drootid – Changed the paragraph that starts with “How do you know the ID?”
Building a different base set with -Dbasename – Mentioned addition of papers set.
Advanced topic: Improving the PDF – Updated introductory paragraphs, including defects listing.
General repair scheme – Updated last list item in the Notes box.
Widowed headers – Heavily edited and id added; also moved part of content into new subsection *When there is no DocBook ID*.
Spaces in filenames, URLs etc. – Deleted.
Split table rows – Altered title, added id, altered Problem and Cause paragraphs.
Overly wide horizontal spaces – Added id.
Inserting zero-width spaces – Added id, slightly reworded the first two list items and removed the third.
Squeezed, truncated or otherwise messed-up page-sized content – Added.
XSL-FO references – Changed FOP version number in last paragraph.
License notice – (C) 2003–2006 -> 2003–2007.
- 1.6 24 Oct 2009 PV *Checking out the manual module* – Added warning against placing the local copy in a path with spaces or other “URL-unsafe” characters. Placed warnings in itemized list, gave warning element an explicit title, and changed the paragraph above.
Building the HTML and PDF docs – Added warning against path names that may change by URL-encoding, and provided a workaround. Gave ids to the five subsections.
Building subsets with -D[root]id, *Building a different base set with -Dbase[name]*, *Setting default values in build.xml* and *How the PDF is built* – Replaced `-Drootid` with `-Did` and `-Dbasename` with `-Dbase` throughout these sections.

Building subsets with -Did – Changed first paragraph.

Building a different base set with -Dbase – Changed paragraph starting with “Meanwhile...”.

If things go wrong – Inserted a paragraph on the problem with URL-un-safe characters in the path name.

When there is no DocBook ID – Gave this subsection an id.

Inserting zero-width spaces – Changed wording of second list item.

License notice – (C) 2003–2007 -> 2003–2009.

Appendix B: License notice

The contents of this Documentation are subject to the Public Documentation License Version 1.0 (the “License”); you may only use this Documentation if you comply with the terms of this License. Copies of the License are available at <http://www.firebirdsql.org/pdfmanual/pdl.pdf> (PDF) and <http://www.firebirdsql.org/manual/pdl.html> (HTML).

The Original Documentation is titled *Getting and building the Firebird manual module*.

The Initial Writer of the Original Documentation is: Paul Vinkenoog.

Copyright (C) 2003–2009. All Rights Reserved. Initial Writer contact: paulvink at users dot sourceforge dot net.

Contributor: Norman Dunbar – see [document history](#).

Portions created by Norman Dunbar are Copyright (C) 2006. All Rights Reserved. Contributor contact: norman-dunbar at users dot sourceforge dot net.