Getting and building the Firebird documentation

Paul Vinkenoog, Mark Rotteveel

Version 2.1.3, 6 June 2023

Table of Contents

Chapter 1. Introduction: Purpose of this Howto — Intended audience

This Howto explains, step by step, how you can download and build the firebird-documentation repository from the Firebird Project.

1.1. What is this "firebird-documentation repository" anyway?

The firebird-documentation repository is part of the Firebird Project on GitHub: https://github.com/FirebirdSQL/firebird-documentation. 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 firebird-documentation repository 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.

1.2. Apart from the firebird-documentation repository, is there any other Firebird documentation?

Yes! At the time of this writing, most useful Firebird documentation has been produced outside the Firebird documentation repository. 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:

- https://www.firebirdsql.org/en/documentation/ — the Firebird Documentation Index.
- https://www.firebirdsql.org/en/external-documentation/ — a listing of other sites with valuable Firebird documentation.

1.3. 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.

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

- You want to check on the absolute 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.

Chapter 2. Getting the firebird-documentation repository from GitHub

The firebird-documentation repository is a repository on the Firebird Project on GitHub. In order to download it, you need a piece of software called a *git client*. This section describes the necessary steps to get the software and the firebird-documentation repository. The actual doc building will be discussed in the next main section: Building the Firebird docs with Gradle.

2.1. What is git?

Git is a distributed version control system. It is a tool to manage software development, useful both for single developers and for teamwork. The Firebird Project on GitHub is divided into several so-called *repositories*, the firebird-documentation repository being one of them.

2.2. Git clients

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

- Linux, BSD and other Unices
  - Command-line git 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 (in ubuntu/debian you can install it from terminal with apt-get install git). If that doesn't work for you, get it at https://git-scm.com/
- Windows
  - Command-line git at https://git-scm.com/
  - GitHub Desktop: https://desktop.github.com/
  - TortoiseGit: A Windows Explorer plugin. Lacks some of the more advanced git functions. Get it at https://tortoisegit.org/
- Mac OS X
  - Mac OS X comes with command-line git already included.
- Others
  - Try your luck at https://git-scm.com/, Google for it, or ask on the firebird-devel list.

Get one or more of the aforementioned clients and install according to the instructions that come with it. After that, you are ready to check out the firebird-documentation repository.

2.3. Checking out the firebird-documentation repository

If you only want to build the documentation, you can *clone* the repository to your local machine and build from there. If you want to contribute, there are two ways of contributing to a repository: contributing through a *pull request* from a *fork* of the repository, or working directly on the repository.

The common method of contributing on GitHub is through so-called pull request. With a pull request, you don't commit directly to the original repository. Instead, you fork the repository to your own GitHub account, make your changes on this fork, and then create a pull request to ask for those changes to be incorporated in the repository of the Firebird Project.

Working directly on the repository is similar to using a fork. The main difference is that working directly on the repository is more suitable for trusted regular contributors, while pull requests are very suitable for occasional or one-off contributions. To be able to work directly on the repository requires that you have been given commit-access to the repository, while anyone can create a pull request.

If you are unfamiliar with git and GitHub, you may also want to read Getting started with GitHub.

2.3.1. Cloning the firebird-documentation repository

To clone the Firebird repository on the command line:

- Go to the directory where you want to clone, the repository will be cloned to a subdirectory in the location, for example `C:\projects\`
- `git clone https://github.com/FirebirdSQL/firebird-documentation.git`
- The repository will be downloaded into `C:\projects\firebird-documentation\`

If you decide to contribute through pull requests, replace the repository URL with the one from your fork.

Cloning through a git (graphical) client usually only requires the repository URL. Check the documentation of your client for more information.

Chapter 3. Building the Firebird docs with Gradle

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 installed on your system.

In the next subsections we will show you:

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

If you already have recent version of Java installed, you may skip the first step.

3.1. Where to get Java

Download and install:

- Java Development Kit, Standard Edition — or JDK SE — version 8 or later.

  This is a much larger package, and it also contains the JRE SE. If you want the JDK, go to https://www.oracle.com/technetwork/java/javase/downloads/index.html or to https://adoptopenjdk.net/ and get the latest stable version. When you have to choose between JRE and JDK, take the JDK. Download the installation program and run it.

  > The docbook tasks will work on Java 8, but are not fully tested on higher Java versions. If you run into problems or errors with newer Java versions, fallback to Java 8

3.2. How to set up the environment for the build

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

- On Windows, this is typically something of the form `C:\Program Files\Java\jdk1.8.0_232`. 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/jdk`, 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` environment variable 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\jdk1.8.0_232` or under Linux/bash with `export JAVA_HOME=/usr/lib/java/jdk`. (Note: these paths are just examples; they may or may not be the same as yours.)

Tip: make the `JAVA_HOME` environment variable 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.

3.3. 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.md` file that lives in the `firebird-documentation` 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.md` instructs you otherwise, go to the folder `firebird-documentation/` and give the command

   `gradlew` (in Windows), or

   `./gradlew` (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. to build the asciidoc documentation:

   `gradlew asciidocHtml` or

   `gradlew asciidocPdf`

   Whatever you build will wind up in the directory tree under `firebird-documentation/build`
5. To build the DocBook documentation, you can use:

   `gradlew docbookHtml` or

   `gradlew docbookPdf`

   We are gradually migrating the DocBook documentation sources to asciidoc. However, some older documentation — like release notes for older versions — will not be migrated.

By default, all documentation in the `firebirddocs` base set will be built. To specify a specific document or subset, see Building subsets with `--docId`. To specify a different base set, see Building a different base set with `--baseName`.

Notes

- If you build the DocBook 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 DocBook 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.

⚠ If you placed your local copy of the firebird-documentation repository in a path that contains spaces or other non-alphanum, 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 best way to avoid these problems is to place the firebird-documentation repository 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.)

3.3.1. Building non-English sets with `--language`

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

```
gradlew asciidocPdf --language=es
```

```
gradlew docbookHtml --language=fr
```

For the asciidoc tasks, output will go into subdirectories like `build/docs/asciidoc/pdf/es/firebirddocs`, `build/docs/asciidoc/pdf/fr/firebirddocs`, etc

For the docbook tasks, non-English output will go into subdirectories like `build/docs/pdf-firebirddocs-ru`, `build/docs/html-firebirddocs-fr`, etc.

If you don’t specify `--language`, the English set will be built.

⚠ 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.

3.3.2. Building subsets with `--docId`

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 article — use the `--docId` argument.

For the asciidoc tasks, supply the directory name containing the document as argument of the `--docId`, for example for this document use:

```
gradlew asciidocPdf --docId=docbuildhowto
```

For the docbook tasks, supply the ID of the element you want to build, for example:

```
gradlew docbookPdf --docId=fbutils
```

```
gradlew docbookPdf --language=fr --docId=qsg15-fr
```

For asciidoc, you can find the ID by looking at the directory structure. Sources for individual books or articles are in a specific directory. The name of that directory is the ID of the document. For example, the sources of this document are in `src/docs/asciidoc/en/firebirddocs/docbuildhowto/`, so the ID is `docbuildhowto`

For DocBook, you can find the ID 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.

3.3.3. Building a different base set with `--baseName`

Since January 2006, the Firebird Release Notes have been integrated with the firebird-documentation repository, 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, `--baseName` (pun intended), whose value should be “`rlsnotes`” to build the Release Notes:

```
gradlew asciidocPdf --baseName=rlsnotes
```

```
gradlew asciidocPdf --baseName=rlsnotes --docId=rlsnotes20
```

```
gradlew asciidocPdf --baseName=rlsnotes --language=fr
```

This example uses `asciidocPdf` as task name, but existing release notes are (still) in DocBook. The `docbookPdf` task supports the same options.

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 (DocBook only) is placed in `build/docs/html-<baseName>`, 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 `build/docs/html-<baseName>-<language>`. For instance, the English HTML Release Notes are written to `build/docs/html-rlsnotes`, the French notes to `build/docs/html-rlsnotes-fr`.

3.3.4. 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.

Chapter 4. Keeping your firebird-documentation repository up to date

The firebird-documentation repository 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 GitHub. Of course, it would be a waste of bandwidth if you had to check out the entire repository 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 git has an `pull` command. With `pull`, 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 easy. If you use command line git, go to the `firebird-documentation` directory and type:

```
git pull
```

This command is the same whether you checked out anonymously or with your GitHub account. Git knows which server to contact and how to authenticate you because this information is saved in the `firebird-documentation/.git` subdirectory, which was created automatically when you first checked out the repository.

If you use another git tool, look for its `pull` command or menu option.

Be aware that git offers multiple ways to update, including fetching and merging changes.

4.1. Keeping a fork up-to-date

If you are working on a fork of the repository, updating requires a bit more preparation. Your local clone of a git repository can be associated with multiple remote repositories. By default, the repository you cloned from is called the *origin*. You can associate multiple remote repositories (or *remote*) with your clone.

To add the firebird-documentation repository as a remote with the name `upstream` to your repository, you can use:

```
git remote add upstream https://github.com/FirebirdSQL/firebird-documentation.git
```

The name `upstream` is a common name for pointing to the original repository you forked.

Then to update your fork with the changes from the remote repository, you can use:

```
git fetch upstream
```

```
git merge upstream/master
```

See Fork a repo on GitHub for more information.

Chapter 5. If things go wrong

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

If a DocBook 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 in 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: Advanced topic: 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-devel list. Please give a good description of your problem, so we can help you better. If you aren’t on the firebird-devel list yet, visit https://groups.google.com/g/firebird-devel for information and subscription.

Chapter 6. Advanced topic: Improving the PDF

Some topics covered in this section apply to the ant build and old versions of Apache FOP, they may no longer apply when using the Gradle-build, or the files may have a different path in the Gradle-build. When in doubt, ask for help on the firebird-devel list.

This section only applies to DocBook.

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.

6.1. 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 `firebird-documentation/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 `firebird-documentation/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 `firebird-documentation/inter/qsq15.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.

6.2. 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, fewer 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.

6.3. 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

6.3.1. Widowed headers

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

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

> 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`.

6.3.2. 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 tables) are implemented using `fo:tables` and thus contain `fo:table-rows`.

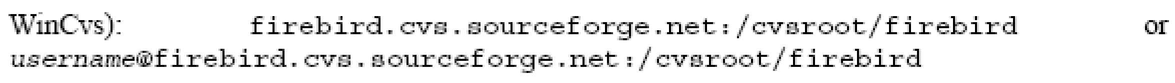
Once the start of the split row is found, add a `break-before` attribute like you did with widowed headers:

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

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

6.3.3. 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.

6.3.4. 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 *in*formal 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.

6.4. XSL-FO references

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

The Apache FOP homepage is here: https://xmlgraphics.apache.org/fop/

The Apache FOP compliance page is here: https://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: Building Chunked HTML For Asciidoc

The primary output types we use are PDF and single-page HTML. However, for search engine optimization, it might make sense to have chunked output for some documents. Unfortunately, AsciiDoctor does not support chunked HTML output.

The workaround is to generate Docbook from the asciidoc sources, and then use DocBook stylesheets to produce chunked HTML. This runs into some problems, as AsciiDoctor output is DocBook 5.0, while our Gradle build uses stylesheets supporting DocBook 4.5. Supporting both seems to be impossible as the required XSLT engine versions are incompatible.

This appendix documents some — largely manual — steps to produce chunked output. Once this is more refined, we might move this into the Gradle build itself.

A.1. Produce DocBook Output

To produce DocBook from the asciidoc sources, use the `asciidocDocbook` task:

```
./gradlew asciidocDocbook --baseName=refdocs --docId=fblangref40
```

The options are same or similar to those of the `asciidocHtml`.

The task will output into `build/docs/asciidoc/docbook/<language>/<basename>/<docid>`, for the example: `build/docs/asciidoc/docbook/en/refdocs/fblangref40`.

A.2. Produce Chunked HTML

To produce the chunked HTML, we will use the experimental *DocBook xslTNG* Docker image.

This Docker image is not available from a registry, so it needs to be downloaded and build yourself:

1. Download the `docbook-xslTNG-1.5.0.zip` from DocBook xslTNG Stylesheets version 1.5.0
2. Unzip it
3. From that directory, run

   ```
   docker build -t docbook-xslt .
   ```

   See also Run with Docker in *DocBook xslTNG Reference*

> The instructions below assume a Windows command prompt. For other OSes or shells, you may need to modify things.

First run the `asciidocDocbook` task and build the *docbook-xslt* image. Go to the root directory of the *firebird-documentation* repository, and follow these steps.

1. Create output directory

   ```
   mkdir build\docs\asciidoc\chunked
   ```

2. Build chunked output using Docker

   ```
   docker run -v %CD%\build\docs\asciidoc\chunked:/output -v
   %CD%\build\docs\asciidoc\docbook:/input ^
          docbook-xslt --resources:/output /input/en/refdocs/fblangref40/firebird-40-
   language-reference.xml ^
          chunk=firebird-40-language-reference.html chunk-output-base-
   uri=/output/en/refdocs/fblangref40/ ^
          resource-base-uri=../../../ component-numbers-inherit=true lists-of-
   tables=false persistent-toc=true ^
          verbatim-style-default=plain verbatim-line-style=
   ```

Appendix B: Document History

The exact file history is recorded in the firebird-documentation git repository; see https://github.com/FirebirdSQL/firebird-documentation

Revision History

0.1	2 Nov 2003	PV	First draft published under the title *How to get and build the Firebird manual module*.
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	*The following changes have accumulated between March 2004 and Feb. 2005:* 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 *Getting and building the Firebird manual module*. 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 `build.xml`. Changed docwritehowto `link` to `ulink`, as the articles will be in separate PDFs from now on.
1.2.1	15 May 2006	PV	Replaced `cvs.sf.net` (3x) and `cvs.sourceforge.net` (6x) with `firebird.cvs.sourceforge.net` to reflect new situation at SF. Also added “on one line” above two examples that are now line-wrapped in the PDF.

Revision History

1.3	17 Jul 2006	PV	Changed all `sectN` elements to `section`. Shortened ID of Introduction and assigned IDs to its child sections. Spelling matters: RDMS → RDBMS, parallell → parallel, OS’es → OSes, CD’s → CDs, wil → will, envvar → envar, linewrapped → line-wrapped. In section *SSH checkout using command line CVS*, 3rd listitem: converted `<quote>` around “username” to `<replaceable>`, changed “SF username” to “SF user name”, and also wrapped “username” in the command example in a `<replaceable>`. In section *SSH checkout using other clients*, item cvsroot: 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.

Revision History

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* — © 2003-2006 → 2003-2007.

Revision History

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-unsafe 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* — © 2003-2007 → 2003-2009.
1.6.1	26 Oct 2009	PV	*Building the HTML and PDF docs* — Fixed the IDs of this section and five of its subsections, which contained the word “`buidling`” instead of “`building`”: • `docbuildhowto-buidling-output-docs` • `docbuildhowto-buidling-non-english` • `docbuildhowto-buidling-subsets` • `docbuildhowto-buidling-other-basesets` • `docbuildhowto-buidling-defaults` • `docbuildhowto-buidling-conclusion` *If things go wrong* — Fixed link to *Building the HTML and PDF docs*: `buidling` → `building`. In the same sentence, the words “at the end of” have been changed to “in”, because the link target is not at the end of the section (it’s just before the first subsection). *When there is no DocBook ID* — Removed ID attribute from title element, because it was a) unnecessary, and b) a duplicate of the section ID and therefore illegal.

Revision History

1.6.2	10 Feb 2013	PMA	Fixed some old links for Java JRE, replaced Java 2 reference with just Java, cvs home page is now fixed
1.7.0	25 Jan 2020	MR	Replaced CVS instructions with git instructions
2.0.0	29 February 2020	MR	Added instructions for the new Gradle-based build
2.0.1	19 May 2020	MR	Converted document to asciidoc
2.1.0	21 May 2020	MR	Added instructions for asciidoc tasks and some copy editing. Fixed conversion from `<sgmltag class="starttag">` to include the angle brackets.
2.1.1	23 May 2020	MR	Fixed rendering issue with bare link immediately followed by a -- generated em-dash
2.1.2	23 May 2020	MR	Updated link to Firebird Docwriting Guide
2.1.3	06 Jun 2023	MR	Replaced firebird-docs references with firebird-devel

Appendix C: License notice