XML proofreading

ver. 1.3

Thomas Hamann

This project was subsidized in part by the EU project “pro-iBiosphere” (Grant agreement 312848).

Introduction

The final step before a taxonomic work can be published on the internet is to proofread the XML file and to ensure the XML validates. The latter means the XML file must be valid according to the XML schema, in this case FlorML. Proofreading is probably the most time-consuming task in the whole mark-up process, but is essential to deliver qualitatively good results. This manual aims at providing examples of the more common problems you may encounter and their solutions. However, it is not all-inclusive. Proofreading XML is more of a process that you learn to do; where you learn to recognise what kind of issues you may encounter and where they are most common.

Table of Contents

XML Spy Professional installation and set-up

Before you can proofread XML files and visualize the FlorML schema for reference, you should have XML Spy Professional (version 2010 or later) installed. Ask your IT department to do this for you if you lack the rights to do so. An x64 version is available, should you have a 64-bit processor in your computer.

Once XML Spy has been installed and started for the first time, you will be shown a window similar to Figure 1.

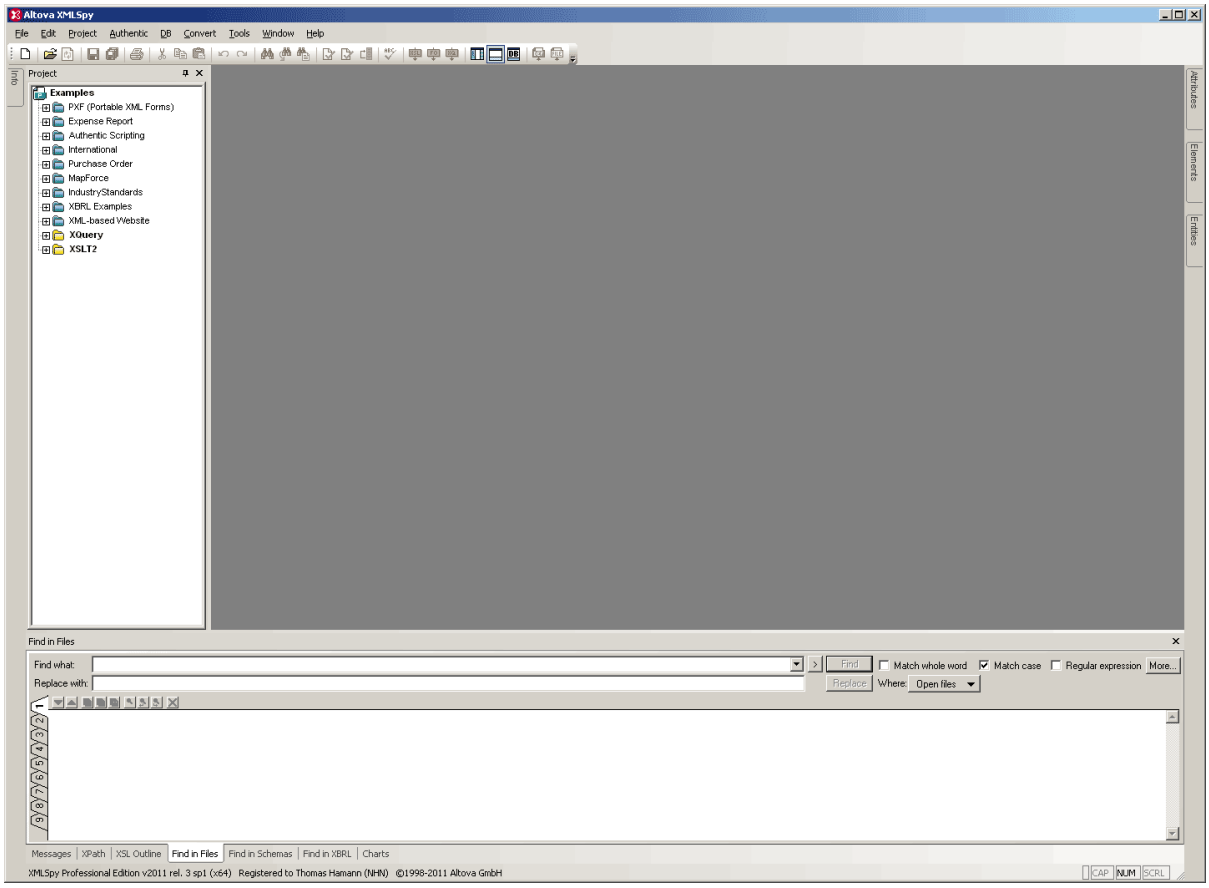


Figure 1: XML Spy Professional main window.

XML Spy has an interface that is entirely customisable. In Figure 1 you can see some tabs at the left ("Info") and right side ("Attributes", "Elements", "Entities") of the screen. When you hover above them with the mouse pointer, each tab will automatically open. If these tabs are shown next to each other and cannot be collapsed, click the "Auto Hide" button on each tab (Figure 2) to get the previously described behaviour. The objective of this is to maximize your screen estate to facilitate working with large XML files and schemas.

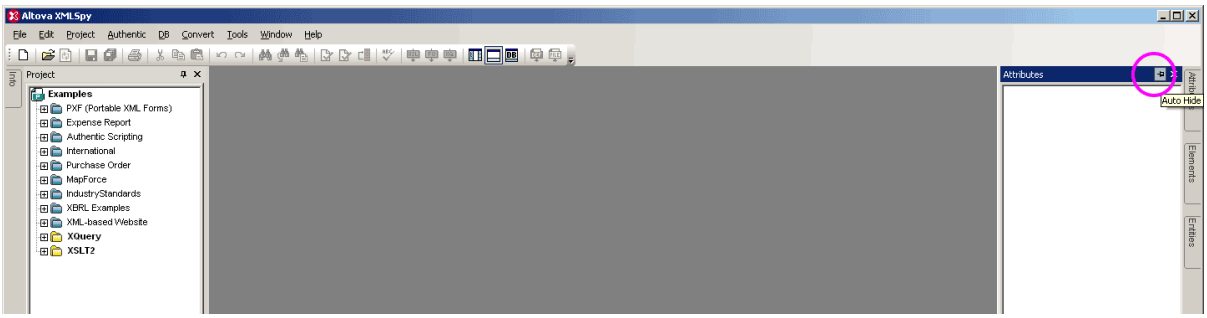


Figure 2: "Auto Hide"-button to toggle automatically hiding a tab.

Once you've done this, go to the "Tools"-menu, and click on "Options…".

1) In the window that opens, click on the "File"-tab and check whether the options shown in Figure 3 are set correctly. The window may look slightly differently in different versions of XML Spy.

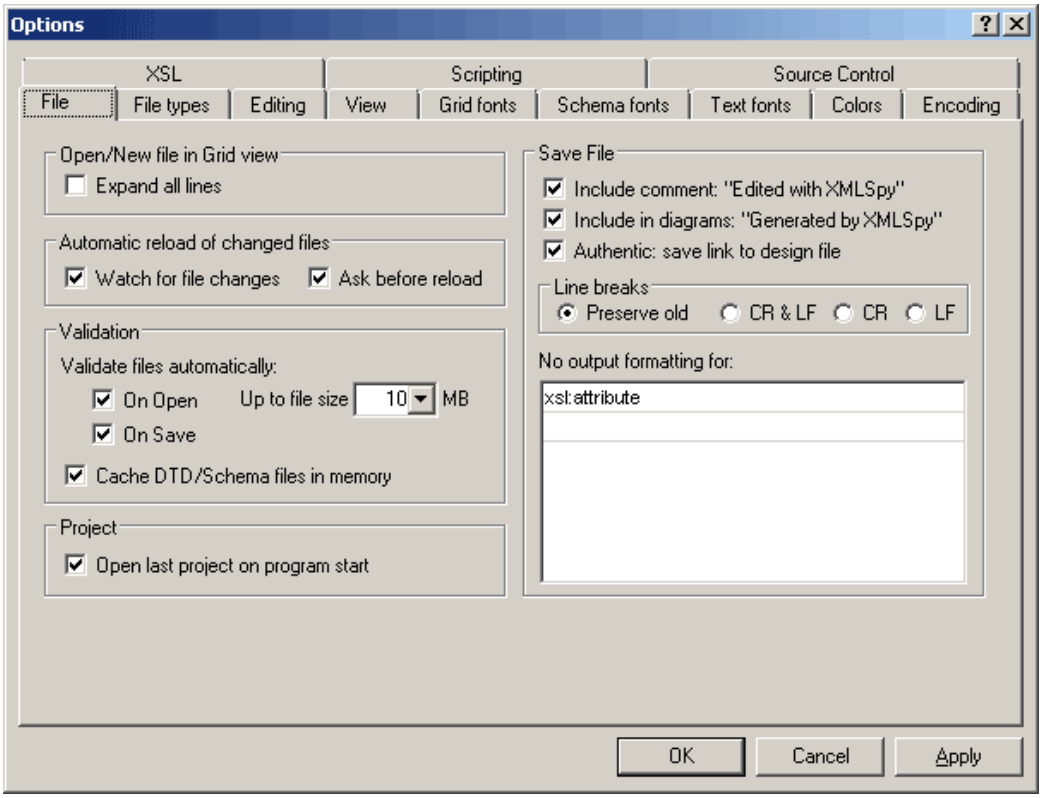


Figure 3: File options in XML Spy.

2) Click on the "Editing"-tab and check whether the options shown in Figure 4 are set correctly.
3) Click on the "View"-tab and check whether the options shown in Figure 5 are set correctly.

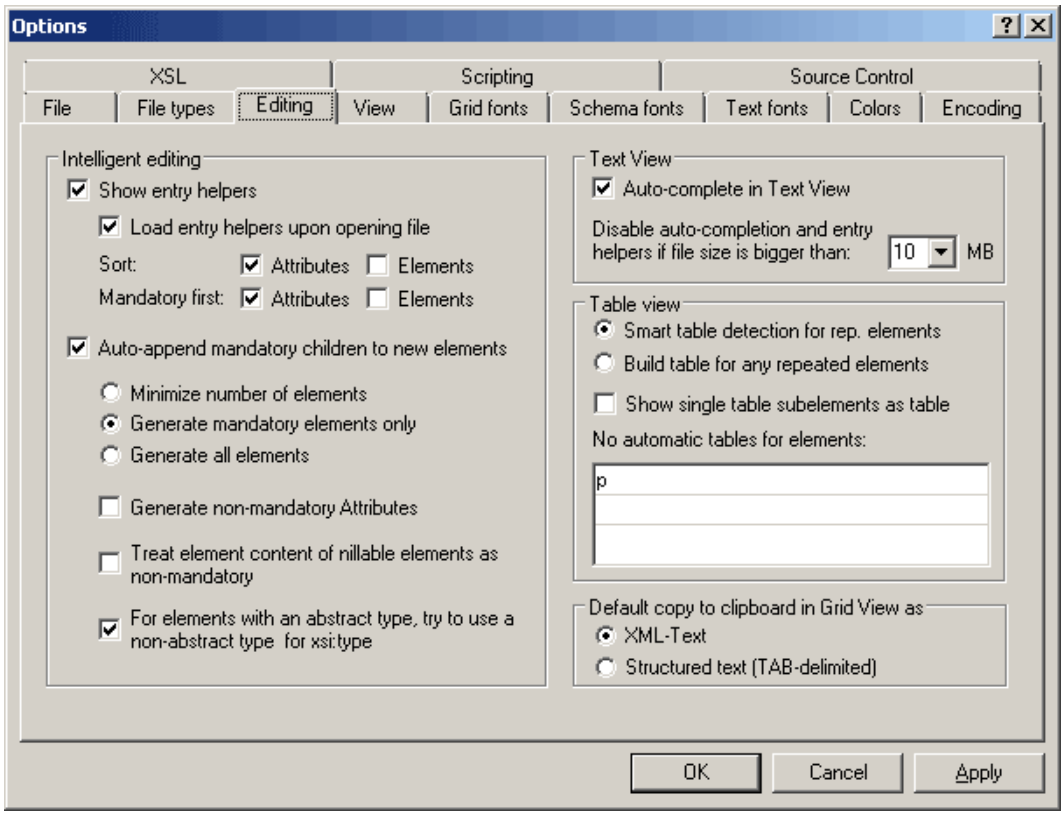


Figure 4: Editing options in XML Spy.

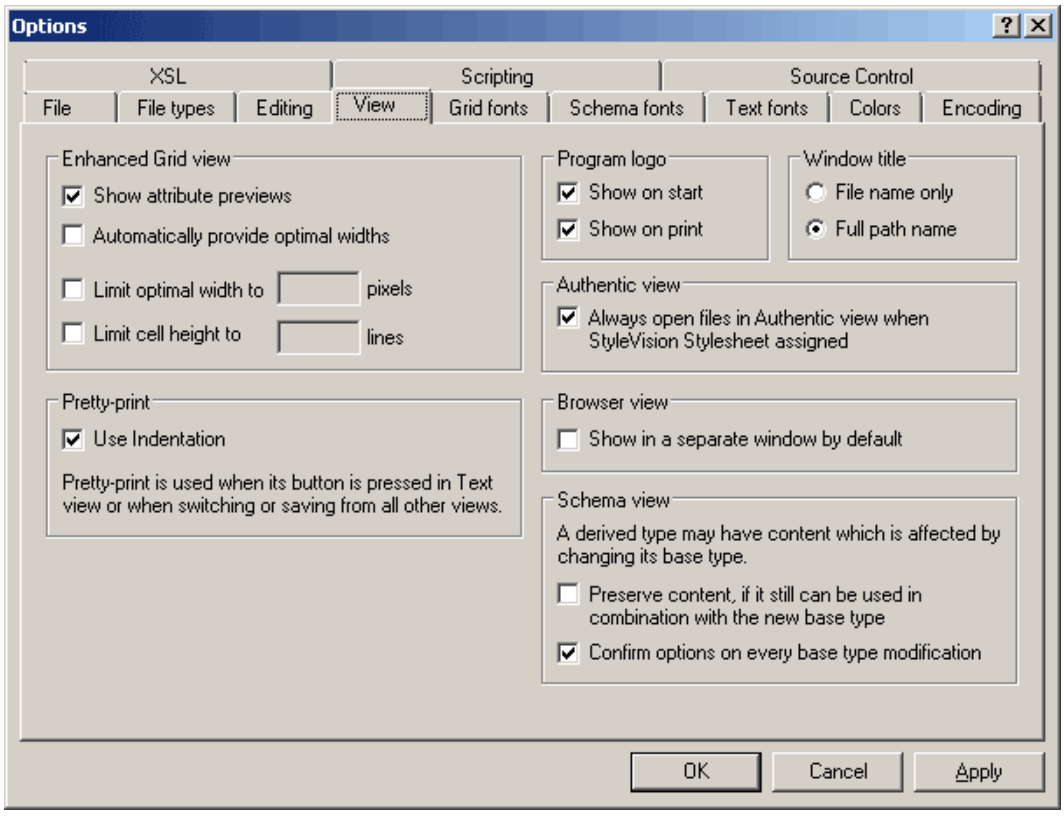


Figure 5: View options in XML Spy.

4) Click on the "Encoding"-tab and check whether the options shown in Figure 6 are set correctly.

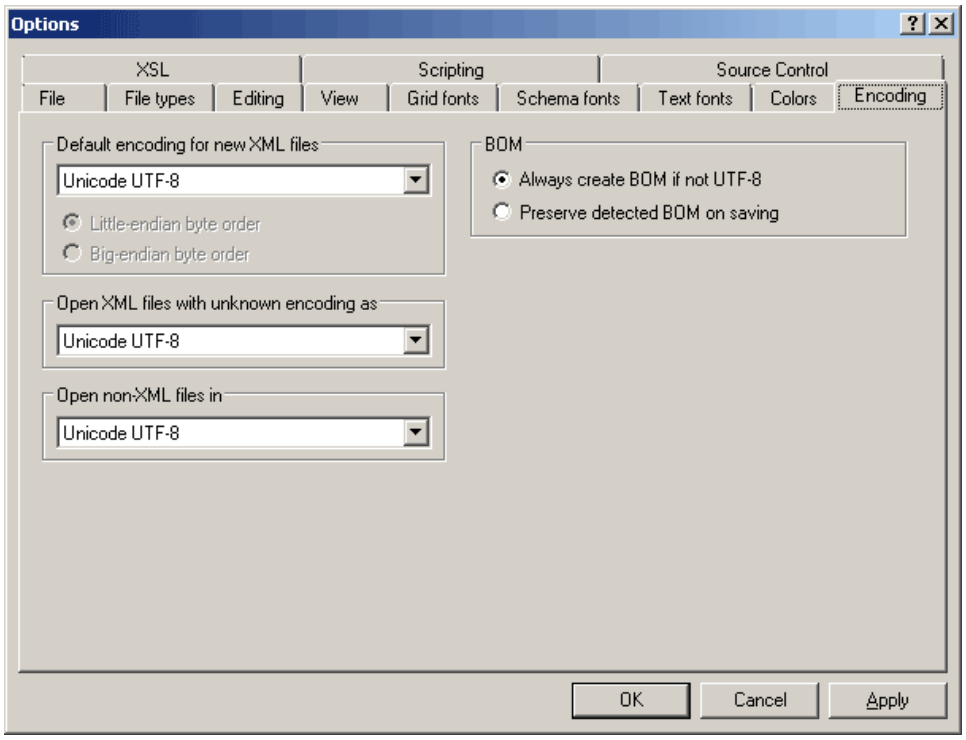


Figure 6: Encoding options in XML Spy.

XML Spy should now be set up properly for you to proofread XML files.

Proofreading XML files with XML Spy

Preparation

Before you start proofreading, you should do the following:

1) First, make a copy of the XML file that was the final result of running the scripts. Place this copy in a separate folder. This prevents the original file from being lost should anything go wrong. Then change the file name of the copy to something that indicates to you that this will be the file that has been proofread, e.g. by appending "final" (without quotes) to the file name (before the file extension). E.g. a file called "fdgvol9_11.xml" becomes "fdgvol9_final.xml".
    a. You can rename a file from "Windows Explorer" by selecting the file and clicking it another time (not double-clicking; this will open the file) to select the text of the file name. You should see something similar to Figure 7. You can click anywhere in the file name and start typing at this point. Note that clicking elsewhere will deselect the file. After changing the file name, you may be asked whether you want to actually change it. Acknowledge this.
    b. You can also use Notepad++ to rename files. Open the file you want to rename in Notepad++, then click on the “File”-menu and choose the

option “Rename…”. This opens the “Save as…”-window, where you can enter the new name for the file (Figure 8). Then click “Save”.

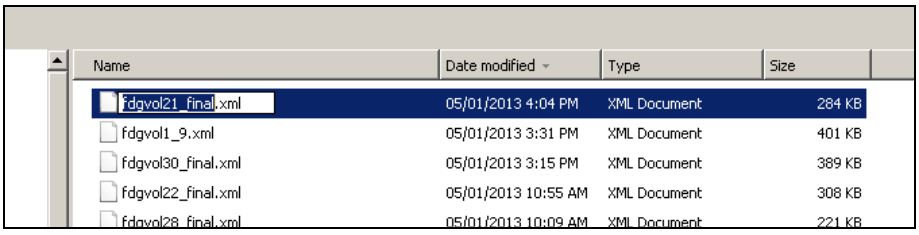


Figure 7: Renaming a file in Windows Explorer.

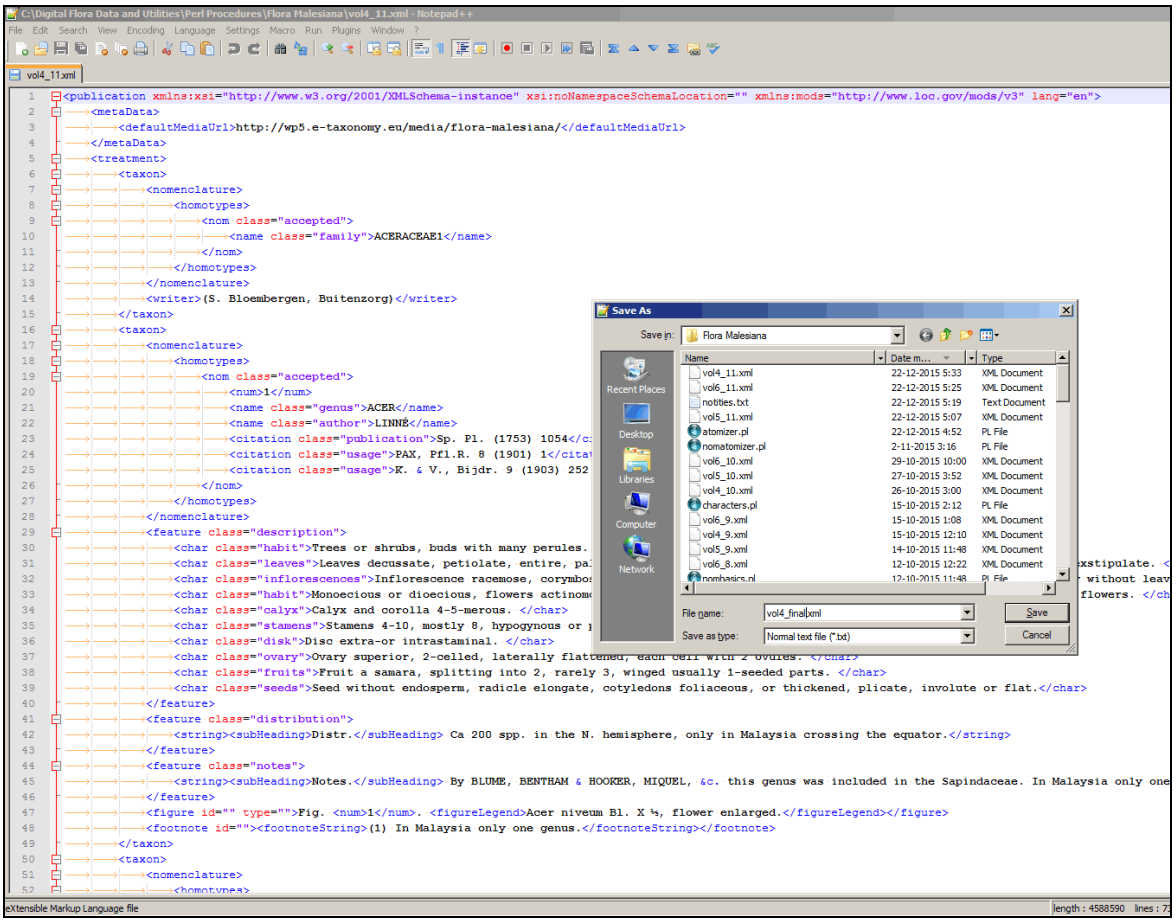


Figure 8: Renaming a file in Notepad++.

2) Now go to XML Spy. Click on the "File"-menu, and select "Open…". Find the file you just renamed, and open it. You will get something similar to what is shown in Figure 9.
    a. If you ran your scripts over multiple volumes pasted end-to-end, you should separate the volumes at this point. Just copy the text of each volume, including all accompanying XML, into blank XML files. Save each volume as a separate file. Then proofread them one-by-one as explained in the next section. Failing to do this with files with volumes pasted end-to-end will cause a validation error.

3) In Figure 9 three different areas can be seen. At the left, there is the "Project"-tab, which can be ignored. Below, there is the "Messages"-pane, which displays error messages, unless the XML file validates. The last area is the editing area, which shows the contents of the XML file. How proofreading actually works in explained in the next section.

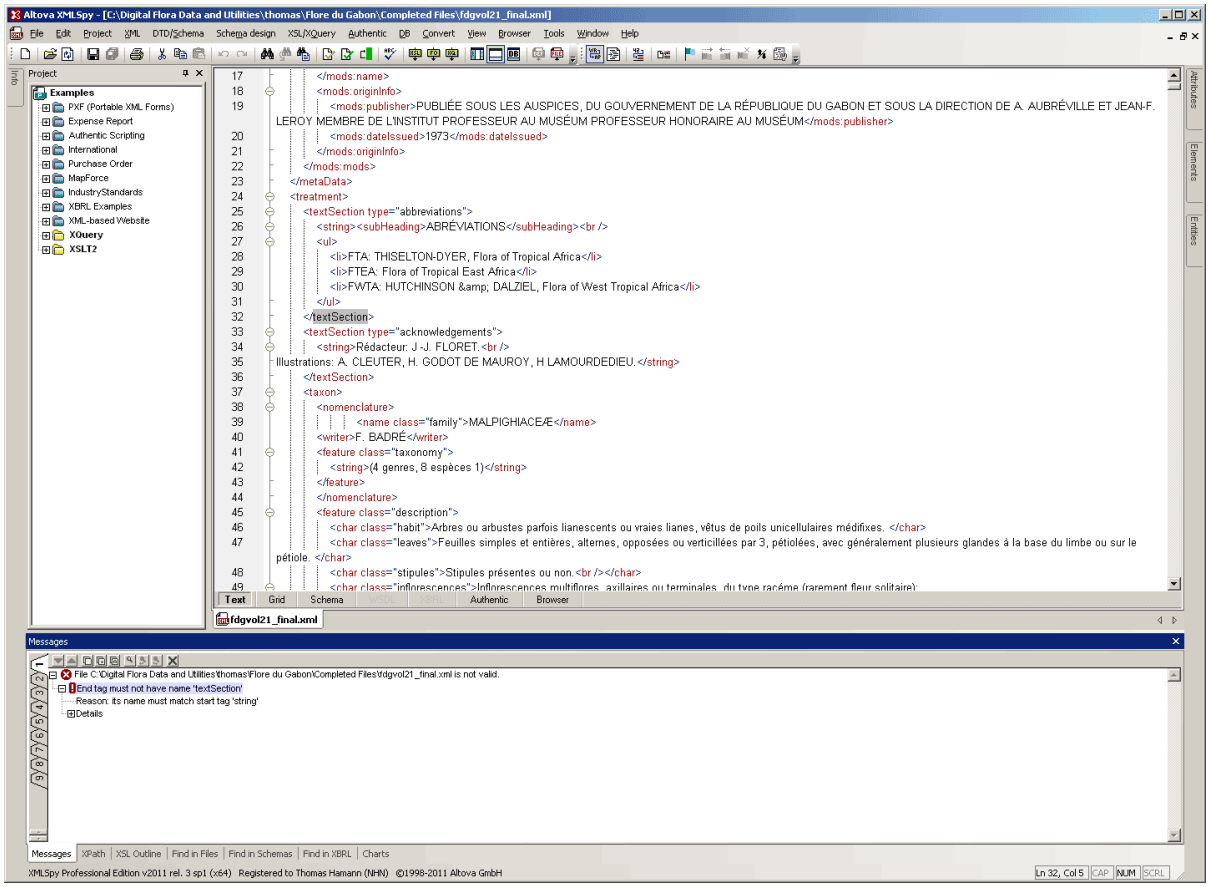


Figure 9: After opening an XML file for proofreading.

4) The last step in the preparation is to open the current version of the FlorML XML schema and save it as a new version, by going to the "File"-menu, selecting "Save as…", and incrementing the version number in the file name by 1 (Figure 10). You can use a new version number for the XML schema for every single volume you proofread, but it is more practical to use a new version number for each new batch of volumes that will be added to a dataportal. The new version number is needed because you will likely need to make small additions or sometimes changes to the XML schema, such as adding more taxonomic character options.

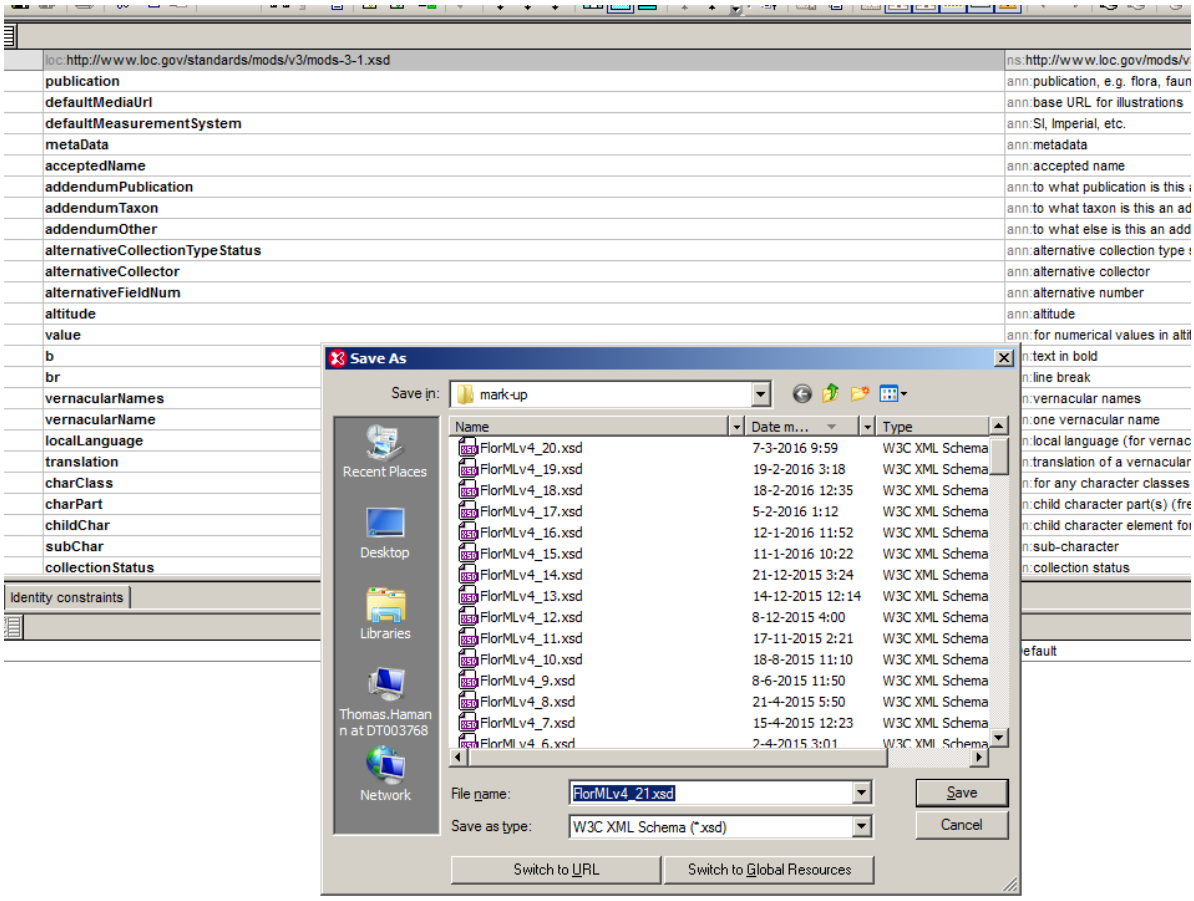


Figure 10: Saving a new version of the XML schema.

5) Finally, click on the button in front of "publication" in the "Schema"-view of FlorML (Figure 11) to display the schema in a graphical manner that is easier to navigate (Figure 12).

   The various branches of the schema can be expanded or collapsed by clicking on the small boxes with "+" or "-" in them, respectively. Each of the boxes represents an element. Some elements have boxes reading "attributes" associated with them. You can expand these to look at that element's attributes. The "Details" and "Facets" tabs on the right of the screen provide additonal information for each element and attribute.

   It is suggested that you play around with the XML schema and XML Spy at this point to see how things work (but do not change anything in the schema). Try to get a feel of how the XML schema is build up and do not only click on the little plusses and minuses, but also read what is written in the schema. Many elements have comments below them explaining what they are for.

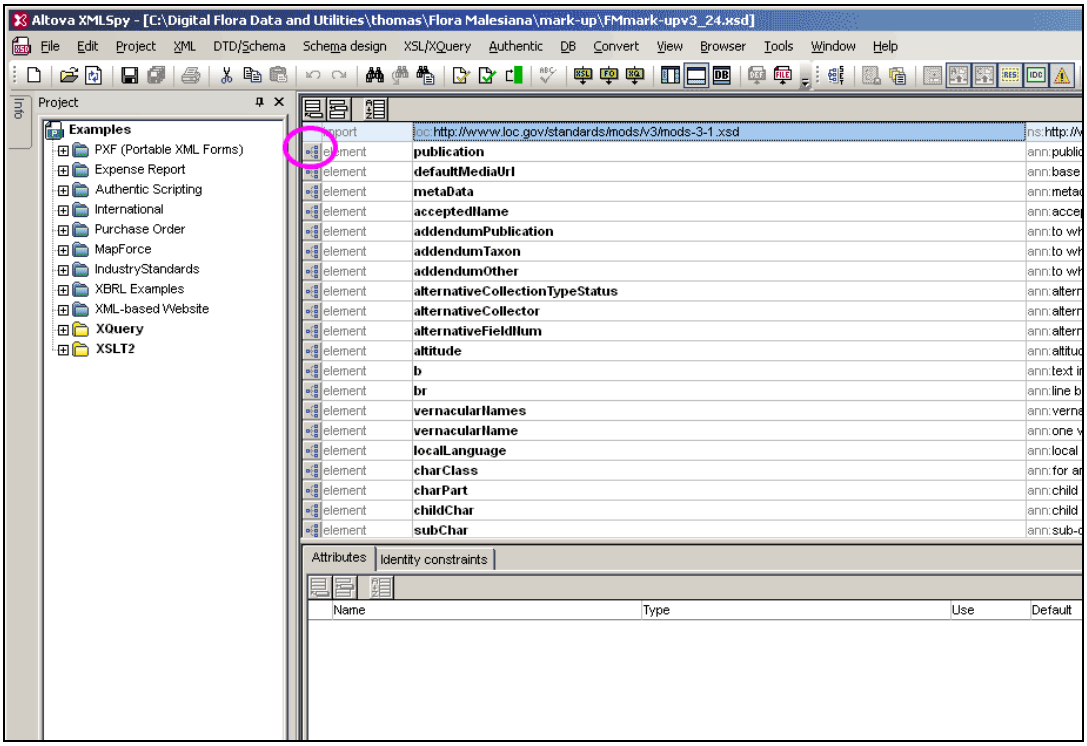


Figure 11: Switching the display of the XML schema to one that is easier to navigate.

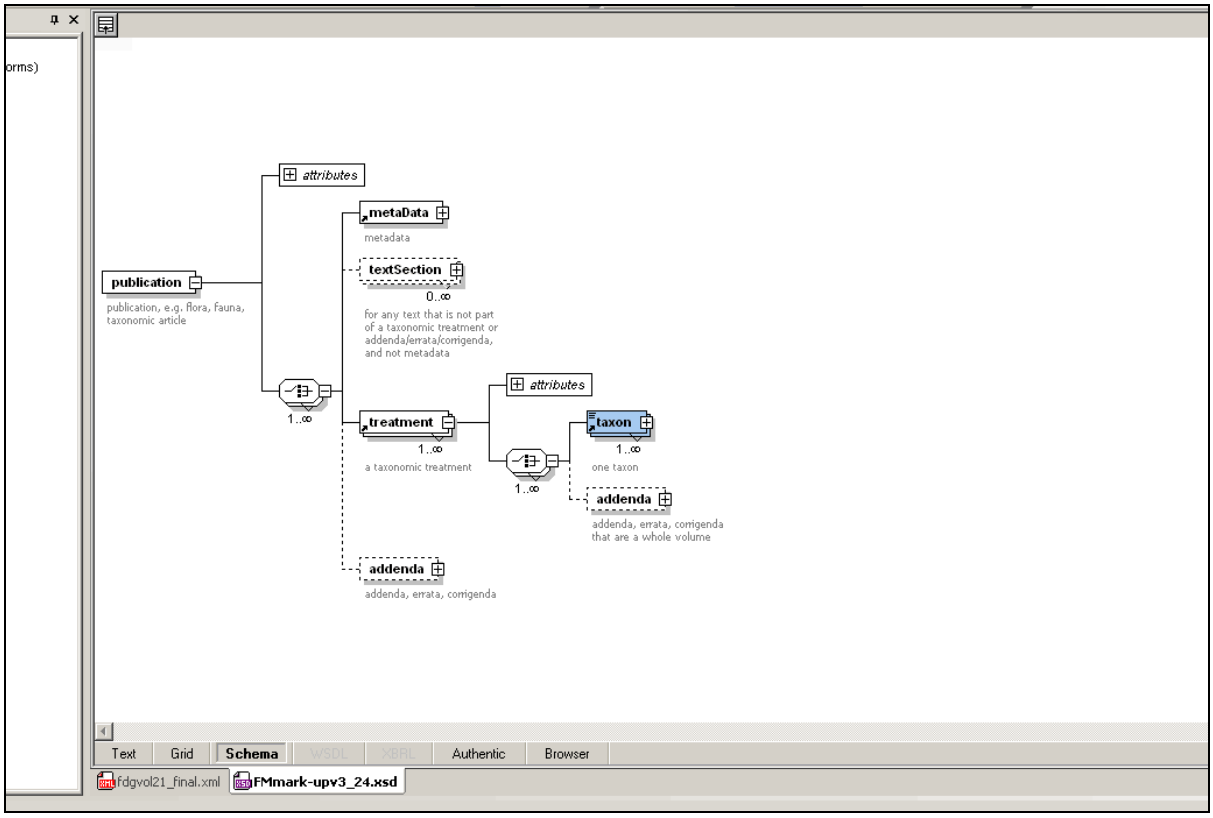


Figure 12: Graphically displaying an XML schema.

Now you are all set to start proofreading!

Advanced users: Using Notepad++ as a companion for proofreading

It is possible to use Notepad++ as a companion for proofreading. Notepad++ has more options to do certain things than XML Spy. Some examples are:

- finding specific problems using regular expressions with the “Find” function.
- replacing text (including line breaks)[1]
- changing the case of text from UPPERCASE to lowercase and the other way around, with a shortcut key combination (“Ctrl-U”).

Notepad++ also has more colourful syntax highlighting, which can be advantageous when looking for things that are wrong.

Just open the same file that you opened in XML Spy in Notepad++. Be aware that before you use Notepad++ to make changes to the file, you need to save it in XML Spy, and vice-versa, otherwise modifications will be lost.

Note: There appears to be a bug in some versions of Notepad++ that causes it to crash when editing opening or closing tags in certain files. This is likely related to some invisible character in these files (Save often!). If this happens, edit the offending piece of code in XML Spy instead.

[1] On the other hand, Notepad++ does not appear to be capable to search for text that is located on two subsequent lines. Use XML Spy’s regular expression search for that.

Proofreading

Conventions

- Each XML element consists of an opening and closing tag, e.g. <string> and </string>.
    - Exceptions: the element for line-breaks combines both tags into one tag:
. There are a few other exceptions, which are mentioned in **FlorML reference.doc**.
- XML code is shown in this font.

Useful keyboard shortcuts

Here are some useful key combinations for when proofreading:

- "Ctrl-S" - To save the document.
- "Ctrl-Z" - To undo an action.
- "Ctrl-Y" - To redo an action.
- "Ctrl-X" - To cut some text out of a document.
- "Ctrl-C" - To copy some text.
- "Ctrl-V" - To paste some text.
- "Ctrl-F" - To open the "Find"-window.
- "Ctrl-H" - To open the "Find & Replace"-window

Try to memorise these. Cutting, copying and pasting also works in the "Find" and "Find & Replace"-windows.

Validation

Before XML Spy can start to validate an XML file against an XML schema, it should know where the XML schema(s) used can be found.

In the case of FlorML, this information is located within the opening publication tag, the first XML tag of the document. Figure 13 shows what the publication tag looks like in an XML file where it has not been indicated where the XML schemas can be found.

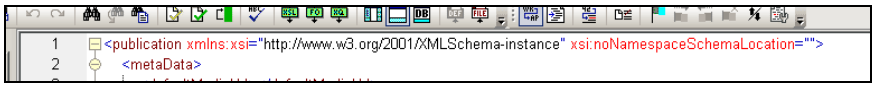


Figure 13: Opening publication tag with little to no indications where to find schemas.

The text in question is reproduced below in an easily copyable format:

<publication xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="" xmlns:mods="http://www.loc.gov/mods/v3" lang="">

xsi:noNamespaceSchemaLocation is used because FlorML currently has no online namespace. The schema location needs to be added in between the quotation

marks following the equal sign. This is a file location path, e.g. file:///C:/XML/schema/FlorMLv4_21.xsd. The version number of the schema will need to be changed accordingly any time a new version of the schema is created.

Another thing visible here is lang="", which indicates the main language of the document that is going to be proofread. The language code for English is "en", for French “fr”, and for Latin is "la". A list of language codes can be found here: http://www.w3schools.com/tags/ref_language_codes.asp.

Once the publication tag has the correct information you can save the XML file. Now XML Spy will try to validate the XML file, and it will likely not validate (Figure 14). Just click on "Yes" to save the file anyway.

Figure 14: XML file validation failure.

How does XML Spy validate a file?

When you save a file, XML Spy validates an XML file against an XML schema in two passes.

In the first pass, it checks the well-formedness of the file. This means it checks whether all opening tags that are present are matched by accompanying closing tags, as well as the inverse. At this point it does not matter whether the tags are in the correct place or not, or whether all required elements are in place or not.

In the second pass, which XML Spy only tries when the first pass has been sucessfully completed, actual validation against the XML schema is performed. Now it is of importance that elements are where the XML schema allows them to be and that all elements that are expected in a location according to the XML schema are indeed present. During this process XML Spy also checks whether required attributes are filled out. Once this pass has been completed succesfully, the XML file will validate.

There are two ways to perform these checks:

1) The first option is to save the XML file. XML Spy will then check well-formedness and try to validate the file against the XML schema, giving the error message shown in Figure 14 if the file does not validate and jumping to the location in the file where the problem is located (again, just click "Yes" to save the file anyway). The advantage of this method is that both steps described above are performed together and the file is saved.
2) The second option is to use the "Check well-formedness" and "Validate"-functions of XML Spy. These can be accessed via the "XML"-menu (Figure 15), or by clicking on the buttons on the toolbar (Figure 16) or more simply hitting the "F7" ("Check well-formedness") or "F8" ("Validate XML") key on the keyboard. When an error is located, XML Spy will jump to that location in the XML file. These two steps have to be performed separately.

Choose whichever option you prefer.

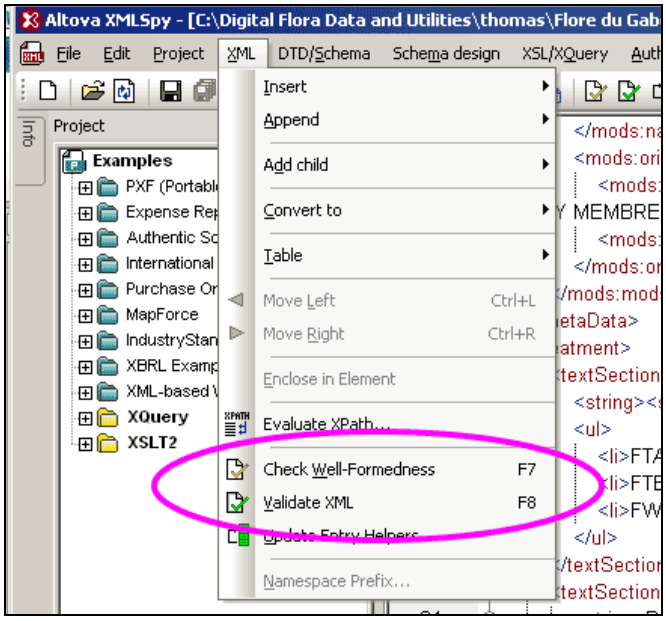


Figure 15: "Check Well-Formedness" and "Validate XML"-options in the "XML"-menu.

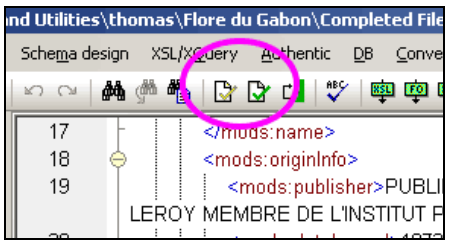


Figure 16: "Check Well-Formedness" and "Validate XML"-options on the XML Spy toolbar.

A combined validation/proofreading approach

In the proofreading process you will make some use of how XML Spy behaves during the validation process to combine it with the actual proofreading. Proofreading consists of:

- checking whether text was properly identified
- correcting mark-up errors whenever it was not
- correcting OCR errors
- manually inserting bolded and/or italics text whenever the original text uses it for explicitly indicated emphasis
- manually inserting additional contextual information into elements whenever required
- manually inserting such information like image file names etc.

Some tips for when proofreading

- When you notice that a certain error seems to occur multiple times in a document, use XML Spy's Find and Replace function. It can be accessed by going to the "Edit"-menu and clicking on "Replace…" or more quickly by using the key combination "Ctrl-H" and is shown in Figure 17. Usually you will want to use "Replace All". It is really recommended that you use some of the surrounding tags in your search and replace queries to avoid replacing text in unwanted places. Failing to do so will likely have unexpected results. Unfortunately, it is impossible to search for text that bridges two or more lines, meaning that you should use text that falls within a single line.

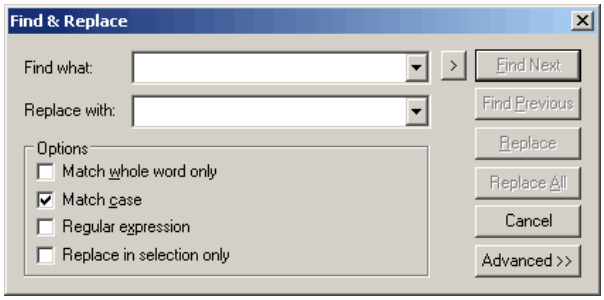


Figure 17: XML Spy's Find and Replace window.

- A checklist of the things you need to keep an eye on is available in Appendix I.
- You can find various XML fragments in Appendix II that will hopefully facilitate the proofreading and correction process.
- XML Spy can help you with its entry helpers, which should be enabled if you properly set up XML Spy. Make use of them. They are explained when encountered later on.
- The proofreading process requires a lot of attention to detail, which can be quite tiring. Take small breaks when you notice that tiredness is starting to impair your judgement. A 5-minute walk and a refreshing drink can do wonders.

How the proofreading is done is explained in more detail in the next sections.

Proofreading Pass 1: Checking well-formedness

As explained above, before actual validation can take place an XML file has to be checked for well-formedness. In this section several examples are shown, together with explanations on how to resolve the issues. In all cases it is useful to use the FlorML XML schema and **FlorML reference.doc** as a reference. When XML Spy encounters an error it will jump to the location of the error and highlight the XML-tag that it thinks is the problem. However, this is not always the location where the actual error is located, as the examples below will show.

1) In Figure 18 the closing </string>-tag is missing, as indicated by the accompanying error message is shown in Figure 19. This is resolved by adding the missing tag right after .

Figure 18: Well-formedness example 1: Missing closing </string>-tag.

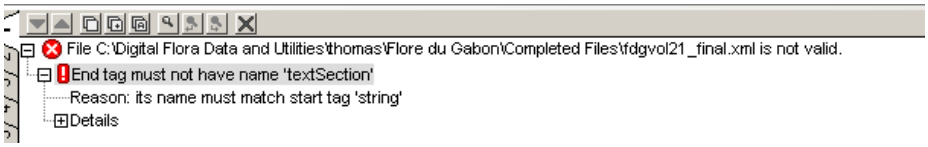


Figure 19: Error message accompanying Well-formedness example 1.

2) Figure 20 shows another problem, with the error message in Figure 21. This is a closing </couplet>-tag that was inserted by a script. It needs to be removed. However, due to the way the scripts work the mark-up for the key will not have the closing </couplet>-tag and closing </key>-tag at the end (Figure 22). These will have to be added (Figure 23). Be sure to respect the indentation.

Now you might have noticed that XML Spy helps you a bit. When you start typing a closing tag ("</"), it helpfully suggests which closing tags could be used at this point. If the correct tag is suggested, just hit the "Enter"-key, and the correct tag is automagically inserted. If multiple closing tags are suggested, select the correct one with the arrow keys (or the mouse) and hit the "Enter"-key followed by ">". If the incorrect tag is suggested, there likely is something wrong elsewhere in the XML above your current position. If you

cannot find that particular error, just type the entire *correct* closing tag and try the check the well-formedness again. Maybe this time you will be able to find the error more easily.

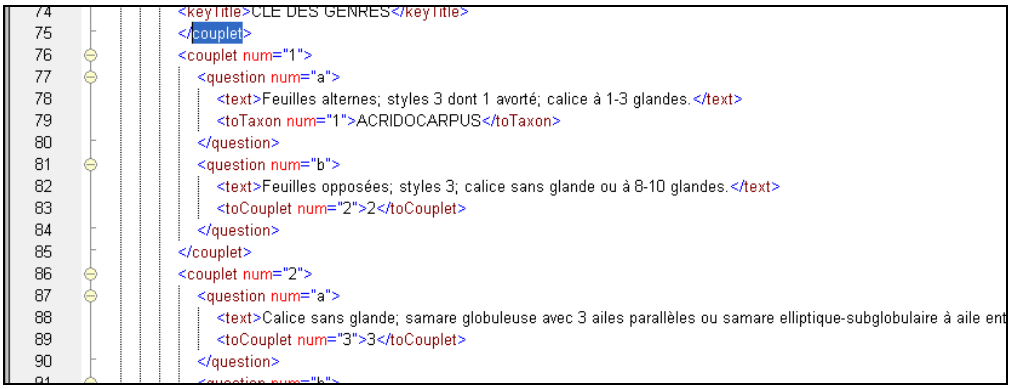


Figure 20: Well-formedness example 2: Excess closing </couplet>-tag at start of key.

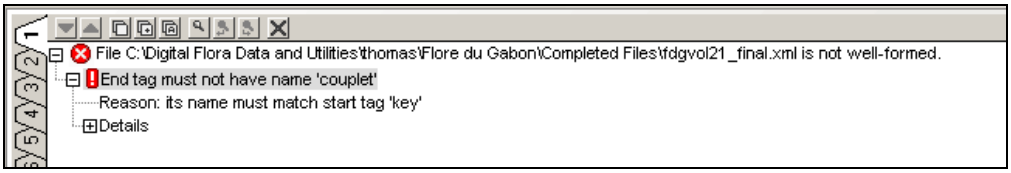


Figure 21: Error message accompanying Well-formedness example 2.

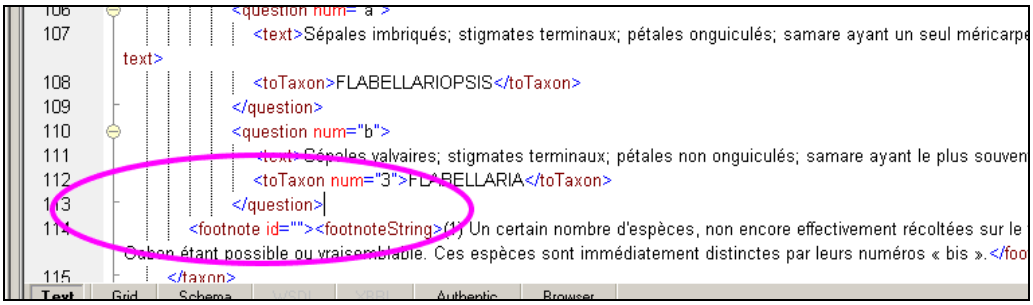


Figure 22: Missing closing </couplet> and </key>-tags at end of key.

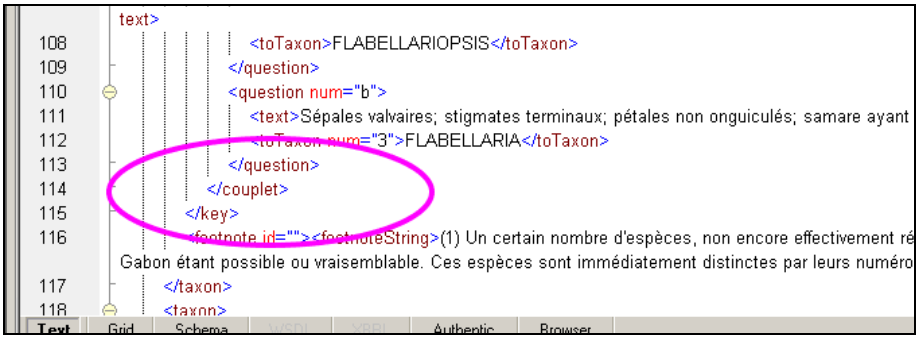


Figure 23: Closing tags inserted.

3) Figure 24 shows yet another problem, with the error message shown in Figure 25. In this case the reason for the error is not right above the closing </taxon>-tag, but a bit further up. If you are observant, you will notice the closing

</string>-tag is not the only one missing, the closing </feature>-tag is also missing. Both will have to be added, taking into account the indentation and removing the
-tag (Figure 26).

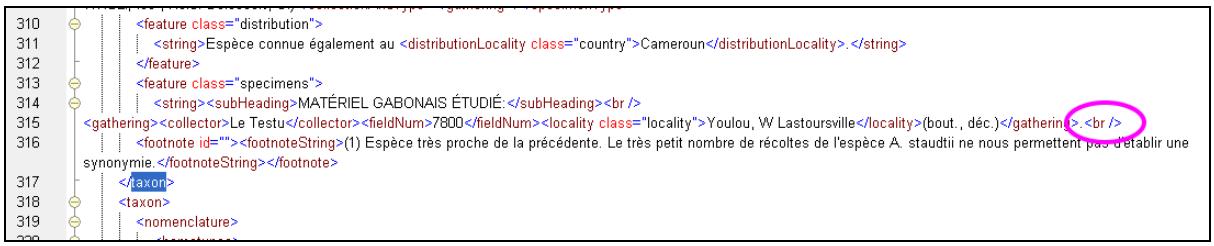


Figure 24: Well-formedness example 3: Missing closing </string> and </feature>-tags.

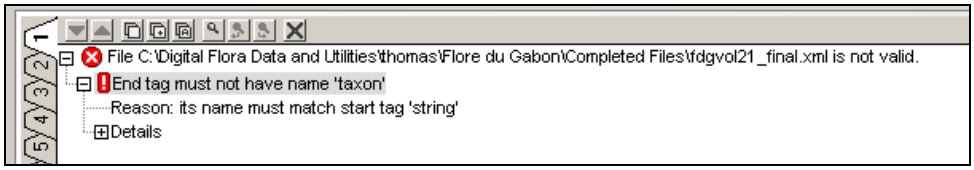


Figure 25: Error message accompanying Well-formedness example 3.

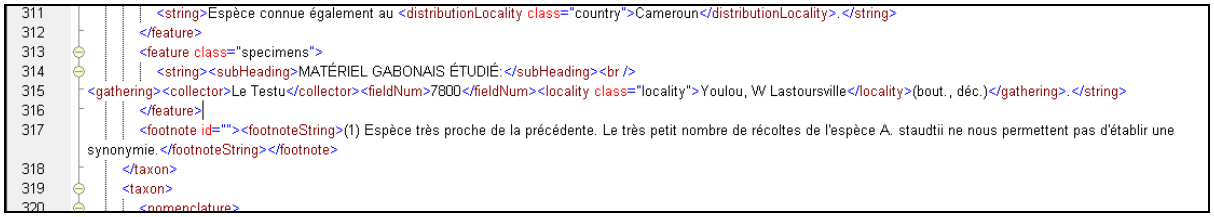


Figure 26: Closing tags inserted.

4) In Figure 27 and Figure 28 the problem is fairly obvious: the closing </nomenclature>-tag is missing and needs to be added. The opening <nomenclature>-tag is not shown, as it is located several hundreds of lines earlier; however the error message indicates one is present.

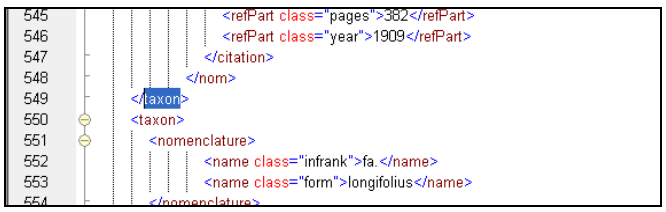


Figure 27: Well-formedness example 4: Missing closing </nomenclature>-tag.

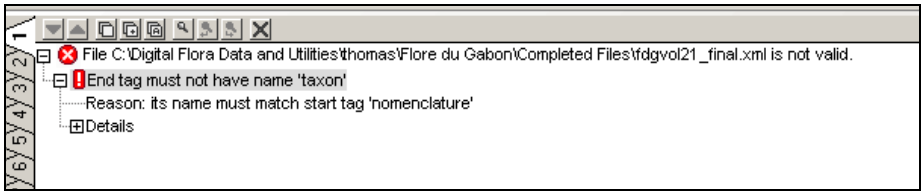


Figure 28: Error message accompanying Well-formedness example 4.

5) Figure 29 and Figure 30 seem similar to example 4). However, looking for the cause requires scrolling up a lot, as it is the opening <nomenclature>-tag that is missing, some 100 lines earlier (Figure 31). Again, respect the indentation.

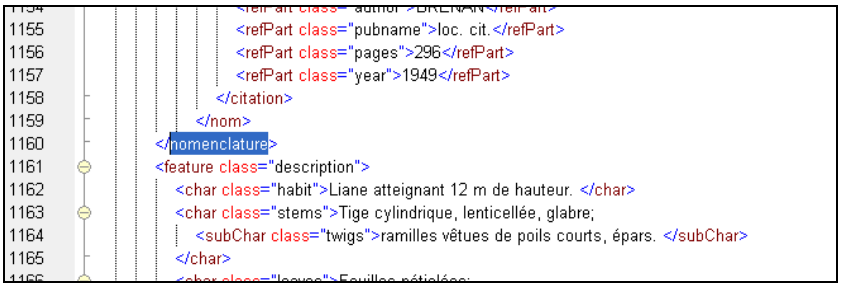


Figure 29: Well-formedness example 5: Missing opening <nomenclature>-tag.

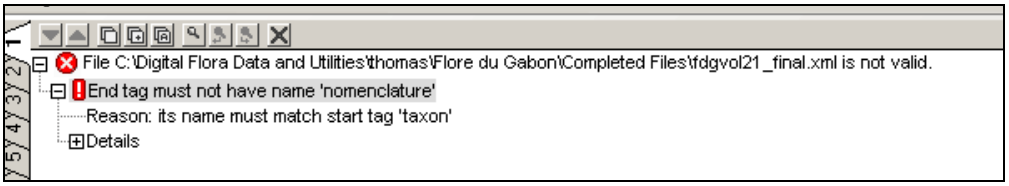


Figure 30: Error message accompanying Well-formedness example 5.

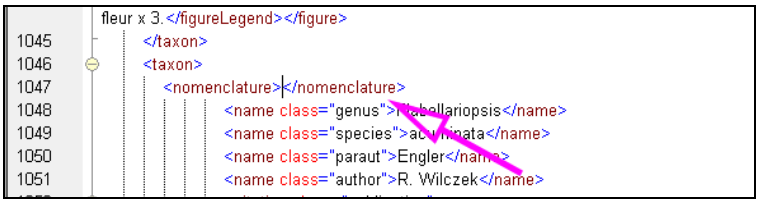


Figure 31: Inserting opening nomenclature tag.

Note that when you type an opening tag in XML Spy, XML Spy auto-completes it with a corresponding closing tag (as indicated by the big arrow in Figure 31). In this particular case you do not want that, so you press the key combination "Ctrl-Z" on your keyboard exactly *once* to remove it.

6) In Figure 32 the closing <subChar>-tag is missing and needs to be inserted, while the closing </char>-tag goes onto a new line, with the proper amount of indentation (Figure 33).

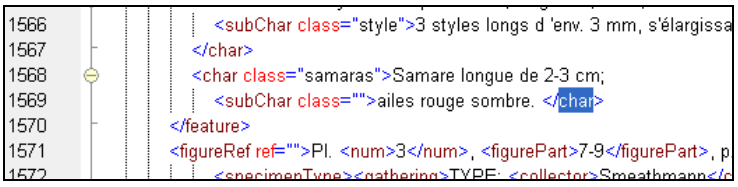


Figure 32: Well-formedness example 6: Missing closing </subChar>-tag.

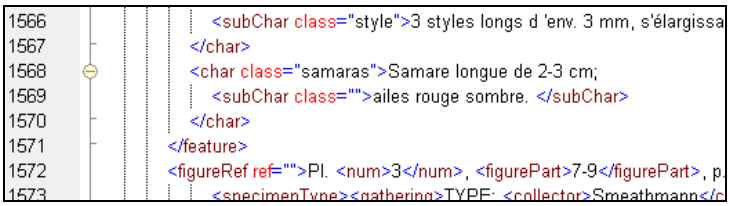


Figure 33: Proper tags inserted.

7) In Figure 34 you can see another example of what may cause problems with well-formedness. An element was accidentally inserted in a place where it does not belong. In this case the element needs to be removed to resolve the problem. Due to this, the text there will then read "fl.fr.". It is suggested that you change this to "fl., fr.".

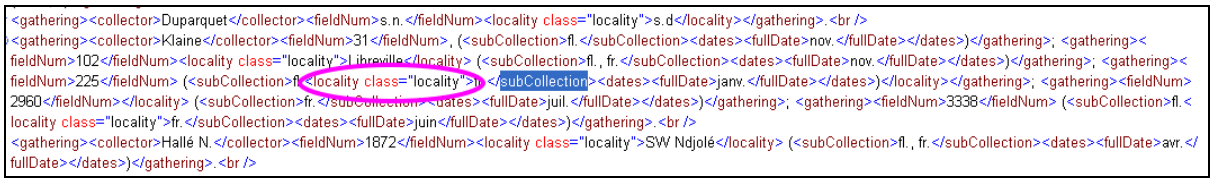


Figure 34: Well-formedness example 7: Wrongly inserted tag.

You can see the same problem two lines lower in the same figure. In such cases it is more efficient to fix both issues at once instead of fixing one issue, saving, fixing the second issue, etc.

These examples should have given you a taste of the type of problems you can encounter during the check for well-formedness. Sometimes you will have to search for the location of the missing tag for a while, but do not be discouraged by that.

You can already correct OCR errors at this stage, but be careful with XML errors that have nothing to do with well-formedness, as you may very well give yourself more work to do because certain entry helpers are disabled when a file is not well-formed. Such errors are best left alone until the next stage, unless you know precisely what you are doing and do it correctly.

When all the problems related to well-formedness have been resolved and the XML is well-formed, the message shown in Figure 35 will be displayed when you use the "Check well-formedness"-function in XML Spy. At this point you can start checking whether the file validates.

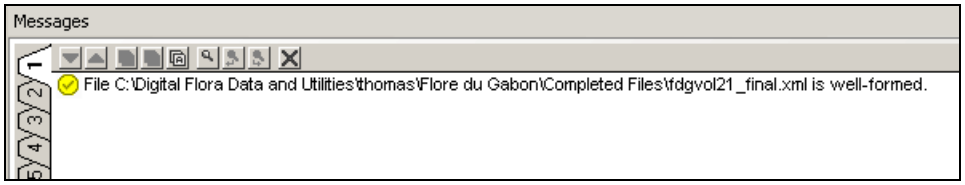


Figure 35: Success message for well-formed XML.

Proofreading Pass 2: Proofreading and validation

During the second pass of proofreading your objective is to obtain a file that both contains valid XML and no major mark-up errors where original text contents is matched to the wrong XML.

You will encounter several types of problems:

- Wrongly positioned tags.
- Missing tags.
- Missing or incomplete attributes.
- Text that was not marked up at all.
- Partially marked up text.
- Misidentified text.
- Text requiring special treatment.
- Text or attribute options not supported by the FlorML schema.
- Text of which the position must be changed.

However, before you start, you should know that the working order is slightly different compared to the first pass discussed above. As mentioned earlier, XML Spy jumps to the next validation error anytime you validate a document. This means that if you try to proofread a document and save it, you will jump to the next validation error instead of staying at the point you were proofreading. There are two possible solutions:

1) Take a note of the line number you are at before saving. Then, after saving, scroll back to that line number. This is a method that is quite error-prone, because you will forget to take note of the line-number at one point.
2) Proofread the document until you encounter a problem of which you know it will not validate, then save. Examples of such problems are essential elements or attributes that are missing. This is very efficient.

Another thing to keep in mind is that any XML validation problems you missed will be found by XML Spy, but not any problems that pertain to text misidentification or the text itself. XML Spy only validates the XML, not the text contents between the XML tags.

Several examples of the possible issues are discussed below. You should keep **FlorML reference.doc** and the FlorML XML schema at hand. You will notice that not all examples have the problems nicely indicated by ovals, arrows or highlighted text. This is because you must learn to spot them in bare XML, where such indicators are often absent.

XML issues

Wrongly positioned tags

Figure 36 shows an example of a wrongly positioned tag, with the error message shown in Figure 37. The problem is not actually that <textSection> is not allowed under <treatment> (which is indeed the case), but that the <treatment>-tag is placed on the wrong location. Figure 38 shows the correct location.

> **Tip:** Selected text can be dragged and dropped.

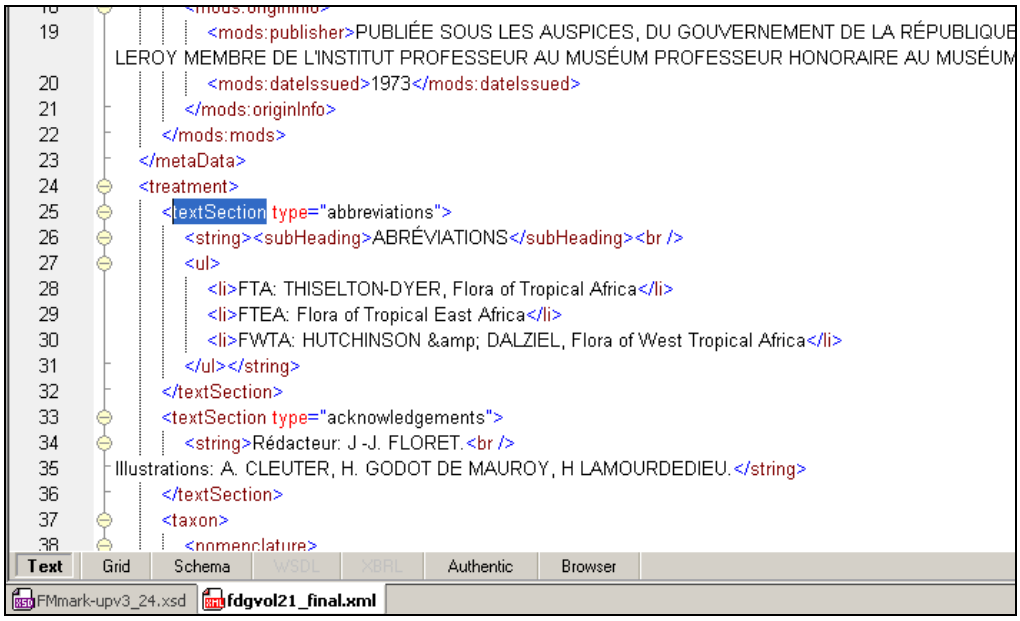


Figure 36: Wrongly positioned tags example 1: <treatment>-tag in wrong position.

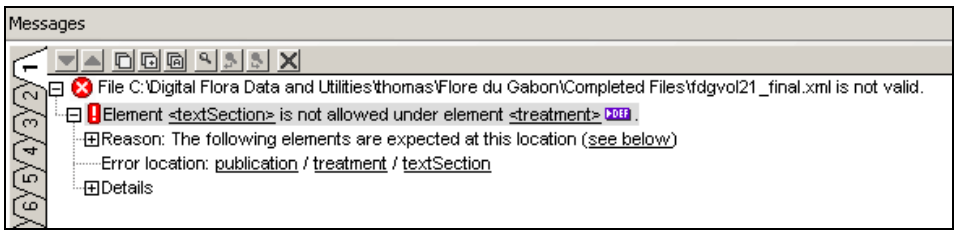


Figure 37: Error message accompanying Wrongly positioned tags example 1.

Figure 38: Wrongly positioned tags example 1: <treatment>-tag in correct position.

Missing tags

1) Figure 39 shows a problem where some tags are missing from the nomenclature section of a taxon treatment, with the error message in Figure 40. Here the required tags are inserted above the family name (Figure 41), after which the family name is dragged into position (Figure 42).

   Observant eyes will have noticed another problem in Figure 39: the closing </nomenclature>-tag is located wrongly; this has also been corrected in Figure 42.

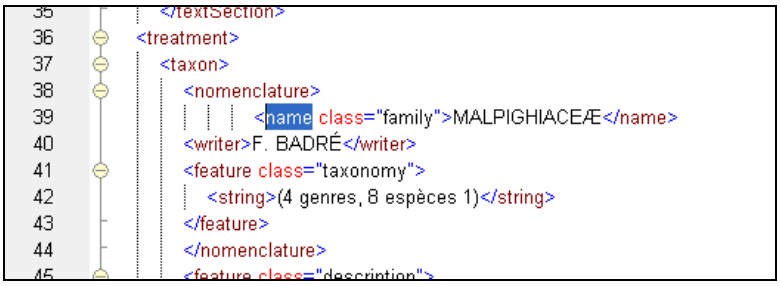


Figure 39: Missing tags example 1: Some tags in the nomenclature section are missing.

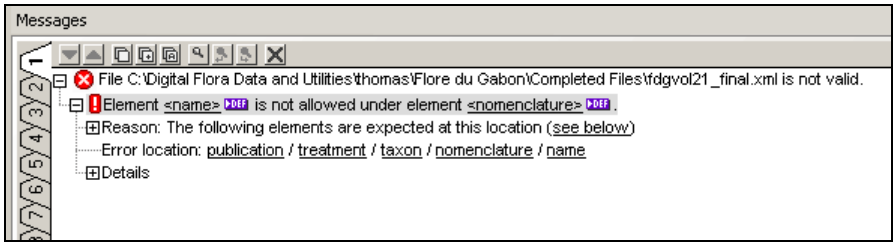


Figure 40: Error message accompanying Missing tags example 1.

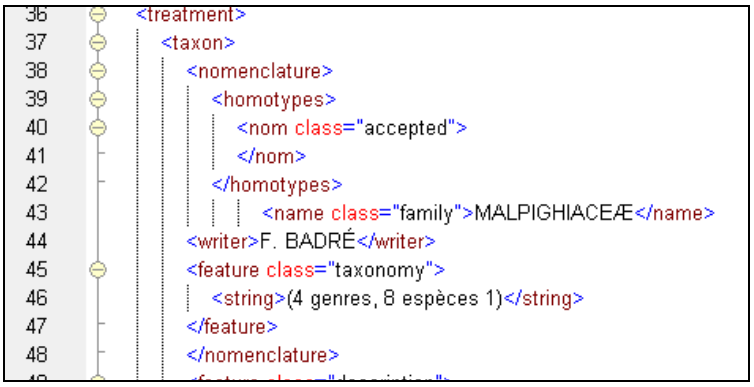


Figure 41: Missing tags example 1: Missing tags inserted.

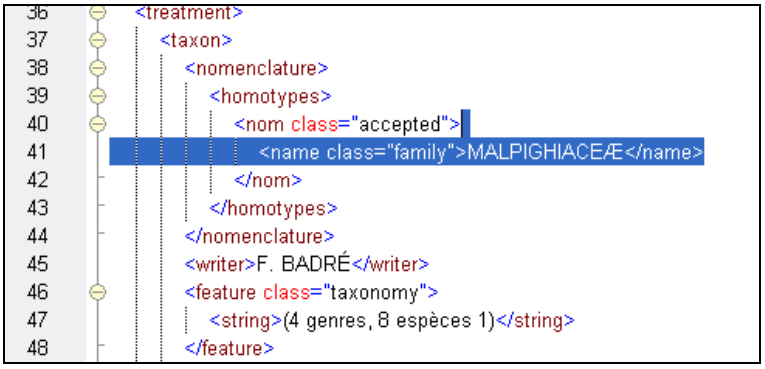


Figure 42: Missing tags example 1: Correct tag order attained.

2) In Figure 43 you can see part of the specimen list for a particular taxon. In three cases the subcollection information of the gathering was not marked up. This kind of error needs to be fixed by adding the tags shown to surround the other subcollection information. Figure 44 shows the corrected version.

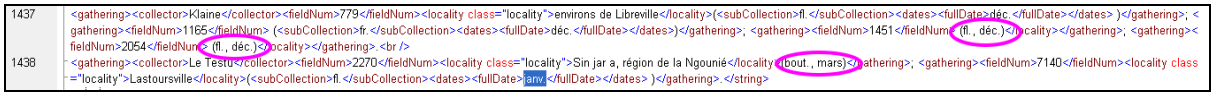


Figure 43: Missing tags example 2: Subcollection tags missing.

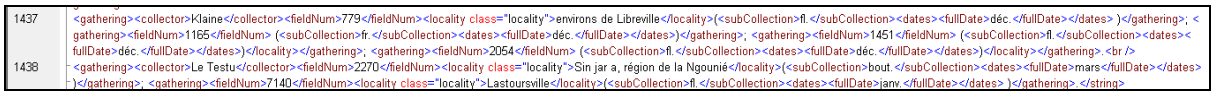


Figure 44: Missing tags example 2: Subcollection tags added.

Missing or incomplete attributes

1) In Figure 45 you can see a piece of text featuring distribution information. The sentence it is located in indicates that the species might occur in Gabon, but still needs to be found. This can be indicated in the XML with a specific attribute that has to be added by hand, as shown in Figure 46.

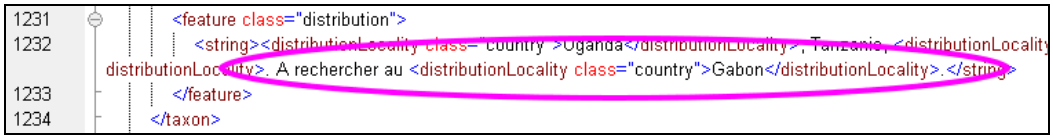


Figure 45: Text that means that additional information will have to be added to an element.

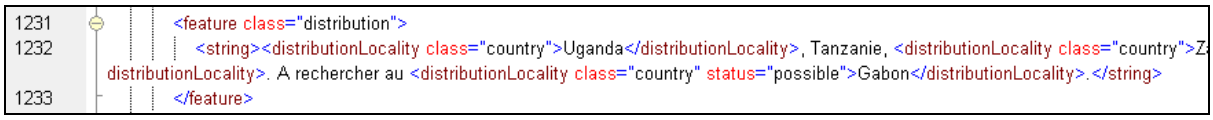


Figure 46: The same text as above, with the additional information added.

This kind of additional information is present in many different types of text (often distribution information, but also types, specimen information, etc.). In a few cases it is consistent enough for its mark-up to be automated, but in most cases the mark-up will have to be added by hand.

2) Figure 47 shows a taxon description where two characters have not been identified. In the first case, the character is "glands". When you put the cursor between the quotation marks and start typing a g, one of XML Spy's entry helpers shows up and you can select the correct character (Figure 48). Hit the "Enter"-key to insert the text (Figure 49). Repeat for the other character. This does not work when the character is not yet included in the FlorML XML schema (see later for information on how to add a character to the XML schema).

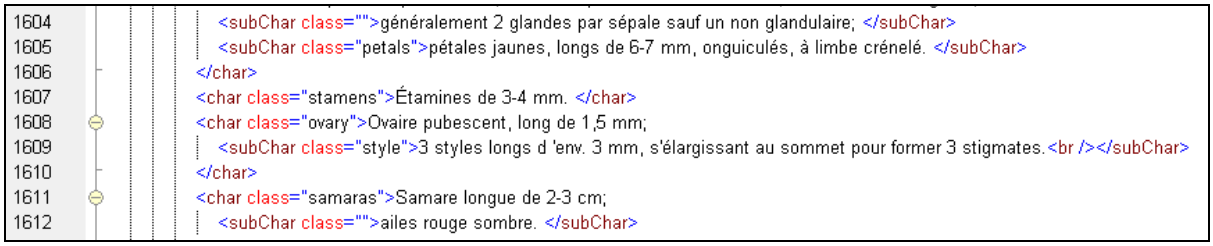


Figure 47: Unidentified characters.

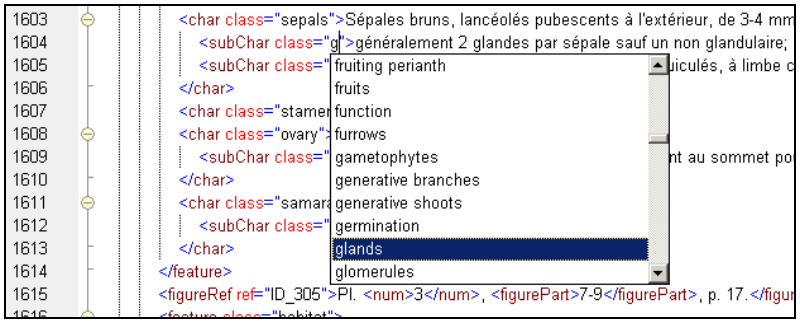


Figure 48: Inserting a character.

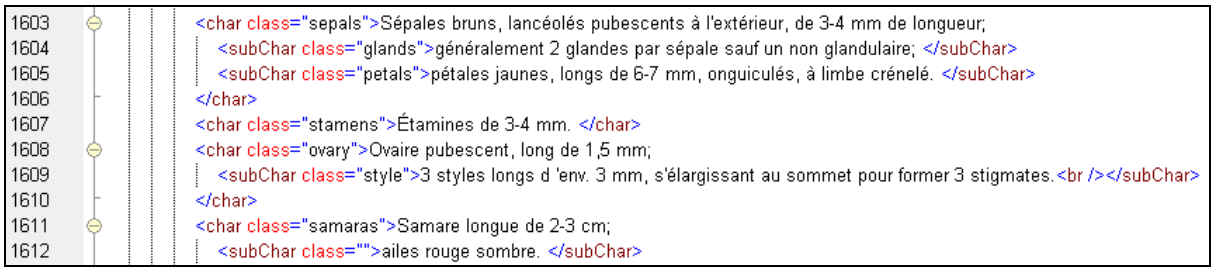


Figure 49: Characters inserted.

Footnotes

Footnotes usually have been properly identified during automated mark-up, but still require some minor additional work. Most importantly, references to footnotes are linked to footnotes using unique identifiers (within a volume) and these have to be added manually. Figure 50 shows a footnote, with its "id" attribute highlighted. In Figure 51 the identifier has been filled in ("FN" stands for "**f**oot**n**ote").

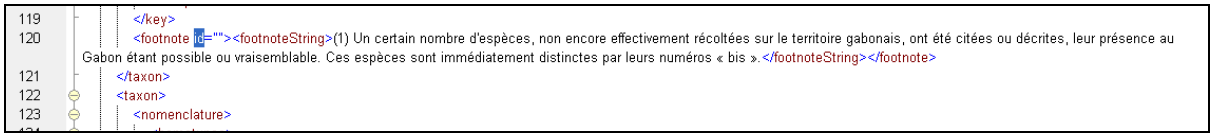


Figure 50: Footnote example with no identifier.

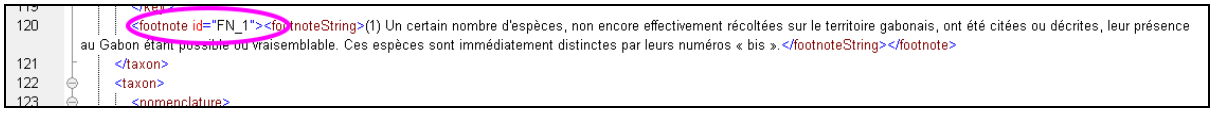


Figure 51: Footnote example with identifier.

Now you have to find the reference (number or asterisk) to the footnote. As these are very hard to recognise in an XML file, it is suggested that you use either a paper copy of the flora or a scan of the page as a reference to locate it. Figure 52 shows where the reference number is located in the XML, and Figure 53 shows the mark-up required.

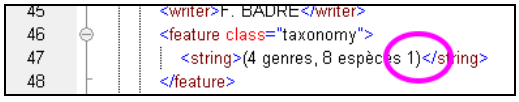


Figure 52: Footnote reference located.

Figure 53: Footnote reference with mark-up linking it to footnote.

Subsequent footnotes should have subsequent identifiers. The actual number used to refer to them in the text (the part between <num> and </num>) does not need to be changed though.

Figures

Figure legends and references to figures should be checked for proper mark-up like all other mark-up, but you also have to add some additional information to them. Figure 54 shows a figure reference (<figureRef>) and a figure legend. The latter is used as a placeholder for the actual figure. Each figure reference and figure needs to have its figure ID filled out (see **FlorML reference.doc** and **image processing.doc** for more information), which can be found in the image administration you or one of your coworkers is keeping. Furthermore, you should add the file name of the figure itself, which can be found in the same image administration, to the mark-up for the figure (Figure 55).

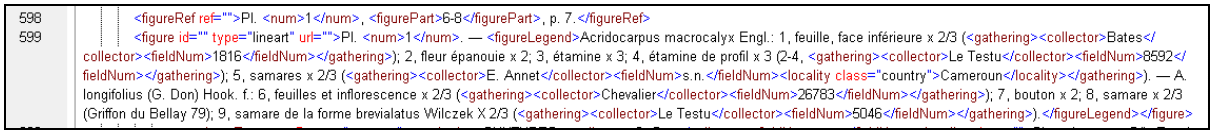


Figure 54: Figure reference and figure example lacking ID and url.

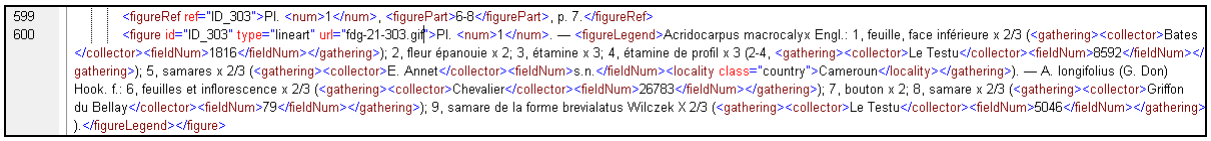


Figure 55: Figure reference and figure example with ID and url.

You probably also noticed that collections used for figures are marked up in the figureLegend. You should check whether this was done properly. In the above example one of the collections was missing mark-up, which was also fixed.

Textual issues

Text that was not marked up at all

Figure 56 shows an example of a citation that was not marked up at all. In these cases, you add the required XML above the problem (Figure 57) and then drag (or cut and paste) all of the text into place (Figure 58).

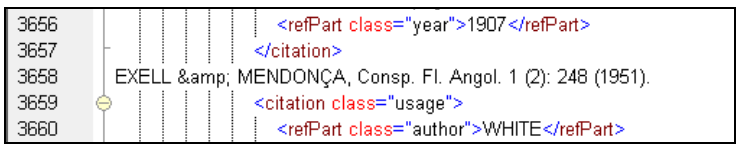


Figure 56: Text without any mark-up example 1: Literature reference not marked up.

```
3657                </citation>
3658                <citation class="usage">
3659                  <refPart class="author"></refPart>
3660                  <refPart class="pubname"></refPart>
3661                  <refPart class="volume"></refPart>
3662                  <refPart class="part"></refPart>
3663                  <refPart class="pages"></refPart>
3664                  <refPart class="year"></refPart>
3665                </citation>
3666  EXELL &amp;amp; MENDONÇA, Consp. Fl. Angol. 1 (2): 248 (1951).
3667                <citation class="usage">
```

Figure 57: Text without any mark-up example 1: Required XML inserted.

```
3657                </citation>
3658                <citation class="usage">
3659                  <refPart class="author">EXELL &amp;amp; MENDONÇA</refPart>
3660                  <refPart class="pubname">Consp. Fl. Angol.</refPart>
3661                  <refPart class="volume">1</refPart>
3662                  <refPart class="part">2</refPart>
3663                  <refPart class="pages">248</refPart>
3664                  <refPart class="year">1951</refPart>
3665                </citation>
3666                <citation class="usage">
```

Figure 58: Text without any mark-up example 1: References manually split up.

Most occurences of text that was not marked up at all occur in nomenclature, citations, references, or types. These are locations where complex pattern matching may fail.

Partially marked up text

1) Figure 59 shows something that sometimes occurs in keys. The number ("3 bis.") preceding the taxon was not marked up in question 4a - this is recognisable in the XML by looking at the <couplet>-tag and the <question>-tag below it. Question b below it shows how it should look, so you just type that out and drag the text for the number into place, giving the result shown in Figure 60.

```
109                <couplet num="4">
110                  <question num="a">
111                    <text>Sépales imbriqués; stigmates terminaux; pétales onguiculés; samare ayant un seul méricarpe développé, celui-ci globuleux avec 3 ailes parallèles . . . 3 bis.</
       text>
112                    <toTaxon>FLABELLARIOPSIS</toTaxon>
113                  </question>
114                  <question num="b">
115                    <text>Sépales valvaires; stigmates terminaux; pétales non onguiculés; samare ayant le plus souvent 3 (rarement 2) méricarpes développés, orbiculaires .</text>
116                    <toTaxon num="3">FLABELLARIA</toTaxon>
117                  </question>
```

Figure 59: Partially marked up text example 1: Number in key not recognized.

```
109                <couplet num="4">
110                  <question num="a">
111                    <text>Sépales imbriqués; stigmates terminaux; pétales onguiculés; samare ayant un seul méricarpe développé, celui-ci globuleux avec 3 ailes parallèles</text>
112                    <toTaxon num="3 bis">FLABELLARIOPSIS</toTaxon>
113                  </question>
114                  <question num="b">
115                    <text>Sépales valvaires; stigmates terminaux; pétales non onguiculés; samare ayant le plus souvent 3 (rarement 2) méricarpes développés, orbiculaires .</text>
116                    <toTaxon num="3">FLABELLARIA</toTaxon>
117                  </question>
```

Figure 60: Partially marked up text example 1: Correction made.

2) Figure 61 shows several problems in some citations. From top to bottom, you encounter the following: 1) a status that was partially atomised, but that also starts with the publication date, 2) a footnote reference that was misidentified as details (this was noticed by comparing the XML to a scan), and 3) a publication date that was misidentified. Observant eyes with some experience in FlorML mark-up will also notice the wrongly positioned closing </nom> and </homotypes>-tags at the top.

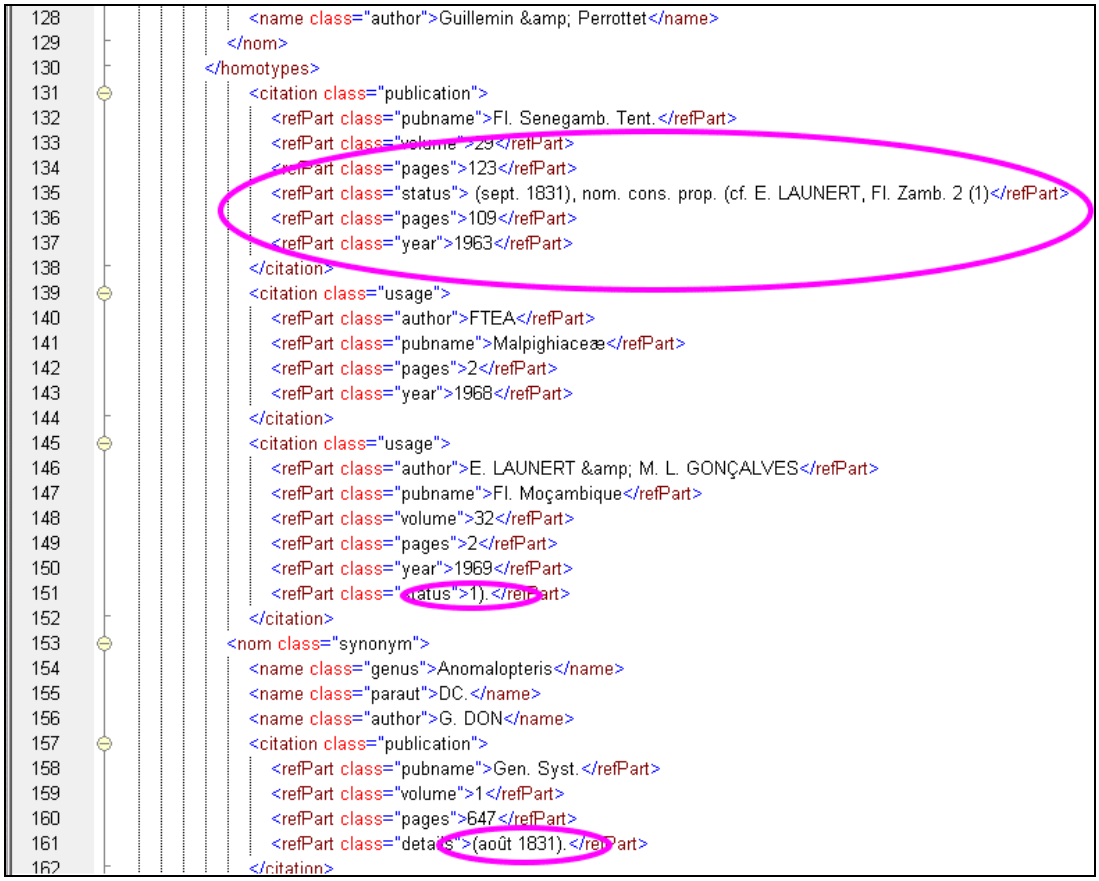


Figure 61: Partially marked up text example 2: Several problems.

In Figure 62 all of these problems have been resolved. Look well to see how.

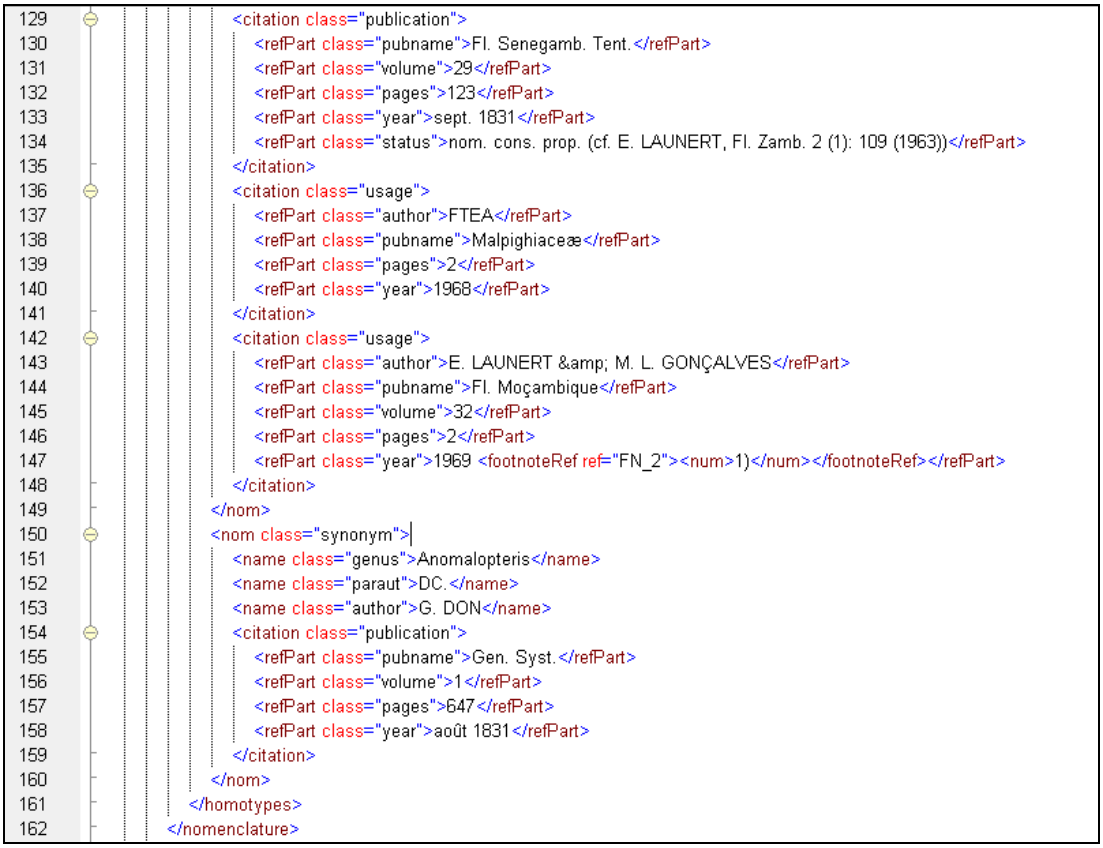


Figure 62: Partially marked up text example 2: All problems fixed.

3) Figure 63 shows a publication with a volume number that has not been properly marked up (because the space is missing between the publication name and volume number) and also a taxonomic name that was not marked up properly. First, the proper tags are inserted (Figure 64), and then the text is dragged into place and assigned to the correct category (Figure 65).

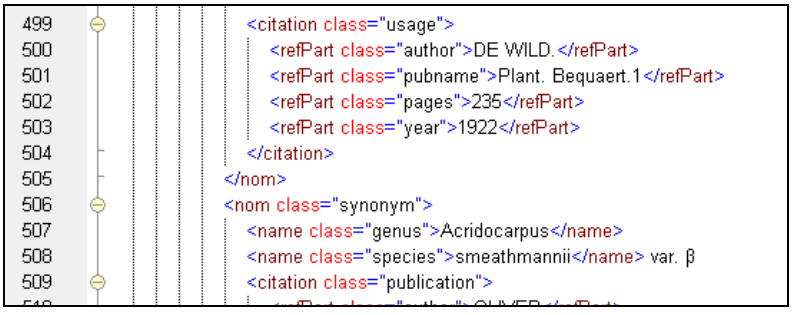


Figure 63: Partially marked up text example 3: Volume number and variety inferior rank and name not marked up.

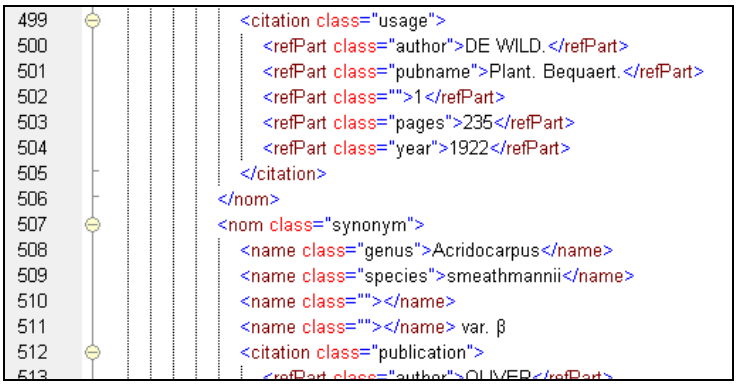


Figure 64: Partially marked up text example 3: Tag insertion.

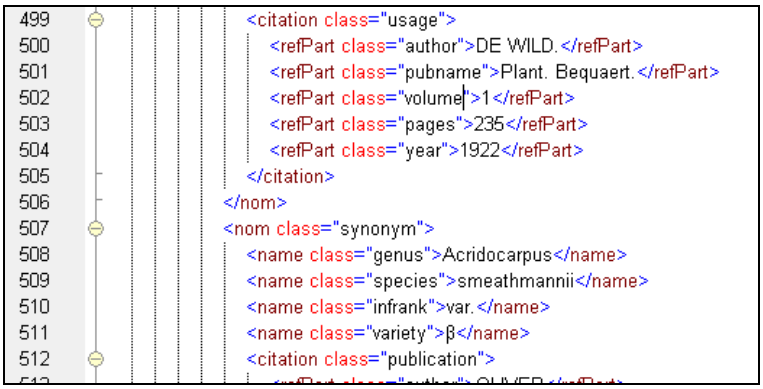


Figure 65: Partially marked up text example 3: Problems fixed.

4) Figure 66 shows another example of this problem, with several other references having the next problem: misidentified text. How would you solve this?

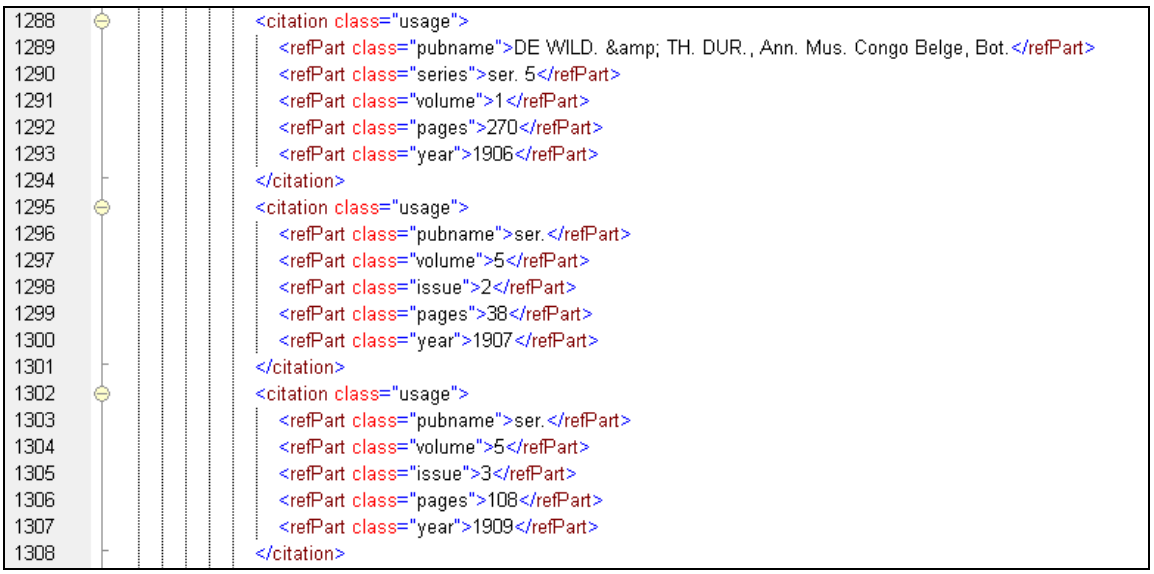


Figure 66: Partially marked up text and text that has been split up too far.

Misidentified text

Figure 67 shows a case where an author name has been identified as a vernacular name, and subsequently the accompanying citation also suffered some misidentifications. This kind of problem usually occurs due punctuation errors in combination with certain text formats and sometimes requires quite a bit of fixing. It can occur anywhere in text where the scripts have to perform complicated patterns matching. The fix for this is shown in Figure 68 and consists of removing the wrong tags and inserting the correct ones. It is worthwhile to highlight one change that is easily forgotten: changing "publication" to "usage" because this citation is one where a name is used, not the one where the name was first published in.

```
1317                <refPart class="volume">1</refPart>
1318                <refPart class="pages">233</refPart>
1319                <refPart class="year">1922</refPart>
1320              </citation>
1321            </nom>
1322            <nom class="synonym">
1323              <name class="vernacular">NIEDENZU</name>
1324              <citation class="publication">
1325                <refPart class="author">Pflanzenr.</refPart>
1326                <refPart class="pubname">Malpighiaceæ</refPart>
1327                <refPart class="pages">38</refPart>
1328                <refPart class="year">1928</refPart>
1329              </citation>
```

Figure 67: Misidentified text example.

```
1317                <refPart class="volume">1</refPart>
1318                <refPart class="pages">233</refPart>
1319                <refPart class="year">1922</refPart>
1320              </citation>
1321              <citation class="usage">
1322                <refPart class="author">NIEDENZU</refPart>
1323                <refPart class="pubname">Pflanzenr.</refPart>
1324                <refPart class="pubtitle">Malpighiaceæ</refPart>
1325                <refPart class="pages">38</refPart>
1326                <refPart class="year">1928</refPart>
1327              </citation>
```

Figure 68: Misidentified text fixed.

If you have trouble figuring out what goes where in a case like this, have a look at the printed treatment or a scan of it.

Text requiring special treatment

In some cases, text in taxonomic works is printed in italics or bold to emphasize that a certain taxonomic name or character is important. This is often mentioned in the text itself.

Figure 69 shows a fragment from a description where a characteristic character of the taxon described is printed in bold. Figure 70 shows the corresponding XML. As you can see, the corresponding XML lacks the bolding because it was stripped off during the mark-up process. This kind of information has to be added manually. For bold text the tags and are used. These are placed around the text that should appear in bold (Figure 71).

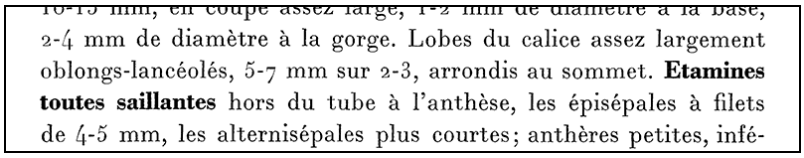

10-15 mm, en coupe assez large, 1-2 mm de diamètre à la base, 2-4 mm de diamètre à la gorge. Lobes du calice assez largement oblongs-lancéolés, 5-7 mm sur 2-3, arrondis au sommet. **Etamines toutes saillantes** hors du tube à l'anthèse, les épisépales à filets de 4-5 mm, les alternisépales plus courtes; anthères petites, infé-

Figure 69: Bolded text to emphasize a character in a printed description.

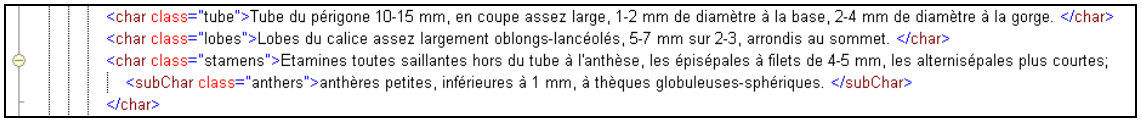


Figure 70: Corresponding XML.

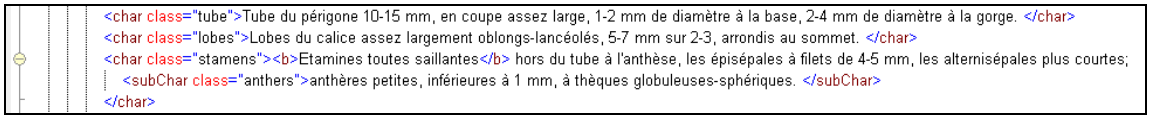


Figure 71: Corresponding XML with bold tags inserted.

For italics you should use the and tags.

Keys or text referring to taxa with no taxon treatment in flora

Sometimes a key may refer to a taxon that does not actually have its own taxon treatment in the legacy taxonomic work volume concerned. These often are taxa that occur outside the geographic focus area of the taxonomic work, which is usually mentioned in the key itself.

Furthermore, sometimes infrageneric or infraspecific taxa are mentioned in a note under a parent taxon. These may be taxa that do not occur in the geographic area concerned by the taxonomic work, but sometimes may also be subspecies and varieties that simply did not get their own entry.

For such taxa, separate taxon entries need to be created by hand. Please see **FlorML reference.doc** for the elements to use.

Text or attribute options not supported by the FlorML schema

Figure 72 shows a character that has not been encountered before and that therefore is absent from the FlorML XML schema. It will have to be added to FlorML, otherwise the character in question will remain unsupported and the XML document will not validate. The accompanying error message is shown in Figure 73.

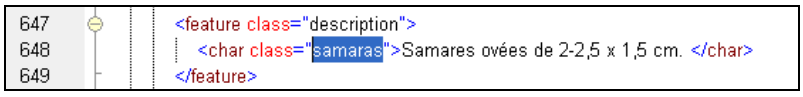


Figure 72: Unsupported descriptive character.

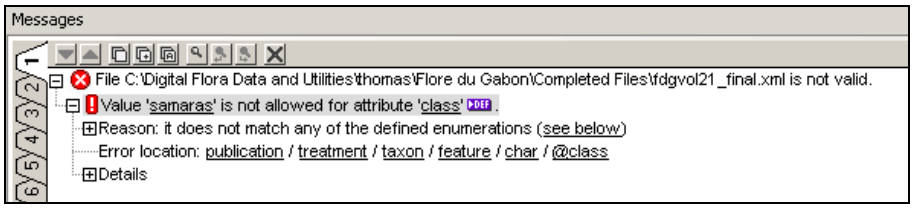


Figure 73: Error message for unsupported descriptive character.

To add the character to the FlorML XML schema, go to the schema (in graphical view as explained earlier), and expand the schema by clicking on the plusses until you find the <char>-element. Expand its "attributes"-box and select "class", as shown in Figure 74.

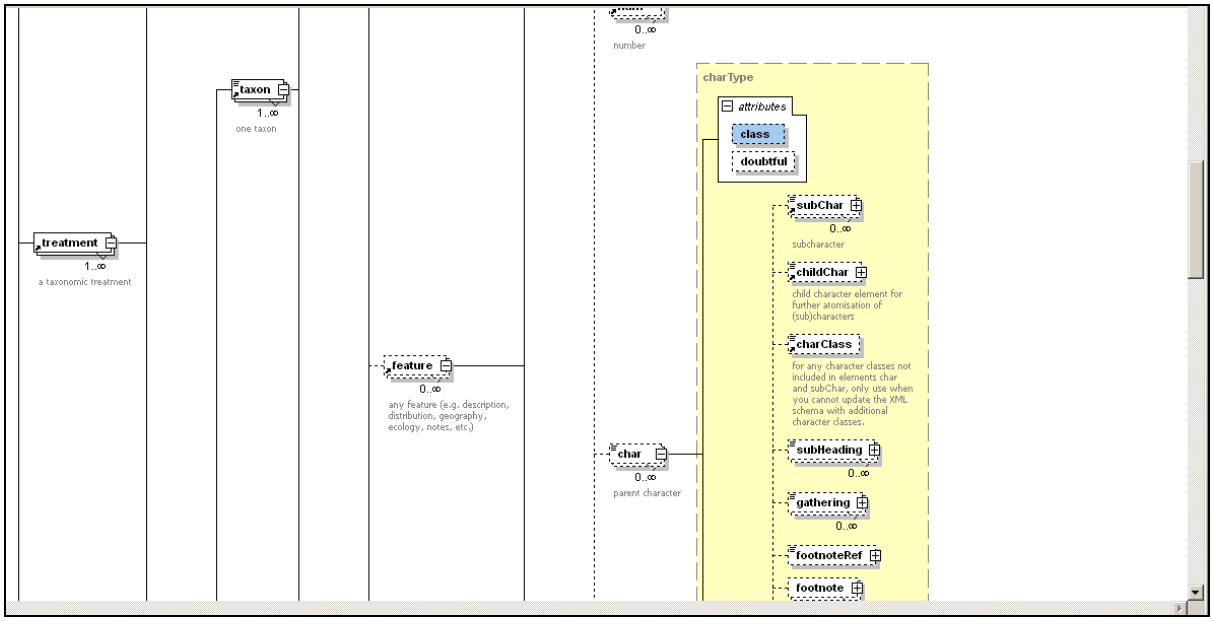


Figure 74: Adding a character class to the FlorML XML schema.

Then open the "Facets"-tab in XML Spy and select "Ennumeration" at the bottom. This contains an alphabetic list of possible characters (Figure 75). Find the location where you have to insert the new character, select the character that will be just below it, and click the "Insert"-button (Figure 76).

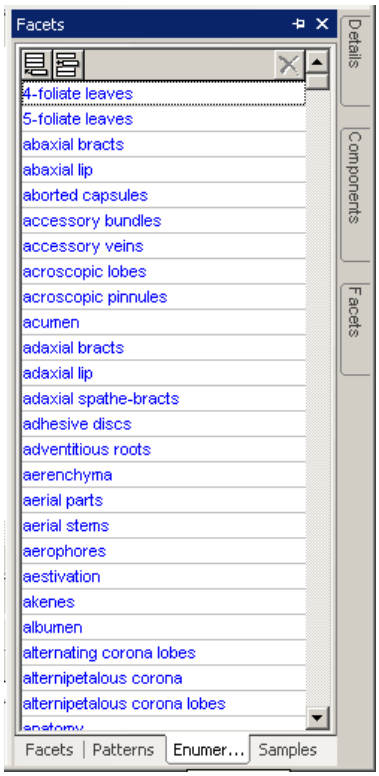


Figure 75: "Ennumeration"-tab in "Facets"-tab.

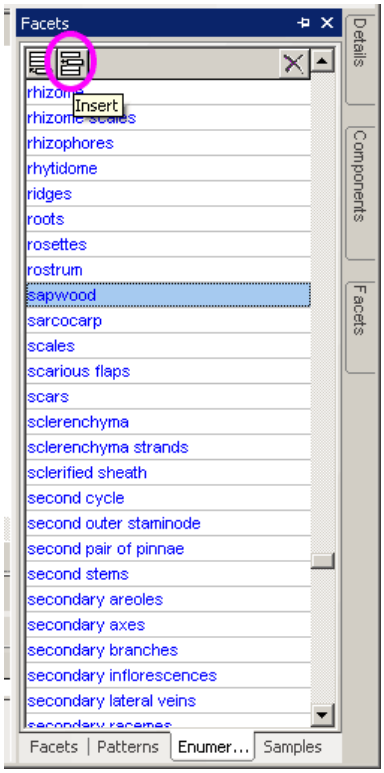


Figure 76: Inserting a new character in the list of characters.

After clicking on the "Insert"-button, a blank line should appear just above the character you selected. Select this blank line and type the name of the character you want to add (Figure 77).

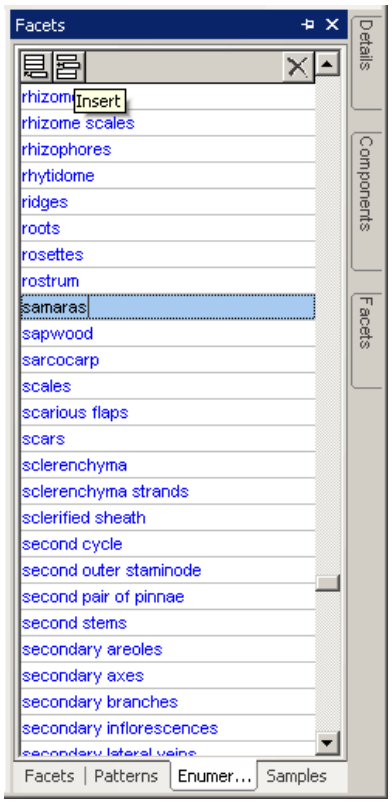


Figure 77: Inserting the new character.

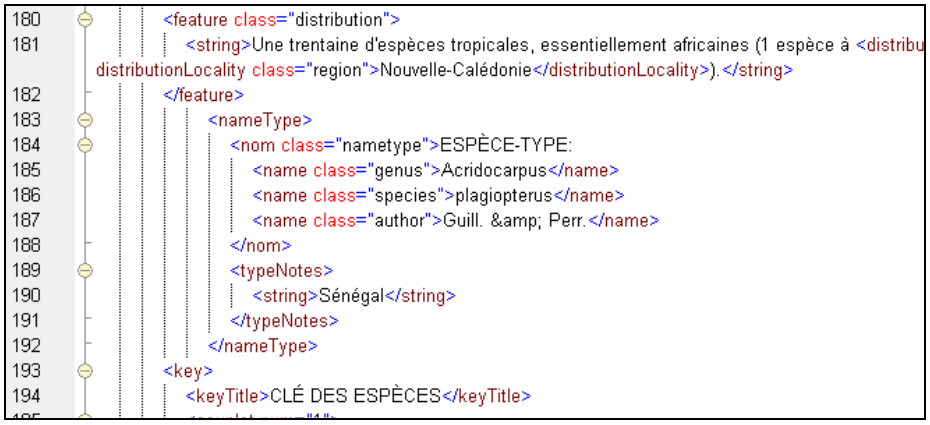
Now repeat the steps above to add the character to the list of possible subcharacters (<subChar>-element).

Finally, save the schema. Then go back to the XML document you were working on and try to validate it again. Normally, it should work. Otherwise, you did something wrong.

> It is **very important** (**!!!**) that you understand that this should only be done when you encounter a character or feature that cannot be accommodated at all within the boundaries of the current XML schema. If you are not the person who is responsible for the FlorML XML schema and you have doubts, ask the person responsible for advice.
>
> Another thing to keep in mind is to try to not add characters both in singular and plural. Use either singular or plural, but not both.

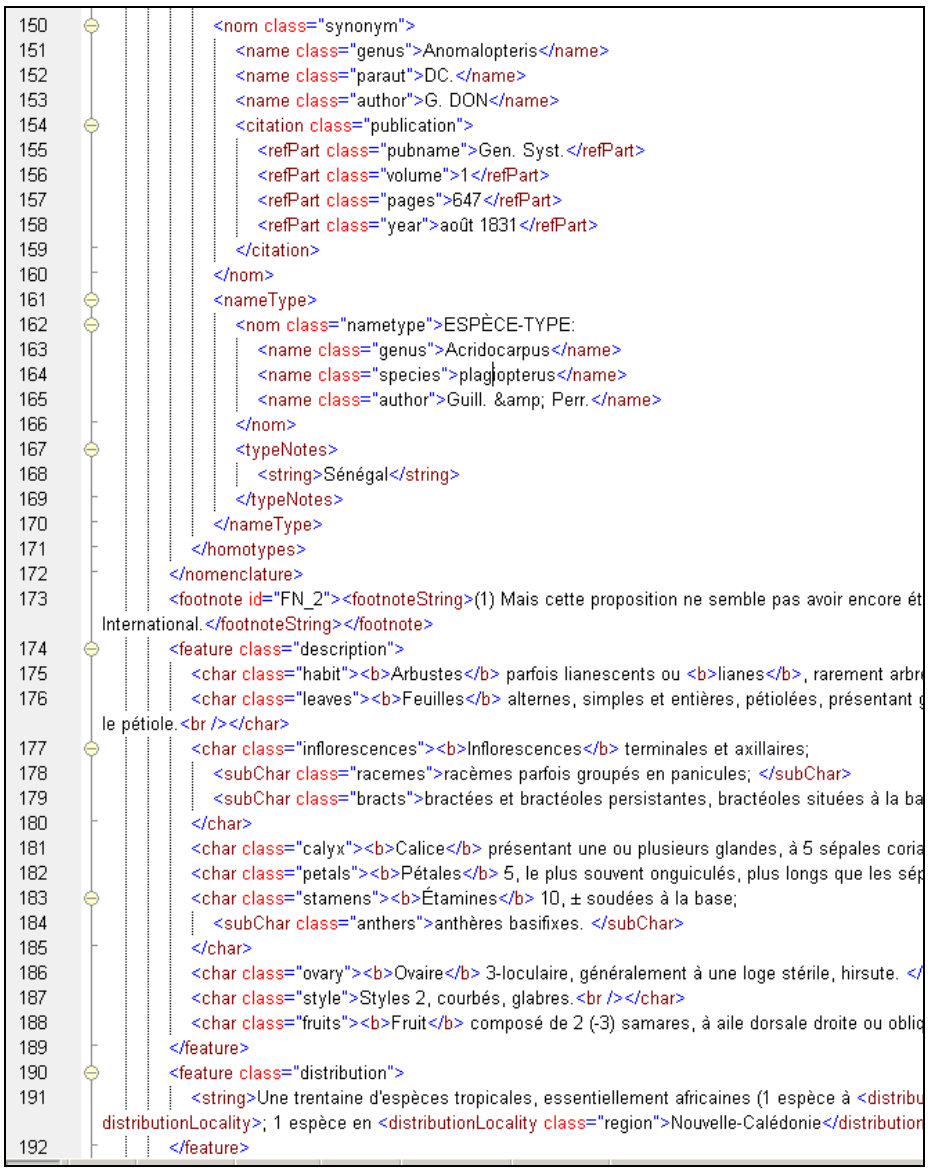
Text of which the position must be changed

In some cases, text in taxonomic works is not located at all where FlorML expects it to be. For example, the flora Flore du Gabon has its types separated from the rest of the nomenclature. These will need to be moved to the correct position. An example is shown in Figure 78, with Figure 79 showing the type in the proper position (you can see the distribution information shown at the top in Figure 78 at the bottom of Figure 79).



```
180         <feature class="distribution">
181           <string>Une trentaine d'espèces tropicales, essentiellement africaines (1 espèce à <distribu
    distributionLocality class="region">Nouvelle-Calédonie</distributionLocality>).</string>
182         </feature>
183         <nameType>
184           <nom class="nametype">ESPÈCE-TYPE:
185             <name class="genus">Acridocarpus</name>
186             <name class="species">plagiopterus</name>
187             <name class="author">Guill. &amp;amp; Perr.</name>
188           </nom>
189           <typeNotes>
190             <string>Sénégal</string>
191           </typeNotes>
192         </nameType>
193       <key>
194         <keyTitle>CLÉ DES ESPÈCES</keyTitle>
```

Figure 78: Wrongly located type in Flore du Gabon.



```
150           <nom class="synonym">
151             <name class="genus">Anomalopteris</name>
152             <name class="paraut">DC.</name>
153             <name class="author">G. DON</name>
154             <citation class="publication">
155               <refPart class="pubname">Gen. Syst.</refPart>
156               <refPart class="volume">1</refPart>
157               <refPart class="pages">647</refPart>
158               <refPart class="year">août 1831</refPart>
159             </citation>
160           </nom>
161           <nameType>
162             <nom class="nametype">ESPÈCE-TYPE:
163               <name class="genus">Acridocarpus</name>
164               <name class="species">plagiopterus</name>
165               <name class="author">Guill. &amp;amp; Perr.</name>
166             </nom>
167             <typeNotes>
168               <string>Sénégal</string>
169             </typeNotes>
170           </nameType>
171         </homotypes>
172       </nomenclature>
173       <footnote id="FN_2"><footnoteString>(1) Mais cette proposition ne semble pas avoir encore ét
    International.</footnoteString></footnote>
174       <feature class="description">
175         <char class="habit"><b>Arbustes</b> parfois lianescents ou <b>lianes</b>, rarement arbr
176         <char class="leaves"><b>Feuilles</b> alternes, simples et entières, pétiolées, présentant
    le pétiole.<br /></char>
177         <char class="inflorescences"><b>Inflorescences</b> terminales et axillaires;
178           <subChar class="racemes">racèmes parfois groupés en panicules; </subChar>
179           <subChar class="bracts">bractées et bractéoles persistantes, bractéoles situées à la ba
180         </char>
181         <char class="calyx"><b>Calice</b> présentant une ou plusieurs glandes, à 5 sépales coria
182         <char class="petals"><b>Pétales</b> 5, le plus souvent onguiculés, plus longs que les sép
183         <char class="stamens"><b>Étamines</b> 10, ± soudées à la base;
184           <subChar class="anthers">anthères basifixes. </subChar>
185         </char>
186         <char class="ovary"><b>Ovaire</b> 3-loculaire, généralement à une loge stérile, hirsute. </
187         <char class="style">Styles 2, courbés, glabres.<br /></char>
188         <char class="fruits"><b>Fruit</b> composé de 2 (-3) samares, à aile dorsale droite ou oblic
189       </feature>
190       <feature class="distribution">
191         <string>Une trentaine d'espèces tropicales, essentiellement africaines (1 espèce à <distribu
    distributionLocality>; 1 espèce en <distributionLocality class="region">Nouvelle-Calédonie</distributior
192         </feature>
```

Figure 79: Type in proper position.

Successful validation

Figure 80 shows the message you receive when a file successfully validates.

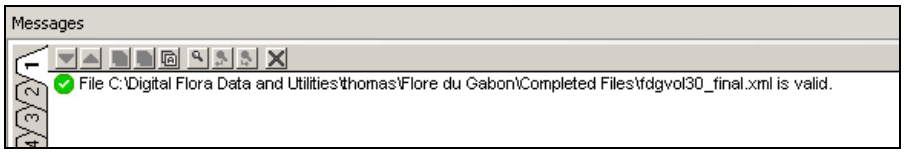


Figure 80: Success message for valid XML.

Concluding notes

These examples should have given you a taste of the kind of problems you can encounter during checking for well-formedness of the XML. In general, making the corrections consists of learning what kind of errors can occur and what their solutions are, together with keeping alert while proofreading to spot errors that XML Spy cannot see.

Appendix I contains a checklist that can be helpful while proofreading.

If you notice a particular error occurs repeatedly, it is worthwhile to take note of it and see whether the person creating the scripts can do something about it for later volumes.

It can also be worthwhile to see whether you can use Notepad++ to easily fix reoccurring errors for the current volume. Appendix III contains some examples of things you can try in Notepad++ to track down any errors that you may have missed when proofreading in XML Spy.

Perhaps you may also find that, as your knowledge of the XML schema increases, you prefer proofreading in Notepad++ and only using XML Spy to fix major problems.

If you do not manage to make all the corrections in one pass, you can try first fixing the XML so the file validates in XML Spy, and then proofreading the file in one or more passes. However, you will probably notice that it is more practical to fix errors when you encounter them, as you might miss them on a second proofreading.

Appendix I: Checklist for proofreading

The following table is not all-inclusive, but should help you get started. See **FlorML reference.doc** for more information on the various elements.

Checking well-formedness	**Yes/No**
Do all opening tags present have their closing tags?	
Do all closing tags present have their opening tags?	
Validation	
Are all required elements in place at the correct locations?	
Are all required attributes filled out?	
Proofreading	
Publication level:	
Are the publication, treatment, and metadata tags properly inserted?	
Is the metadata properly identified?	
Non-taxonomic matter:	
Is all non-taxonomic text properly identified and marked up?	
Taxa:	
Is each separate taxon surrounded by <taxon>-tags?	
Are taxa belonging to a specific class (e.g. introduced species) properly identified?	
Nomenclature:	
Is the nomenclature of each taxon properly identified?	
Is all contents in nomenclature properly atomised?	
Is all atomised contents in nomenclature properly identified?	
Is all contents in citations properly atomised?	
Is all atomised contents in citations properly identified?	
Types:	
Is all contents in types properly atomised?	
Is all atomised contents in types properly identified?	
Keys:	
Is each key properly marked up?	
Is the numbering of the questions in each key correct?	
Is the numbering of the taxa in each key correct?	
Is the numbering leading to questions in each key correct?	
Does a key refer to one or more other keys?	
Features:	
Is each separate feature properly identified and marked up?	
Descriptions:	
Is all contents in descriptions properly atomised?	
Is all contents in descriptions properly identified?	
Are there no duplicate characters in the descriptions?	
Are there no duplicate subcharacters in the descriptions?	
Distributions:	
Is all contents in distributions properly atomised?	
Is all contents in distributions properly identified?	

Is any additional information in distributions properly marked up?	
Specimens:	
Is all contents in specimens properly atomised?	
Is all contents in specimens properly identified?	
Vernacular names:	
Is all contents in vernacular names properly atomised?	
Is all contents in vernacular names properly identified?	
Tables and Lists:	
Are all tables and lists properly marked up?	
Figures:	
Are all figures properly marked up?	
Are all references to figures properly marked up?	
Are all figure identifiers correct?	
Are all figure URLs correct?	
Is all specimen information in figure captions properly marked up?	
Footnotes:	
Are all footnotes properly marked up?	
Are all references to footnotes properly marked up?	
Are all footnote identifiers correct?	
References:	
Have all references that need to be atomised been found?	
Is all contents in references properly atomised?	
Is all contents in references properly identified?	
Emphasis:	
Has all text with special emphasis been properly marked up?	
Other contents:	
Are all (sub)headings properly marked up?	
Are all writers properly marked up?	
Is all text that has to link to a specific table, key, or reference properly marked up using the correct identifiers?	
Is superscript or subscript text properly marked up?	
Are linebreaks present in the correct places?	
Are there any name, refPart, collector, or fieldNum tags with no contents?	
Are there any page numbers present in <refPart class="details"> tags?	
Have all editor names in citations/references been split off?	
Are there any non-standard abbreviations present, e.g. "ssp." instead of "subsp."?	
Optional: Errata/Addenda/Corrigenda[2]:	
Are any errata/addenda/corrigenda properly marked up?	
Are any errata/addenda/corrigenda properly atomised?	
Are any atomised errata/addenda/corrigenda properly identified?	

[2] In the current approach the favored option is integrating errata etc. into the main text prior to clean-up of the volume. Therefore, this part only applies if you chose not to do so.

Appendix II: Sample XML for easier correction during proofreading

The following small pieces of XML can be used to quickly add XML to an XML document in those cases where a large portion of text was not properly marked up. After copying and pasting such a fragment, you can select the offending text and drag it into place. This is less error-prone than having to type all the elements manually and allows a faster work pace.

```
<publication xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="" xmlns:mods="http://www.loc.gov/mods/v3" lang="">
```

When using MODS, precede MODS element names by "mods:"

```
<mods:mods>
```

```
<mods:mods>
    <mods:titleInfo>
        <mods:title></mods:title>
        <mods:partNumber></mods:partNumber>
        <mods:partName></mods:partName>
    </mods:titleInfo>
    <mods:name type="personal">
        <mods:description></mods:description>
        <mods:namePart></mods:namePart>
        <mods:affiliation></mods:affiliation>
    </mods:name>
    <mods:originInfo>
        <mods:publisher></mods:publisher>
        <mods:dateIssued></mods:dateIssued>
    </mods:originInfo>
    <mods:identifier></mods:identifier>
```

```
        </mods:mods>


<textSection type="">
        <string><heading></heading><br /></string>
</textSection>


                <textSection type="">
                        <string><heading></heading><br /></string>
                </textSection>


                <table>
                        <tr>
                                <th></th>
                                <th></th>
                                <th></th>
                        </tr>
                        <tr>
                                <td></td>
                                <td></td>
                                <td></td>
                        </tr>
                </table>


                <ul>
                        <li></li>
                </ul>
```

```
<ol>
        <li></li>
</ol>

<feature class="description"></feature>

<feature class="">
        <string><subHeading></subHeading></string>
</feature>

<feature class="">
        <heading></heading>
        <string></string>
</feature>

<feature class="">
        <string></string>
</feature>

<feature class="habitat">
        <string><habitat></habitat></string>
</feature>

<habitat></habitat>

<feature class="distribution">                    <string></string>
</feature>
```

```
                    <feature class="specimens">
                        <string><subHeading>Specimens examined:</subHeading>
                        <gatheringGroup geoscope="guyana">Guyana:
<gathering></gathering>
                        </gatheringGroup>
                        <gatheringGroup geoscope="suriname">Suriname:
<gathering></gathering>
                        </gatheringGroup>
                        <gatheringGroup geoscope="french guiana">French Guiana:
<gathering></gathering>
                        </string>
                    </feature>

                        </couplet>
                        <couplet num="">
                            <question num="a">
                                <text></text>
                                <toTaxon num=""></toTaxon>
                            </question>
                            <question num="b">
                                <text></text>
                                <toTaxon num=""></toTaxon>
                            </question>
                        </couplet>

                        </couplet>
```

```
                <couplet num="">
                    <question num="a">
                        <text></text>
                        <toTaxon></toTaxon>
                    </question>
                    <question num="b">
                        <text></text>
                        <toTaxon></toTaxon>
                    </question>
                </couplet>

            <key>
                <keynotes></keynotes>

                </homotypes>
                <homotypes>

            <footnote id=""><footnoteString></footnoteString></footnote>

<distributionLocality class="region"></distributionLocality>
<distributionLocality class="continent"></distributionLocality>

<lifeCyclePeriods class="flowering and
fruiting"><dates><fullDate></fullDate></dates></lifeCyclePeriods>
<lifeCyclePeriods class="flowering and
fruiting"><dates><month></month></dates></lifeCyclePeriods>

<coordinates><latitude></latitude><longitude></longitude></coordinates>
```

<gatheringNotes></gatheringNotes>

<gathering><fieldNum></fieldNum><locality class="locality"></locality><dates><day></day><month></month><year></year></dates><subCollection></subCollection><collectionAndType></collectionAndType></gathering>;

<gathering><collector></collector><fieldNum></fieldNum></gathering>

<gathering><collector></collector><fieldNum></fieldNum><locality class="country"></locality></gathering>

<gathering><collector></collector><fieldNum></fieldNum><locality class="country"></locality><collectionAndType></collectionAndType></gathering>

<gathering><collector></collector><fieldNum></fieldNum><collectionAndType></collectionAndType><locality class=""></locality></gathering>

<gathering><collector></collector><fieldNum></fieldNum><collectionAndType></collectionAndType></gathering>

<gathering><locality class="region"></locality><collector></collector><fieldNum></fieldNum></gathering>

<gathering><collector></collector><fieldNum></fieldNum><locality class="locality"></locality></gathering>

<gathering><collector></collector><fieldNum></fieldNum><locality class="region"></locality></gathering>

<locality class="locality"><coordinates><latitude></latitude><longitude></longitude></coordinates></locality>

<locality class="locality"><coordinates><latitude estimate="true"></latitude><longitude estimate="true"></longitude></coordinates></locality>

<gathering><collector></collector><fieldNum></fieldNum><collectionAndType></collectionAndType></gathering>

<gathering><collector></collector><fieldNum></fieldNum><collectionAndType></collectionAndType><locality class="region"></locality></gathering>

<gathering><collectionTypeStatus></collectionTypeStatus><collector></collector><fieldNum></fieldNum><collectionAndType></collectionAndType><locality class="region"></locality></gathering>

```
<specimenType><gathering><collector></collector><fieldNum></fieldNum><locality
class="region"></locality><collectionAndType></collectionAndType></gathering></specimenType>

        <taxon>

            <nomenclature>

                <homotypes>

                    <nom class="accepted">

                        <name class="genus abbreviation"></name>

                        <name class="species"></name>

                        <name class="author"></name>

                    </nom>

                </homotypes>

            </nomenclature>

        </taxon>


        <taxon>

            <nomenclature>

                <homotypes>

                    <nom class="accepted">

                        <name class="genus"></name>

                        <name class="species"></name>

                        <name class="author"></name>

                    </nom>

                </homotypes>

            </nomenclature>

        </taxon>


            <nomenclature>
```

```
        <homotypes>
            <nom class="accepted">
            </nom>
        </homotypes>
</nomenclature>


            </nom>
        </homotypes>
        <homotypes>
            <nom class="accepted">


        <homotypes>
            <nom class="accepted">
            </nom>
        </homotypes>


        <homotypes>
            <nom class="accepted">
                <name class="genus"></name>
            </nom>
        </homotypes>


            <nameType>
                <nom class="nametype">
                    <name class="genus"></name>
                    <name class="species"></name>
                    <name class="author"></name>
```

```
            </nom>
        </nameType>
            <nom class="basionym">=
                <name class="genus"></name>
                <name class="species"></name>
                <name class="author"></name>
            </nom>

        <nom class="accepted">
            <name class="genus"></name>
            <name class="species"></name>
            <name class="author"></name>
            <citation class="publication">
                <refPart class="author"></refPart>
                <refPart class="pubname"></refPart>
                <refPart class="volume"></refPart>
                <refPart class="year"></refPart>
                <refPart class="pages"></refPart>
            </citation>
        </nom>

<figureRef ref=""></figureRef>

<figureRef ref="">Fig. <num></num></figureRef>

<figureRef ref="">Fig. <num></num><figurePart></figurePart></figureRef>
```

<figureRef ref="">Pl. <num></num>, <figurePart>fig. </figurePart></figureRef>

<figureRef ref="">Pl. <num></num>, <figurePart></figurePart></figureRef>

<figure id="" type="">Fig. <num></num>.
<figureLegend></figureLegend></figure>

<literatureRef ref="">

<refPart class="author"></refPart>

<refPart class="year"></refPart>

<refPart class="pages"></refPart>

</literatureRef>

<references><reference>

<refPart class="author"></refPart>

<refPart class="pubname"></refPart>

<refPart class="volume"></refPart>

<refPart class="year"></refPart>

<refPart class="pages"></refPart>

</reference></references>

<references><reference>

<refPart class="author"></refPart>

<refPart class="pubname"></refPart>

<refPart class="volume"></refPart>

<refPart class="year"></refPart>

<refPart class="pages"></refPart>

</reference></references>

```
<references><reference>
                                        <refPart class="url"></refPart>
                                    </reference></references>

            <taxon>
                <nomenclature>
                    <homotypes>
                        <nom class="accepted">
                            <name class="genus"></name>
                        </nom>
                    </homotypes>
                </nomenclature>
            </taxon>

            <taxon inGeoScope="false">
                <nomenclature>
                    <homotypes>
                        <nom class="accepted">
                            <name class="genus abbreviation"></name>
                            <name class="species"></name>
                        </nom>
                    </homotypes>
                </nomenclature>
                <feature class="distribution">
                    <string><distributionLocality
class="country"></distributionLocality></string>
                </feature>
```

```
</taxon>
```

Appendix III: Correcting more errors with Notepad++

As mentioned earlier in this document, Notepad++ can be used by advanced users to find and correct errors more easily. This is mostly because Notepad++ has rather powerful "Find"- and "Replace"-functions that can use Perl-like regular expressions.

A few examples, using "Find All" and "Replace All":

- Finding all cases where page numbers ended up between <refPart class="details"> tags is quite time-consuming in XML Spy, but they can be found much more quickly by using Notepad++' "Find"-function, entering **"details">\d** in the search box and setting the "Search Mode" to "Regular expression".
- Likewise, manually finding editor names that were not split off is very difficult. Entering <refPart class=".+">.+ in the search box with the option above enabled simplifies things a lot.
- Furthermore, putting additional refPart tags on a separate line is very easy in Notepad++. Assuming the additional refPart tags that had to be added manually are placed end to end, going to the "Replace"-window, setting the "Search Mode" to "Regular expression", and entering refPart><refPart in the "Find What:"-field and refPart>\n\t\t\t\t\t\t\t<refPart in the "Replace With:"-field will add a line break and the correct number of tabs in between tags.

It is worthwhile to play around with this.