Paratext Tips and Tricks

When send/receive will not work at all

See Help, send and receive is not working!

When Paratext will not run, or frequently crashes

See *Help! Paratext has stopped working*

When search does not work at all

A simple mistake in editing the language settings can make the search feature almost useless. When you enter lower case/upper case pairs in the Alphabetic characters section, the lower case character **MUST** come first. If your case pairs are in the wrong order in the language settings; searching for “a” will only find “A”, unless you select “match case”, in which case it will find “a” but not “A”. Searching for “sheep” will only find “SHEEP”.

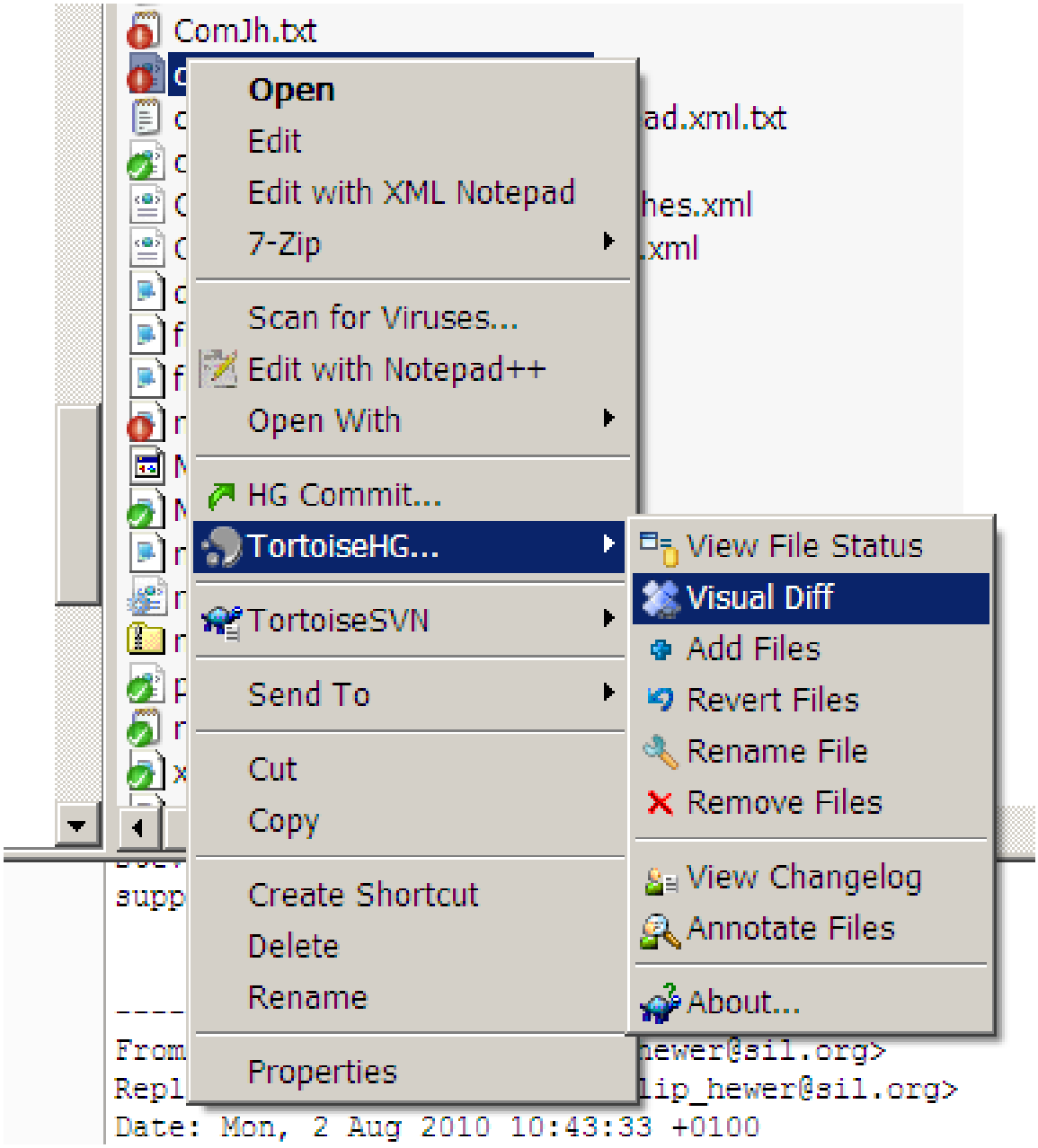
Getting your shared projects back after a computer disaster

If your Paratext projects folder became damaged or corrupted after a computer disaster, and files are missing from your project, **DO NOT** do a send/receive. The send/receive will first send your present damaged project to the repository, deleting or corrupting files for your colleagues. This can happen even if you do not have permissions to make changes in the project.What you should do is remove the project using Tools > Delete Entire Project Resource. Then you can do a send/receive, and you will receive a fresh copy of the project from the server, rather than distribute your corrupted version.

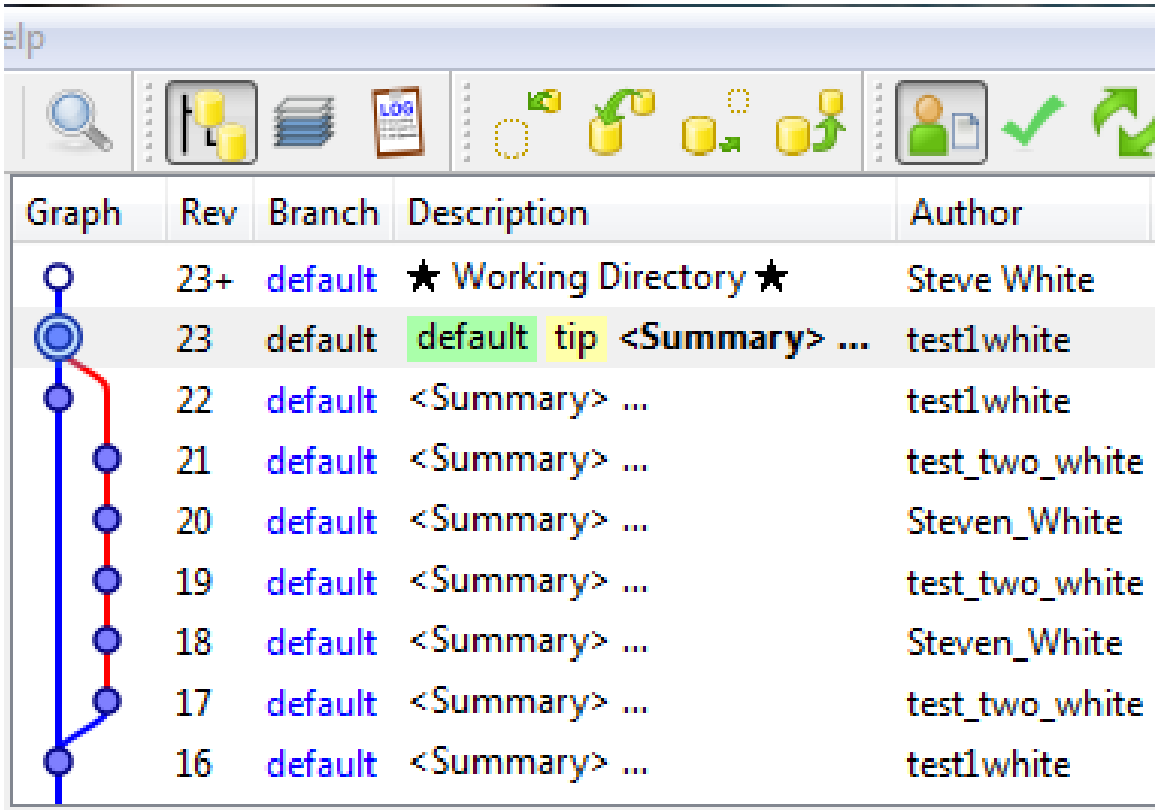
More details on changes to shared project files

The **View Project History** command shows you the history of the Scripture books in the project. But if you need to peruse the history of other files in the project, such as the consultant's notes, or the interlinear files, you can do this with a utility called **Tortoise HG**, which will show you the database Mercurial maintains on each file that is part of the project. With this utility you could do things like recover notes files that became corrupted or were deleted by some other program or mishap to your computer. For information or to download Tortoise HG, go to https://tortoisehg.bitbucket.io/. (The name HG relates to the Mercurial program that Paratext uses for managing the repositories and the send/receive process. HG is the chemical abbreviation for mercury).

When you install TortoiseHG, it adds several new options to Windows Explorer. If you go to **My Paratext Projects** and right click on your project folder, you'll probably see three commands in the middle of the context menu. TortoiseHG has a lot of commands that hopefully you won't need, it is intended for people to use for managing repositories directly. Paratext does a lot of this for you.

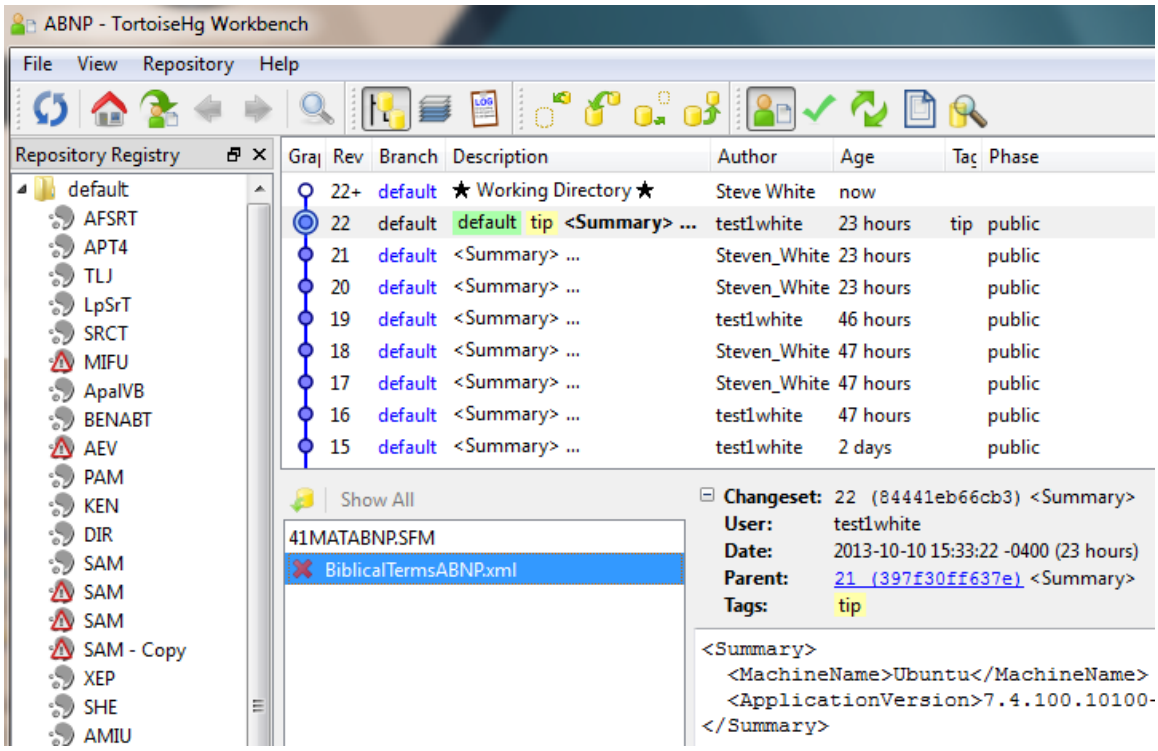


Perhaps the most useful one for seeing what is happening with your project files is **HG Workbench**. It will show you in a graphic form the history of changes in your project. Here is an example:



The red and blue lines show that recently changes from two users were merged with changes from a third user, which happened when the third user did not send and receive as frequently as the other two users. This does not mean a conflict necessarily occurred, if the users are assigned different books or different chapters to edit, there shouldn't be any conflicts even if one or more users have gone for a while without sending and receiving.

The HG workbench also permits you to recover a deleted file that does not appear in the Paratext project history. For example, in this project the BiblicalTerms file was deleted by mistake, and then send/receive deleted the file from other users' copies as well.



This shows at revision 22, the file BiblicalTermsABNP.xml was deleted. If this was a mistake, how can the file be recovered? TortoiseHG offers a way. If you right click on the version before the file was deleted (version 21 in this example) you can choose "Browse at revision". This will list all the files that were part

of the project at that revision. You can find the file you want to restore in the list, right click on it and choose "revert to revision." This will restore the file as it was.

To find when a file might have been deleted, you can press Ctrl-S, then in the *### revision set query ###* box which will appear, type **removes('*.*')** and Tortoise HG will show you only those history points when a file was deleted.

If you want to download Tortoise HG, you can download version 5.02 from this page. 5.02 is not the latest version, but it is the version compatible with the version of Mercurial that Paratext installs.

Merging lost notes with new notes

If a glitch deletes a user's note file (as described above) and that user has added some new notes, simply restoring the deleted notes file will remove the newest notes that the user added. You want to merge the notes file that was deleted with any newer notes. You can do this by copying notes out of one version of the file and pasting into another if you keep the XML formatting that Paratext is expecting.

Any notes file begins with the same two lines. The first one says this is an XML file. The second line says it is a comment list file.
~~>>< strike="">>>>>-?xml version="1.0" encoding="utf-8"?>~~

```
<?xml version="1.0" encoding="utf-8"?>
<CommentList>
```

A notes file ends with this line closing the comments list

```
</CommentList>
```

Each note begins with

```
<Comment Thread="56948783" User="Steven White" VerseRef="MAT 1:1-4" Language="" Date="2022-01
```

and ends with `</Comment>`

You can cut and paste notes if you preserve the necessary XML format markers: You need to have the two beginning lines at the beginning (and no where else in the file), the end line at the end (and no where else in the file) and each note

Step by step: merging two notes files

If File1 is the notes file that was deleted that you have recovered with TortoiseHG, (don't restore it to the project folder yet, restore it to another folder), and File2 is the current notes file with the newest notes, you can do this:

- close Paratext
- copy File2 from the project folder to the folder where File1 is
- open FIle1 in Notepad, delete the last line, (but leave the cursor at the beginning of a new line at the bottom of the file)
- open File2, delete the first two lines

- select all of File2, copy, and paste at the cursor at the end of File1
- save your new merged file with a new name, so you keep File1 and File2 unchanged in case of a mistake
- copy the merged file into the project folder, replacing the original of FIle 2, and renaming it notes_*user name*.xml
- now start Paratext

If successful, Paratext will open the project normally. If it detects an error in your merged file, it will say notes_*user name*.xml was corrupted and has been renamed notes_*user name*.corrupt. If this happens, review the list of XML markers above to find your mistake, close Paratext and try again.

If successful, you can mark a point in project history saying you have restored the missing notes, then send/receive to circulate the merged notes to the other team members.

Why is it important to close Paratext when introducing the new merged notes file to the project folder? Because Paratext when it opens the project loads the notes file into memory, and will not detect a change in that file while it is open, or when you close it, it might save its version from memory back to disk and overwrite what you just modified.

Autocorrect.txt tips

- The basic syntax: a change rule has --> (hyphen, hyphen, greater than) in the middle.

x–>z will change x to z.

- A line beginning with a # is a comment.

```
# change x to zx-->z
```

- You can specify characters by their Unicode number:

```
b-->\u0253
```

will change b to ɓ (Latin Small Letter B with Hook).

- Q: Is there a way to limit whether it matches whole words only?
- A: No. The example

```
teh-->the
```

in the Paratext help file is a bit dangerous, because it will change any sequence of “teh” in the middle or end of words. You can add spaces as part of your string (underscore used to show spaces here):

```
teh_-->the_
```

will not change “teh” in the middle of a word, but only at the end. And it would not change “teh” at the end of the word if the word was immediately followed by a punctuation mark. You can add a space at the end of your change string, so the space at the end of the word does not disappear.

- Q: Can you put - or > characters in your string to match?
- A: Yes. The program can find the –> sequence within a longer sequence of - or > characters. For example:

--->>

(three hyphens, two greater thans), changes hyphen into greater than.

- If Paratext is running when you change the autocorrect.txt file, you need to exit it and restart it for the changes in autocorrect.txt to work.

Older content

What is below this line may only apply to Paratext 7.

Settings and data files

Paratext lets you define what the Scripture book files will be called in your project, but it makes other files that you may not be aware of. Here is a list. Many of these names use the project short name, but not all. I'll abbreviate the project short name as PSN.Files that may be common to several projects These are in My Paratext Projects

- usfm.sty or usfm_draft.sty or various other .sty files. The style sheet. This informs Paratext how the markers are used, and the formatting to use for them in standard, or preview views, also when you save as RTF or print draft.
- .lds files. The language settings file. Stored in My Paratext Projects. The first part of the name is the name you give the language in Paratext.
- eng.vrs or org.vrs or othNN.vrs. Versification files.
- .ptwc files. (New to Paratext 7.4). These files, in My Paratext Projects\WindowsCollections, contain the info for saved text combinations.

Files unique to a project Usually inside the project folder, within My Paratext Projects

- PSN.ssf The project settings file. This stores info such as the name pattern for the book files, their location, whether the project is editable or not, and the number of books created. Located in My Paratext Projects, not inside the project folder.
- BiblicalTermsPSN.xml. The Biblical terms rendering data. In earlier versions of Paratext 7 and Paratext 6 this data was in a file named PSN.kb2. In 7.5, you may also find TermRenderingChanges.xml and ProjectBiblicalTerms.xml.
- ProjectUsers.xml. The list of users, roles and permissions for the project.
- unique.id. A long string of hex digits, I assume it is how Paratext can identify unique projects, even if they are shared on the Internet where there may be other projects with the same short name.
- .hg. The folder containing the repository data. Don't attempt to change anything in here, or you could corrupt your repository.

- gather A folder where Paratext keeps a copy of files the project needs that are not in the project folder, such as the style sheet, the language description and the .SSF file.
- Comments_User Name.xml. The file of notes made by the named user.
- Lexicon.xml Part of the interlinear information – it stores words and their analyses and glosses.
- Interlinear_language A folder containing the specific interlinear information for specific books. Each interlinearized book is represented by a file inside this folder. The language name is that of the model text used to interlinearize in.

The danger of file sync utilities

There are several utilities that will sync files automatically between two different computers. Programs like Dropbox or box.com will let you sync files to the Internet and potentially sync with other users you give permission to connect with you. It would be a bad idea to use one of these services to sync your Paratext project folder with other users. The problem is the Paratext project history repository is stored in a special file structure controlled by a program called Mercurial that Paratext runs. If Paratext and Mercurial need to write to the special repository files, and at the same time Dropbox is writing to these same files because your colleague is also working in Paratext and syncing to your project folder, the result easily could be a corrupt repository, resulting in a loss of your project history and the ability to do search and replace and other kinds of changes to your project. Update: Earlier I had suggested the possibility of using Dropbox to sync Paratext files by doing a send/receive to a folder inside Dropbox. It turns out this idea won't work, the way Dropbox works somehow doesn't match how Paratext works, because Paratext will think the Dropbox folder is read only, even though it isn't.

The list of available books

Paratext maintains inside the .SSF file for the project the list of books that have been created. The copy of the .SSF file stored inside the .gather folder doesn't contain this list. There have been a few cases reported where this list is not accurate. I've heard of a couple of shared projects where a book would disappear from Paratext's list when you shut it down and restarted it, then the book would reappear after a send/receive. And if a book file is added to the project folder by copying the file directly, Paratext won't detect that this book has been added, except as described below.To make Paratext refresh the list of existing books, go to Project > Project Properties and Settings, then click OK to exit the dialog (you don't need to change any settings, just open the dialog and close it with OK.) If the project is shared, and you are not an administrator, you can trigger the refreshing of the book list by going to Project > Language Settings and clicking OK.