Benthic Boundary Layer Project

Kilo Nalu Observatory

Real-time Streaming System

Subversion Guide

Sep 1, 2010

Christopher Jones
Dr. Margaret McManus
Department of Oceanography
University of Hawaii at Manoa

Table of Contents

1. Subversion Guide

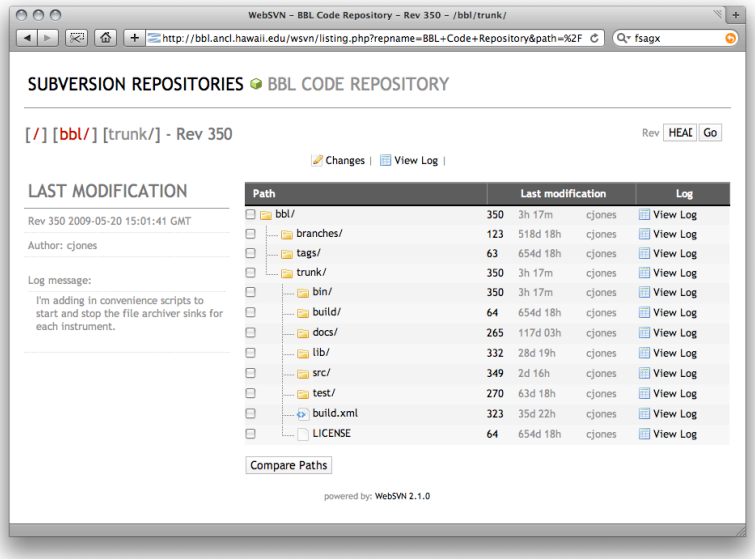
1.1. Overview

This document gives an overview of how to interact with the BBL software repository that is currently managed using the open source versioning system called Subversion. For an in-depth understanding of Subversion, see the Subversion guide at http://svnbook.red-bean.com/.

1.1.1. Working with the BBL Source Code Repository

The repository is a master copy of the source code, and is stored on the BBL server in the `/var/svn/bbl` directory. All changes to this directory are done using the various subversion commands or clients. The repository can be browsed by visiting http://bbl.ancl.hawaii.edu/wsvn. The code is open source, and downloadable by anyone, with the caveat that the copyright of the original authors be maintained, and that any modifications to the source code be made available to the public according to the GNU General Public License.

The Subversion system tracks all of the changes to the source code files, and stores the changes and each of the change log entries submitted by the person who made the particular change. The history of any given file can be viewed to get an understanding of who did what to a file, or to show when new files and features are added.



A screenshot of the BBL code repository web interface that allows browsing of the source code and version history of each file.

The entire source code repository can be downloaded to your local workstation using a Subversion client. On windows, a good program is TortoiseSVN (see http://tortoisesvn.tigris.org/). For the Mac, a good program is SCPlugin (see http://scplugin.tigris.org/), and for Linux, and good graphical client is eSVN (see http://esvn.sourceforge.net). The command line program (`svn`) is also available on Linux and Mac OS by default, and can be the quickest once the commands become familiar.

To make modifications to the source code, download the code (do a Subversion checkout, see below) using one of the clients above using the following repository URL: https://bbl.ancl.hawaii.edu/projects/bbl. We use a secure connection, and so you will have to accept the SSL certificate when the client connects and prompts you.

1.1.2. Modifying the BBL Source Code

Interacting with the Subversion server entails a hand full of operations for communicating with the server, committing changes to the master repository, updating your local copy of the repository, or getting information about selected files or directories. The operations are briefly described below, but look at the SVN Book (http://svnbook.red-bean.com/en/1.5/svn-book.pdf) for the definitive guide.

1.1.2.1. Subversion checkout

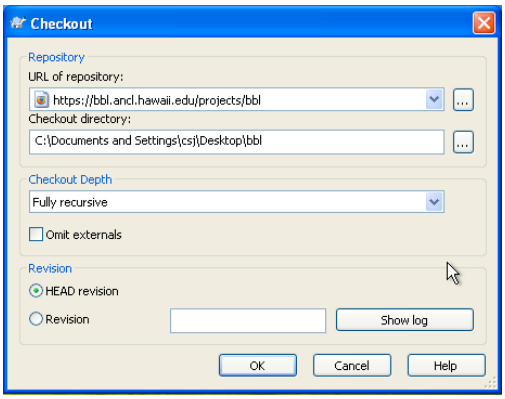
To download the source code from the repository, you perform an operation called *checkout*. If you are on Mac OS or Linux as your desktop, you can open a terminal, change to the desired work directory, and checkout the code using:

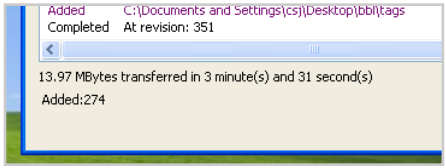
```
$ cd [checkout-directory]
$ svn checkout https://bbl.ancl.hawaii.edu/projects/bbl
```

You may be prompted to accept the secure SSL certificate from the server, and you should choose the 'permanently' option. To checkout the code base from a graphical client such as TortoiseSVN (on Windows), install TortoiseSVN, restart your PC, and then right-click on the desktop, and choose the *SVN Checkout ...* menu item. In the next dialog box, enter the following values

- URL of repository: `https://bbl.ancl.hawaii.edu/projects/bbl`
- Checkout directory: `C:\Documents and Settings\[user]\Desktop\bbl`
- Checkout depth: `Fully Recursive (dropdown)`
- Revision: `HEAD revision (radio button)`

When you click *Ok*, all of the files in the repository will be downloaded to your checkout directory, and are known as your 'working copy'. When the download is complete, you should see a final message that the checkout operation completed:



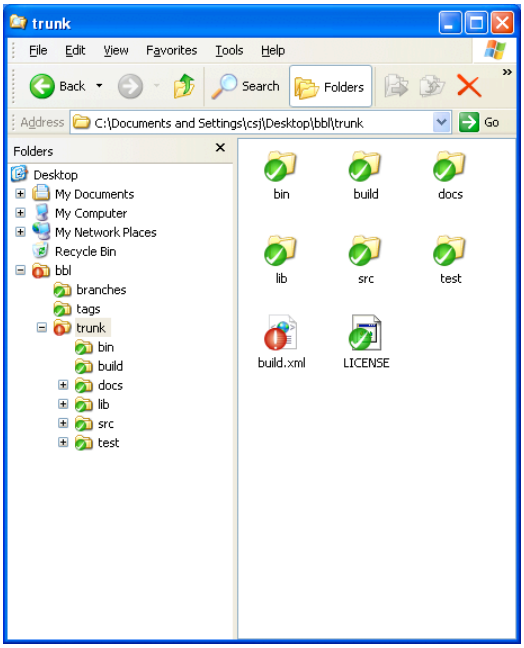


Now, change directories into the new *bbl* directory that was created in order to work with the files. You'll notice three top-level folders - *branches*, *tags*, and *trunk*. The branches folder is used for experimental development of the main source code tree, and the tags folder is used to create stable snapshots of the main code tree, but at the moment we are not using those directories. The main code base is in the trunk folder (i.e the trunk of the tree of code). These names are conventions that come from versioning systems like Subversion.

1.1.2.2. Subversion status

You can get the status of any file or folder by issuing the status command (Mac OS and Linux),

```
$ svn status
```



or in the graphical programs, the status of each file or folder is shown based on the changing icon (green means that the file has not changed from the master repository version, red means it has been altered in your working copy). Try opening the *build.xml* file with your text editor (WordPad, or try downloading the free code editor named ConTEXT from http://www.contexteditor.org/). Make a minor change to the file, save it, and notice the change in the status icon, or issue the `svn status` command. To revert the file back to the original, issue:

```
$ svn revert build.xml
```

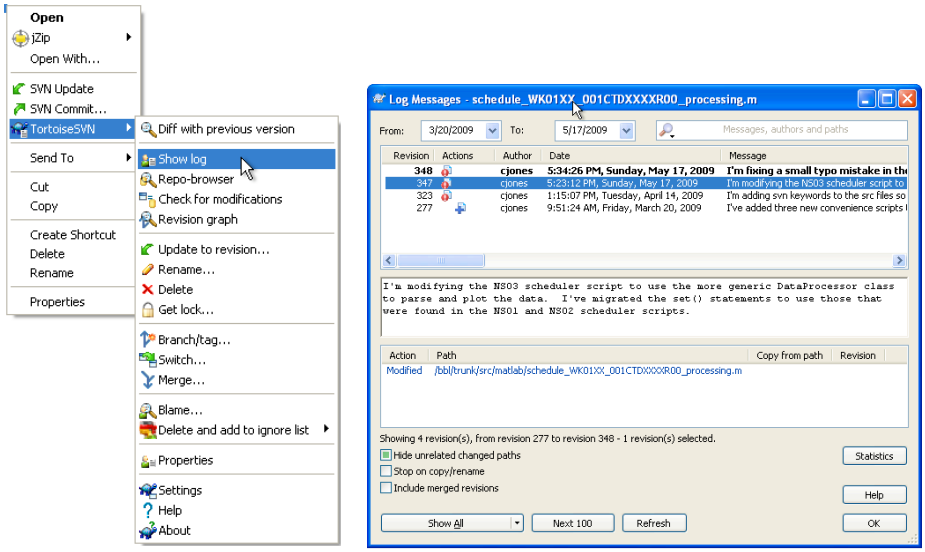
the `svn revert` command, or in Windows, right-click on the file and choose the *TortoiseSVN --> Revert ...* sub-menu item. This process is essentially the same using SCPlugin on the Mac (right click and choose *More --> Subversion --> Revert* menu item).

1.1.2.3. Subversion log

You can view the entire history of a given file by running the `svn log` command. For instance, change directories to the main Matlab source code directory in `trunk/src/matlab`, and type:

```
$ svn log schedule_WK01XX_001CTDXXXXR00_processing.m
```

The output of this command will show all of the log messages associated with this file, along with their revision numbers and revision dates. To do the same in Windows, navigate to the same directory, right-click on the *schedule_WK01XX_001CTDXXXXR00_processing.m* file, and choose the *TortoiseSVN --> Show log* menu item.



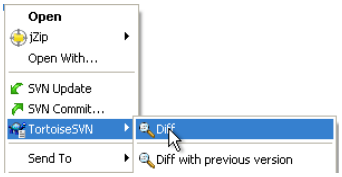
The resulting dialog box will allow you to explore the change history of the particular file. The graphical process is basically the same using SCPlugin on the Mac (right-click and choose the *More --> Subversion --> Log* menu item).

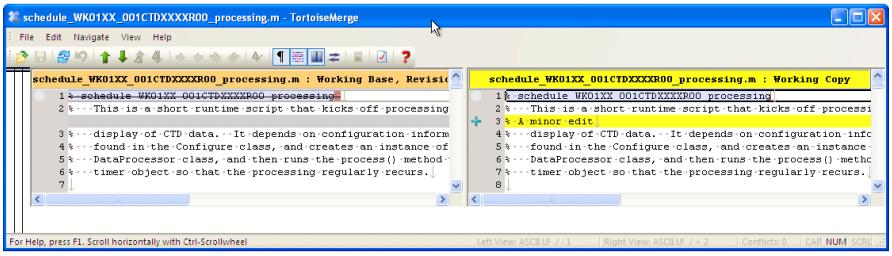
1.1.2.4. Subversion diff

To illustrate how to view the differences between a file in your working directory and the same file in the master repository, use your text editor (or Matlab) to change the *schedule_WK01XX_001CTDXXXXR00_processing.m* file. Then, if you're using the command line tools, issue the following command:

```
$ svn diff schedule_WK01XX_001CTDXXXXR00_processing.m
```

This will show the line-by-line differences of the file. Using TortoiseSVN, right click on the file and choose the *TortoiseSVN --> Diff* menu item. Likewise on the Mac with SCPlugin (*More --> Subversion --> Diff* menu item).





The dialog box will highlight line-by-line differences between the two versions (additions, deletions, etc.).

1.1.2.5. Subversion update

A subversion update operation merely updates your working copy (of a single file or whole folders) with the latest versions that have been committed to the master repository. When multiple people are working on the same code base, it's critical to keep your working copy up-to-date with the master versions. By doing so, you'll minimize conflicts and not overwrite someone elses changes. **Note: It's good practice to always run the Subversion update command to your local working copy of the code base before making any local changes and committing them to the master repository.** Subversion will flag conflicts, but it's good to avoid them in the first place. This is merely good etiquette in shared code development.

To update your working copy directory via the command line tools, issue the following commands:

```
$ cd [checkout-directory]/bbl/trunk
$ svn update
```

This will grab the latest versions of all of the files in the master repository, and will list which files have been added [A], which have been deleted [D], which have been updated [U], which have been merged with your local changes [G], and which are in conflict with your local working version [C]. If you see files that are conflicting, you'll need to reconcile the file differences manually in your file editor. To update in TortoiseSVN, right-click on the *[checkout directory]\bbl\trunk* folder and choose the *SVN Update* menu item. Using SCPlugin on the Mac, right-click and choose *More --> Subversion --> Update*.

1.1.2.6. Subversion commit

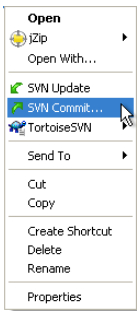
The Subversion commit operation is used to upload your changes from your local working copy to the master repository. This is often confused with Subversion update, but update pulls changes down to your machine, commit pushes changes up to the server. When you've finished modifying a source file to your liking, and it is stable (i.e. won't break the functionality of the overall code base), use the svn commit command along with a descriptive summary of the changes you've made. For instance, via the command line:

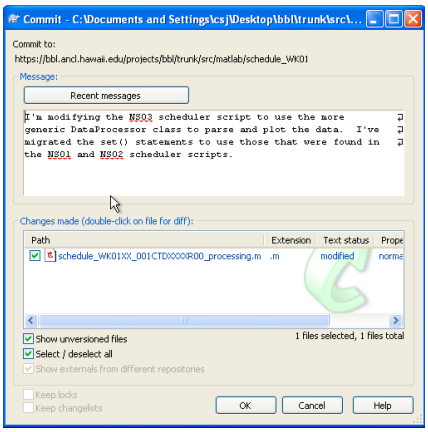
```
$ cd [checkout-directory]/bbl/trunk/src/matlab
$ svn commit schedule_WK01XX_001CTDXXXXR00_processing.m -m "I'm modifying
the NS03 scheduler script to use the more generic DataProcessor class to
parse and plot the data.  I've migrated the set() statements to use those
that were found in the NS01 and NS02 scheduler scripts."
```

Note: It's good practice to make changes, additions, or deletions to files in small, atomic, easy to understand steps. Try not to modify twenty unrelated files and commit them all at once, because it defeats the purpose of communicating the changes to others, and makes it difficult to revert the changes if something isn't correct. For instance, if you add a Matlab function, commit the function to the repository in a solitary commit with good documentation about its purpose, inputs, and outputs.

Note: Send an email to Chris (cjones@soest.hawaii.edu) to have your account added to the commit group for the BBL subversion repository. Read access is public, whereas write access is limited to Subversion accounts on the BBL machine.

To perform the same commit above using TortoiseSVN, right-click on the schedule_WK01XX_001CTDXXXXR00_processing.m file and choose the *SVN Commit ...* menu item. This will bring up the commit dialog box, and enter your detailed commit message to document the changes. This process is the same using SCPlugin on the Mac, just right-click and choose *More --> Subversion --> Commit* menu item.



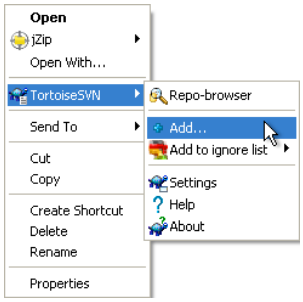


1.1.2.7. Subversion add

If you've created a new file that is part of the larger code base, use the Subversion add command to add it to your working copy. For instance, once you've finished writing a new `schedule_AW03XX_001CTDXXXXR00_processing.m` file, commit it to the repository with the following commands:

```
$ svn add schedule_WK02XX_001CTDXXXXR00_processing.m
$ svn commit schedule_WK02XX_001CTDXXXXR00_processing.m -m "I've added a new
scheduler script for the NS04 CTD located on Waikiki Beach.  The script
first creates a Configuration class, and calls the set() function on each of
the properties needed to configure the plotting for the instrument data
stream.  It then calls the DataProcessor class with the configuration
variable as it's input.  Finally, it creates a Matlab timer, and calls the
DataProcessor.process() function on a twenty minute schedule to process and
plot the newest data from the Data Turbine."
```

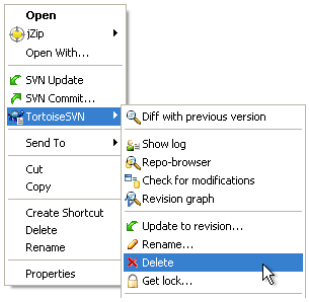
Again, the same can be done using TortoiseSVN. Right-click on the new file and select the *TortoiseSVN --> Add ...* menu item. The file icon should change to have a blue plus in front of it indicating that it has been added to your local working copy, but needs to be committed to the master repository. Right-click on the file again and choose the *SVN Commit ...* menu item to commit the changes, and add the commit message to the dialog box. The same process will work using SCPlugin on the Mac. Right-click on the file and choose *More --> Subversion --> Add*, and then, again, *More --> Subversion --> Commit*.



1.1.2.8. Subversion remove

Lastly, if you've added a file to the repository and decided that it was a mistake, or if a file is no longer pertinent to the code base, remove it using the Subversion remove command. For instance, in the case of the scheduler script above:

```
$ svn remove schedule_WK02XX_001CTDXXXXR00_processing.m
$ svn commit -m "I've mistakenly added this file and am
removing it."
```



Using TortoiseSVN, right-click on the file and choose *TortoiseSVN --> Delete*, and then right-click on the containing folder and choose the *SVN Commit ...* menu item. For SCPlugin on the Mac, right-click on the file, choose *More --> Subversion --> Remove*, and then *More --> Subversion --> Commit*.

1.1.3. Installing the BBL Source Code

On each of the servers, the source code should already be installed in `/usr/local/bbl/trunk`. In the event that the code needs updating, change to this directory on each of the servers, and update the code with the following commands as the *kilonalu* user:

```
$ cd /usr/local/bbl/trunk
$ svn update
```

In the event that there are changes to the Java code that need to be installed, additionally use the following command:

```
$ ant clean compile
```

This will remove the old code and compile and install a new version of the Java code.