Published on *ROOT* (http://root.cern.ch/drupal)

Subversion HOWTO

Table of Contents

Checking Out and Updating

How do I check out the ROOT Subversion repository?

Note: the repository at *https://root.cern.ch/svn/root/* is only for ROOT developers. Non-developers should use *http://root.cern.ch/svn/root/* instead. For browsing the repository with a web browser, use http://root.cern.ch/viewcvs [1].

To check out the ROOT repository, you do the following:

```
svn co https://root.cern.ch/svn/root/trunk [2] root
```

That will check out the whole ROOT trunk (i.e. the *head* in CVS language) and store it in local directory `root`.

Note: watch out not to checkout *https://root.cern.ch/svn/root* as that will checkout the trunk as well as all branches and tags.

How do I check out a specific tag?

To check out a specific tag or branch, you do:

```
svn co https://root.cern.ch/svn/root/tags/v5-14-00h [3] root-51400h
```

This will check out the v5-14-00h release and store it in local directory `root-51400h`.

To find out which tags and branches are available you can browse the repository via the web interface http://root.cern.ch/viewcvs [1] or using:

```
svn ls https://root.cern.ch/svn/root/tags
```

I have checked out starting at trunk. How can I check out branches without downloading everything again?

You can check out the other directories somewhere else in your system, even in /tmp. Subversion won't have a problem doing copies from different WCs, as long as they live in the same repository.

If you did not check out the toplevel dir but want to do it now, you can try the following trick:

```
svn co -N https://root.cern.ch/svn/root [4]
cd root
svn up -N trunk branches tags
rm -rf trunk
mv /path/to/older/checkout/of/trunk .
```

How do I switch a subdirectory to a (different) branch?

Often you only want to work on a subdirectory of ROOT that is kept in a development branch (say gui/), while all other directories should be from the trunk or a different branch. You can achieve this by checking out the trunk, then cd'ing into gui/ and switching it to the branch using

```
svn switch https://root.cern.ch/svn/root/branches/dev/gui/android
```

How do I update my checked out version?

To update your working dir, all you have to do is run:

```
svn up
```

from inside it, just as you did with CVS.

Now that I have the source, how do I build ROOT?

Have a look at the installation [5] guide.

Editing Files and Committing

Editing files

Editing files with Subversion is no different than doing so with CVS.

To see which files you've edited do:

```
svn st
```

This is a very fast operation as it does not contact the remote repository (and does not bring the repository to the HEAD level).

Committing changes

The only difference you should know about is that Subversion does atomic commits, and you are encouraged to make your commits complete.

In other words, suppose you add a new file to Subversion and you update Module.mk. You should first svn add the new file, and then commit both it and Module.mk, in one go. That way, anyone doing a checkout cannot possibly get a revision

in which one change had happened but not the other. To commit just do:

```
svn ci
```

Renaming and copying files and directories

One of the advantages of Subversion over CVS is that you can rename and copy files and directories without loosing the history. Where in CVS you would do:

```
cvs rm oldname.cxx
cvs add newname.cxx
cvs ci oldname.cxx newname.cxx
```

You now simple do:

```
svn mv oldname.cxx newname.cxx
svn ci newname.cxx
```

Reverting changes

If you want to undo some changes you've made but not yet committed, try this command:

```
svn revert <file>
```

If you did commit your changes, you can do this:

```
svn merge -r COMMITTED:PREV <file>
```

(type COMMITTED and PREV, in capitals, as shown).

If you have to revert an entire commit made at say revision *xyz* do this:

```
svn merge -c -xyz .
```

Correcting commit log messages

If for some reason when you commit a change you don't put the proper (or complete) information into the commit log message, you can correct it. Get the revision number of the commit with the incorrect message (for example, 9915), and then:

```
svn propedit svn:log --revprop -r 9915 https://root.cern.ch/svn/root
```

This will open an editor window (using $EDITOR) and let you correct your mistakes, and then commit the change. Keep in mind that these properties are unversioned, meaning once you change it, the original version is gone.

Resolving conflicts

It might happen that an svn up will update a file you just edited and that the conflicts could not be automatically resolved. In that case the file will contain conflict markers "<<<<<<<" and ">>>>>>>" and you have to edit the file to remove the conflict. Before being able to commit this file you first have to tell svn that the conflict has been resolved:

```
svn resolved myfile.cxx
```

Only then can you proceed to commit the file.

Branching and Tagging

What's the difference between branches and tags in Subversion?

There are no differences between branches and tags in Subversion. In fact, Subversion doesn't even know the concept of branches or tags: everything is a file or a directory for it.

A branch or a tag is nothing more than a copy of your files under a different path. It is an O(1) operation in time and disk space, so there's no harm copying everything.

How does the ROOT repository do branches and tags?

ROOT uses the following scheme for its branches and tags:

- /branches: contains all official branches. They are created after a release and are mainly for backporting bugfixes

from the trunk.
- /branches/dev: contains development, temporary branches. This is where unstable features are developed before they are merged into /trunk.
- /tags: contains all official tags. That is, when a public release is make a tag here to mark the revision in which it happened.

In /branches and /tags, the naming convention used is like this:

- /branches/branchversion, (example /branches/v5-14-00-patches)
- /tags/releaseversion (example: /tags/v5-14-00)

/branches/dev has no naming convention, but we ask you to give your branches meaningful names. Names like "my-cool-branch" aren't very descriptive, whereas "new-schema-evolution" is. Please remember to erase your work branches after you're done.

So, how do I create a branch or copy of my work?

In order to create a branch or a tag, you must have /branches, /branches/dev or /tags checked out, even if not recursing (-N).

If your dev's directory doesn't exist yet, create it:

```
svn mkdir branches/dev/mybranch
```

Now copy the trunk:

```
svn cp trunk branches/dev/mybranch
```

Of course, this will include the full ROOT, but there's no harm in doing so.

Creating the branch directly on the server

You can also create the branch directly without having a checked-out working copy:

```
svn cp https://root.cern.ch/svn/root/trunk [2] \
       https://root.cern.ch/svn/root/branches/dev/mybranch
```

The branch is now a identical copy of trunk, and you can check out mybranch. When you are done, you can merge the changes back to trunk. If your work takes a long time, you can easily merge changes from trunk into your working branch to stay in sync with the current development on trunk. This will also make merging your changes back to trunk much easier, as the only differences between trunk and your branch are the actual changes you have done to the branch.

You can read more about branching and merging in the book "Version Control with Subversion" in chapter four at http://svnbook.red-bean.com/en/1.4/svn.branchmerge.html [6].

I'm the Release Dude. How do I make the next ROOT release?

```
svn cp https://root.cern.ch/svn/root/trunk [2] \
       https://root.cern.ch/svn/root/tags/v5-18-00 [7]
svn cp https://root.cern.ch/svn/root/tags/v5-18-00 [7] \
       https://root.cern.ch/svn/root/branches/v5-18-00-patches
```

It's done.

Working with Branches

Switching

I currently have /trunk checked out. How do I switch to the /branches/v5-14-00-patches branch?

The switch subcommand can be used for that:

```
cd trunk
svn switch https://root.cern.ch/svn/root/branches/v5-14-00-patches
```

More on the switch [8] man page.

Getting information about the working directory

Where am I? I am about to check in my local files - will they end up on my branch or destroy the trunk?

You can ask your current directory what part of the subversion repository it corresponds to. `svn info` will tell you the directory's URL, the current revision, and when the last change occurred within the current directory:

```
$ svn info
Path: .
URL: https://root.cern.ch/svn/root/branches/dev/axel/cintexternaltest/cint [9]
Repository UUID: 27541ba8-7e3a-0410-8455-c3a389f83636
Revision: 19995
Node Kind: directory
Schedule: normal
Last Changed Author: rdm
Last Changed Rev: 19824
Last Changed Date: 2007-09-19 21:46:41 +0200 (Wed, 19 Sep 2007)
Properties Last Updated: 2007-09-21 17:42:53 +0200 (Fri, 21 Sep 2007)
```

Synchronizing a branch with the trunk

How do I merge the latest patches added to the trunk in my development branch?

Suppose that you want to synchronize your development branch *branches/dev/mydevs* with the trunk. To see what files have been changed compared to your branch do:

```
$ svn st -u
       *    20177   gui/src/TGListTree.cxx
       *    20177   gui/inc/TGListTree.h
       *    20177   gui/inc/TGView.h
M      *    20177   io/src/TFile.cxx
       *    20177   meta/src/TStreamerElement.cxx
       *            tutorials/gui/iconAsXPMData.C
       *    20177   tutorials/gui
```

You can see the detailed differences in each modified file using:

```
$ svn diff -r BASE:HEAD io/src/TFile.cxx
Index: TFile.cxx
===================================================================
--- TFile.cxx   (revision 20177)
+++ TFile.cxx   (revision 20184)
@@ -3120,7 +3120,7 @@
          delete u;
          if (read || sameUser) {
             localFile = kTRUE;
-            if (localFile) {
+            if (localFile && prefix) {
                *prefix = lfname;
             }
          }
```

The next step is to merge the differences in your local working directory:

```
$ svn merge -r BASE:HEAD .
U    gui/src/TGListTree.cxx
U    gui/inc/TGListTree.h
U    meta/src/TStreamerElement.cxx
A    tutorials/gui/iconAsXPMData.C
U    io/src/TBufferFile.cxx
C    io/src/TFile.cxx
```

If not sure, you can first run with the option '--dry-run' to check what is going to happen.

The 'C' in the output indicates that there was a conflict while merging io/src/TFile.cxx. After resolving the conflicts by editing the file (and removing the conflict markers), you have to tell svn that the conflicts have been resolved:

```
$ svn resolved io/src/TFile.cxx
```

Now you can commit the changes to your development branch:

```
$ svn ci -m "Synchronize with the head"
Sending        gui/inc/TGListTree.h
Sending        gui/src/TGListTree.cxx
Sending        io/src/TBufferFile.cxx
Sending        io/src/TFile.cxx
Sending        meta/src/TStreamerElement.cxx
Sending        tutorials/gui/iconAsXPMData.C
Transmitting file data ....
Committed revision 20185.
```

And your development branch is now in sync with the head.

Merging from trunk to branch

How do I merge a fix from the trunk to a patch branch?

Suppose that you want to merge a fix you just made in the trunk at revision 1002 to a patch branch and suppose your patch branch is *~/root-v5-30-patches* and your trunk is *~/root*, then do:

```
$ cd root-v5-30-patches
$ svn merge -c 1002 ../root .
```

If not sure, you can first run with the option '--dry-run' to check what is going to happen.

And revision 1002 is now merged in the patch branch.

Merging a branch into the trunk

After developing on a dev branch for a while, how do I merge my work back into the trunk?

Check out a copy of the trunk:

```
svn co https://root.cern.ch/svn/root/trunk [2] root
```

Look up the revision at which you cut your dev branch:

```
svn log --stop-on-copy https://root.cern.ch/svn/root/branches/dev/mybranch [10]
```

This will display back to you the changes that have been made back to the point the branch was cut. Remember that number (should be rXXXX, where XXXX is the revision number).

Change your current working directory to the trunk and perform an update:

```
cd root
svn up
```

This will update your copy of trunk to the most recent version, and tell you the revision you are at. Make note of that number as well (should say “At revision YYYY” where YYYY is the second number you need to remember).

Now we can perform the merge:

```
svn merge -rXXXX:YYYY https://root.cern.ch/svn/root/branches/dev/mybranch [10]
```

This will put all updates into your current working directory for trunk.

During merging resolve any conflicts.

Check in the results:

```
svn ci -m "MERGE mybranch [XXXX]:[YYYY] into trunk"
```

That is it. You have now merged "mybranch" with the trunk.

Displaying differences

How can I check what are the differences between my working copy of a file and revision 20474?

This is simply obtained using `svn diff`:

```
$ svn diff -r 20474 xrootd/Module.mk
Index: xrootd/Module.mk
===================================================================
--- xrootd/Module.mk    (.../v5-14-00-patches/xrootd/Module.mk) (revision 20474)
+++ xrootd/Module.mk    (.../dev/v5-14-00-newxrd/xrootd/Module.mk) (working copy)
@@ -6,7 +6,7 @@
MODDIR     := xrootd
MODDIRS    := $(MODDIR)/src

-XROOTDVERS := xrootd-20060928-1600
+XROOTDVERS := xrootd-20071001-0000
XROOTDDIR  := $(MODDIR)
XROOTDDIRS := $(MODDIRS)
XROOTDDIRD := $(MODDIRS)/xrootd
```

You can also get a more elaborated 'diff' result by running your preferred 'diff' command, for example, to get a context diff, you can use the system 'diff' and option '-c' in the following way:

```
$ svn diff -r 20474 --diff-cmd diff -x -c xrootd/Module.mk
Index: xrootd/Module.mk
===================================================================
*** xrootd/Module.mk    (.../v5-14-00-patches/xrootd/Module.mk) (revision 20474)
--- xrootd/Module.mk    (.../dev/v5-14-00-newxrd/xrootd/Module.mk) (working copy)
***************
*** 6,12 ****
MODDIR     := xrootd
MODDIRS    := $(MODDIR)/src

! XROOTDVERS := xrootd-20060928-1600
XROOTDDIR  := $(MODDIR)
XROOTDDIRS := $(MODDIRS)
XROOTDDIRD := $(MODDIRS)/xrootd
--- 6,12 ----
MODDIR     := xrootd
MODDIRS    := $(MODDIR)/src

! XROOTDVERS := xrootd-20071001-0000
XROOTDDIR  := $(MODDIR)
XROOTDDIRS := $(MODDIRS)
XROOTDDIRD := $(MODDIRS)/xrootd
```

Vendor branches

Read here how we manage vendor branches [11] containing third party source code.

Miscellaneous

What's this PREV, HEAD, etc. stuff?

Those are symbolic revision names for Subversion, just like the normal numeric ones. They mean the following:

- HEAD: latest (youngest) revision in the server
- BASE: the revision your checkout was last updated against
- COMMITTED: last revision a file or directory was changed
- PREV: the last revision the file or directory was changed immediately before COMMITTED

Maybe this is better explained with an example:

1. You check out trunk at revision 200
2. You make a change to trunk/a_file and commit it: revision 201 is created
3. A day later, you update your working dir and find out it's now revision 208
4. You make another modification and commit: revision 209
5. One day later, you update again, this time to revision 212
6. The following day, before updating, the server has progressed to revision 218

Under those circumstances, here's what each one of those 4 mean:

- HEAD = 218
- BASE = 212
- COMMITTED (for a_file) = 209
- PREV (for a_file) = 201

When you run "svn up", you bring BASE up to HEAD.

How to avoid seeing generated files?

To avoid seeing the generated files, like the dictionary G__* or dependency *.d files, when doing `svn st` you have to add

```
G__*.cxx G__*.c G__*.h *.so *.dylib *.dll *.lib *.pdb *.obj *.def
*.exp *.ilk *.manifest *.d *.rootmap *.pbxuser *.perspective* *.mode*
```

to the `global-ignores` in your `~/.subversion/config` file. You can at any time see all files doing `svn st --no-ignore`.

How to set automatically basic properties on new files?

Subversion keeps for each file and directory a hidden file with property information. Using properties Subversion keeps

track, for example, of the mime type of a file (.jpg, .png) or if a file is an executable (.sh) or which keywords should be expanded (e.g. Id), etc. To make sure that new files get a correct set of default properties, you have to set in your `~/.subversion/config` file:

```
enable-auto-props = yes
[auto-props]
*.c = svn:eol-style=LF;svn:keywords=Id
*.C = svn:eol-style=LF;svn:keywords=Id
*.cxx = svn:eol-style=LF;svn:keywords=Id
*.cpp = svn:eol-style=LF;svn:keywords=Id
*.cc = svn:eol-style=LF;svn:keywords=Id
*.h = svn:eol-style=LF;svn:keywords=Id
*.hh = svn:eol-style=LF;svn:keywords=Id
*.m = svn:eol-style=LF;svn:keywords=Id
*.mm = svn:eol-style=LF;svn:keywords=Id
*.f = svn:eol-style=LF;svn:keywords=Id
*.F = svn:eol-style=LF;svn:keywords=Id
*.inc = svn:eol-style=LF;svn:keywords=Id
*.dsp = svn:eol-style=CRLF
*.dsw = svn:eol-style=CRLF
*.sh = svn:eol-style=LF;svn:executable;svn:keywords=Id
*.py = svn:eol-style=LF;svn:executable;svn:keywords=Id
*.pl = svn:eol-style=LF;svn:executable;svn:keywords=Id
*.txt = svn:eol-style=LF;svn:keywords=Id
*.png = svn:mime-type=image/png
*.jpg = svn:mime-type=image/jpeg
Makefile = svn:eol-style=LF;svn:keywords=Id
Makefile.* = svn:eol-style=LF;svn:keywords=Id
*.mk = svn:eol-style=LF;svn:keywords=Id
*.cmake = svn:eol-style=LF;svn:keywords=Id
```

For more on properties and how to change, list and delete them see http://svnbook.red-bean.com/en/1.4/svn.advanced.props.html [12].

How do I track changes?

There are three convenient ways to track changes:

- http://root.cern.ch/root/rootsvn.rss [13], real-time RSS feed of last 20 check-ins
- http://root.cern.ch/viewcvs/ [14], real-time browsing of revisions
- http://root.cern.ch/root/ChangeLog.phtml [15], concise summary of all logs, updated every 6 hours

Where to get the `svn` client for my machine?

Subversion is by default installed on most recent Linux distributions. On Mac OS X 10.4 you can get it from Fink or from a .dmg containing pre-built binaries. On Mac OS X 10.5 svn is part of the system. On Windows you can get it from cygwin. For all other platforms see http://subversion.apache.org/packages.html [16].

Where to find more on Subversion?

The definite information can be found on the Subversion project page http://subversion.apache.org/ [17] and in the online Subversion book http://svnbook.red-bean.com/ [18].

Source URL: http://root.cern.ch/drupal/content/subversion-howto

Links:
[1] http://root.cern.ch/viewcvs
[2] https://root.cern.ch/svn/root/trunk
[3] https://root.cern.ch/svn/root/tags/v5-14-00h
[4] https://root.cern.ch/svn/root
[5] http://root.cern.ch/drupal/installing-root-source
[6] http://svnbook.red-bean.com/en/1.4/svn.branchmerge.html
[7] https://root.cern.ch/svn/root/tags/v5-18-00
[8] http://svnbook.red-bean.com/en/1.4/svn.ref.svn.c.switch.html
[9] https://root.cern.ch/svn/root/branches/dev/axel/cintexternaltest/cint
[10] https://root.cern.ch/svn/root/branches/dev/mybranch
[11] http://root.cern.ch/drupal/how-we-manage-vendor-branches
[12] http://svnbook.red-bean.com/en/1.4/svn.advanced.props.html
[13] http://root.cern.ch/root/rootsvn.rss
[14] http://root.cern.ch/viewcvs/
[15] http://root.cern.ch/root/ChangeLog.phtml
[16] http://subversion.apache.org/packages.html

[17] http://subversion.apache.org/
[18] http://svnbook.red-bean.com/