

Getting Started with **SubVersion** (Migrating from CVS to SVN)

Version 1.0

Frank C. Anderson

Department of Mechanical Engineering, Stanford University
fca@stanford.edu

SimTK.org uses SVN (**SubVersion**, <http://subversion.tigris.org>) as its source-code versioning system. It stores a revision history for files under version control in a repository and enables potentially large numbers of programmers to work on source code concurrently. It is intended to replace CVS (Concurrent Versions System, <http://www.cvshome.org>). Some of the key improvements over CVS include:

- Repository-wide version numbering. Each time a change is committed to the SVN repository, the revision number of the entire repository is incremented. This is in contrast to CVS, which keeps revisions on a per-file basis. The versioning system used by SVN allows for easier retrieval of self-consistent snapshots of the code.
- Directories and renames are versioned. This allows one to change the names of files and directories and reorganize a repository, something that is difficult to do in CVS.
- Efficient versioning of binary files. All diffs in SVN are on a binary basis, thus text files and binary files are handled the same.
- Branching and Tagging are inexpensive. Branching is the mechanism for making major code revisions without breaking the main development trunk. Tagging is the mechanism for making release snapshots. Both mechanisms use the copy command. In SVN, copies are done through symbolic links, so the disk space that is needed only for changes that are made in the files.

Using SVN is much like using CVS; many of the commands are the same. A user's guide can be found at <http://svnbook.red-bean.com/>. Several client implementations of SVN are available. For example, Cygwin comes with a command-line version of SVN (<http://www.cygwin.com/>), and TortoiseSVN is a graphical implementation that integrates nicely into Microsoft Windows (<http://tortoisesvn.tigris.org/download.html>).

Migrating from CVS to SVN

If you have an existing CVS source code repository, there are two basic approaches for migrating your repository to SVN. The first is straightforward. One simply adds the most recent source code files maintained in your CVS repository (i.e., the head revision) to the SVN repository. This approach unfortunately does not preserve the revision history that is maintained in the CVS repository. The second, more-involved approach is to convert your CVS repository into an SVN repository, which does preserve the revision history. This section describes this second approach by providing a specific example.

The basic procedure is to use a utility called **cvs2svn** (e.g., <http://cvs2svn.tigris.org/>) to convert your CVS repository into an SVN dump file and then load the dump file into the repository for your SimTK.org project. The procedure outlined below assumes that you are able to logon to the SimTK.org server where your project repository resides. Alternatively, you may follow the procedure

on any computer where the cvs2svn, svn, and svnadmin commands are available, and then provide the final dump file to the webmaster (webmaster@simtk.org) for loading into your project repository.

The following example shows how two existing CVS repositories, RDI and SU, were converted and loaded into the nmlbtk repository maintained by SimTK.org.

Step 1. Create a local temporary repository

```
$ svnadmin create /home/fca/nmlbtk
```

Step 2. Convert CVS repositories to SVN dump files

```
$ cvs2svn --trunk-only --dump-only --dumpfile=rdi_dump.svn RDI
$ cvs2svn --trunk-only --dump-only --dumpfile=su_dump.svn SU
```

Step 3. Exclude (or include) specific trees in the dump files

```
$ cat rdi_dump.svn | svndumpfilter exclude trunk/Ncompass trunk/RD/CMC trunk/RD/Maya
trunk/RD/Simulation/Mimic --drop-empty-revs --renumber-revs > rdi_dump_excluded.svn
```

```
$ cat su_dump.svn | svndumpfilter exclude trunk/SU/TrackController --drop-empty-revs --
renumber-revs > su_dump_excluded.svn
```

Step 4. Make directories into which to load the dump file

```
$ svn mkdir -M "Point for loading existing svn repository dump files."
file:///home/fca/nmlbtk/ZLoadPoint
```

```
$ svn mkdir -M "Point for loading Realistic Dynamics code."
file:///home/fca/nmlbtk/ZLoadPoint/RealisticDynamics
```

```
$ svn mkdir -M "Point for loading NMBL code." file:///home/fca/nmlbtk/ZLoadPoint/NMBL
```

Step 5. Load the dump at a desired load point

```
$ svnadmin load --parent-dir ZLoadPoint/RealisticDynamics /home/fca/nmlbtk <
rdi_dump_excluded.svn
```

```
$ svnadmin load --parent-dir ZLoadPoint/NMBL /home/fca/nmlbtk < su_dump_excluded.svn
```

Step 6: Structure the repository

```
$ svn mkdir -m "Main trunk for code development" file:///home/fca/nmlbtk/trunk
```

```
$ svn mkdir -m "Directory for branch development" file:///home/fca/nmlbtk/branches
```

```
$ svn mkdir -m "Directory for trunk snapshots, such as releases. Changes should not be
made to code in this directory." file:///home/fca/nmlbtk/tags
```

```
$svn mv file:///home/fca/nmlbtk/ZloadPoint/NMBL/Analyses file:///home/fca/nmlbtk/trunk
```

```
$ ... (Continue moving files and directories to the Trunk in the structure you wish)
```

Step 7: Dump the local SVN repository

```
$ svnadmin dump /home/fca/nmlbtk > nmlb_dump.svn
```

Step 8: Load the unified dump file into the actual SimTK.org repository

```
$ svnadmin load /var/gforge/svn/nmlbtk < nmlb_dump.svn
```

--- or, provide your dump file to the webmaster (webmaster@simtk.org) for loading.

Some useful commands:

```
replace "from string" "to string" -- file
svnlook tree /home/fca/Repositories/nmlbtk
```