# Updating SNePS Using CVS
# SNeRG Technical Note 32

Frances L. Johnson
Department of Computer Science and Engineering
and Center for Multisource Information Fusion
and Center for Cognitive Science
University at Buffalo, The State University of New York
201 Bell Hall
Box 602000
Buffalo, NY 14260-2000
Phone: 716-645-3180
flj@cse.buffalo.edu

September 30, 2002

## 1 Introduction

### 1.1 The purpose of this report

This report describes how to make a new SNePS release (or just alter SNePS source code) using CVS in the system set up for altering SNePS in the Computer Science and Engineering Department at University at Buffalo, The State University of New York (UB). These instructions are directed solely to the person who is altering/updating the *official* department SNePS code — referred to as *snwiz* (the SNePS wizard) — and are written by Frances Johnson, the current snwiz, as she learned how to use CVS.

There is an online CVS manual located at: `http://www.loria.fr/~molli/cvs/doc/cvs_toc.html`. **Snwiz should not start making any changes before reading this document** — at least through Section 3.

For those who are making code alterations for their own research, these instructions do not apply. These people should create new code in their own directories. Any code changes that are selected for addition to the official SNePS release are to be given to snwiz who will make the official changes following the instructions below.

**Note:** Anyone making changes to sneps code should read the warning statement in Section 3.2.3.

### 1.2 Reasons for making changes to the official SNePS release

#### 1.2.1 The importance of using CVS

CVS maintains a record of any changes to the SNePS source code. Any changes must be made using the techniques described in this report — otherwise, they might not be retained when the next alteration to the code is made. Below are a few basic reasons why snwiz might need to alter/update the SNePS source code:

- Code alterations — due to changes in the underlying theory
- Code fixes — improvements in usability or bug repair
- Updating for a new ACL release — updating the files to run on a new LISP release

### 1.2.2 Code alterations

Students alter SNePS code in their own directories to test their theories for alternative Knowledge Representation techniques. Theories that improve the system can be added to SNePS once the SNePS Research Group (SNeRG) agrees on *how* they should be added. The altered version is typically designated as a new release with a new version number, and the magnitude of the difference between this and the previous version number reflects the magnitude of the code alteration.

### 1.2.3 Code fixes

Code fixes improve the speed of the system or remove bugs/errors in the *implementation* of the theories. These changes do not indicate a change in the theory and do not result in a new release. They often go relatively unnoticed by most of the users — with the exception of bug repairs which are noticed and appreciated by those who reported them.

### 1.2.4 Updating for a new ACL release

At UB, SNePS runs on Franz Allegro Common Lisp. Whenever a new ACL version becomes the default version for the CSE Department, the SNePS files have to be recompiled in this new version. Additionally, the files for the other systems SNePS calls (like Garnet) must also be compiled in the new ACL and the paths to these newly compiled files must be correctly assigned. Any change to the paths in the source code must be done using CVS to insure the changes will remain after further updates or alterations.
NOTE: recompiling without making any changes to the source code does *not* require using CVS.

## 1.3 The notation and structure of this report

### 1.3.1 Font styles used

The font used for the main text of this report is in Roman style.

Input to and output from the SNePS system is in `Type` style. These input and output sections are differentiated by the **Bold** statements (**Input:** and **Output:**) which precede them. Included in the input line is the prompt for either the terminal window or lisp. Examples of these prompts are `pollux<~/src>` and `cl-user(1):`, respectively. Comments within input or output are in *italics*.

The word **Code:** indicates that the following `Type` style section is an example of changes to the source code. A series of dots (`...`) indicates that there was more code or output, but it was irrelevant and/or too extensive to include in the example.

### 1.3.2 Section breakdown

The next section tells you how to set up a CVS working directory with SNePS source code in it. Section 3 describes the changes that need to be made before and/or in addition to the proposed alterations. Section 4 walks you through every step needed to convert the updated code from the CVS working directory to a compiled system in its official release directory. The last section deals with the final steps needed to complete a new release — including permissions, download files, web instructions, the SNePS manual and user notification.

# 2 Before making any changes...

## 2.1 What NOT to do

**DO NOT** just copy SNePS source code to a separate directory to alter and test, because it needs to be in the "CVS working directory" for *re-insertion* into the "official" CVS repository (where changes are maintained).

## 2.2 Making a CVS working directory

First, create a new directory which will be the CVS working directory. All changes to the code should be done in this directory. Second, copy the SNePS source files into this directory:

from the working directory, invoke the command: `cvs checkout Sneps`.

This creates a sub-directory called "Sneps" in the directory *from which it was called*. That directory will then contain all "source" files (.lisp, .atn, etc.) from the current version of SNePS — NOTE: it will not include non-source files (.fasl, .o, etc.).

The checkout procedure also puts CVS information — home repository, etc — into directories named `CVS`. Do not remove or alter these directories or you will not be able to commit the altered SNePS code back to the CVS repository (which is located in `/projects/snwiz/src/CVS-Repository/`).

I made a directory in `/projects/snwiz/src/` called `flj26`. From `flj26` I called `cvs checkout Sneps`.

**Input:**

```
pollux<~/src>cd flj26/
pollux<flj26>cvs checkout Sneps
```

**Output:**

```
cvs checkout: Updating Sneps
U Sneps/COPYING
U Sneps/TAGS
U Sneps/load-logical-pathnames.lisp
U Sneps/load-sneps.lisp
U Sneps/logical-pathnames.lisp
U Sneps/packages.lisp
U Sneps/system-utils.lisp
U Sneps/system.lisp
cvs checkout: Updating Sneps/Doc
U Sneps/Doc/COPYING
U Sneps/Doc/GARNET
U Sneps/Doc/NEWS
.
.
.
U Sneps/xginseng/toplevel.lisp
U Sneps/xginseng/xginseng.lisp
```

Now `flj26` contains a directory (called `Sneps`) with all the source files in it, but no .fasl, .o, etc. files

To understand the options available when using `checkout` (or any other CVS command), type `cvs -H` followed by the command name:

**Input:**

```
pollux<~>cvs -H checkout
```

**Output:**

```
Usage:
  cvs checkout [-ANPRcflnps] [-r rev | -D date] [-d dir]
    [-j rev1] [-j rev2] [-k kopt] modules...
        -A      Reset any sticky tags/date/kopts.
        -N      Don't shorten module paths if -d specified.
        -P      Prune empty directories.
        -R      Process directories recursively.
        -c      "cat" the module database.
        -f      Force a head revision match if tag/date not found.
```

```
        -l      Local directory only, not recursive
        -n      Do not run module program (if any).
        -p      Check out files to standard output (avoids stickiness).
        -s      Like -c, but include module status.
        -r rev  Check out revision or tag. (implies -P) (is sticky)
        -D date Check out revisions as of date. (implies -P) (is sticky)
        -d dir  Check out into dir instead of module name.
        -k kopt Use RCS kopt -k option on checkout.
        -j rev  Merge in changes made between current revision and rev.
(Specify the --help global option for a list of other help options)
```

# 3 Making changes

## 3.1 Initial changes

Once the code has been checked out to the the "CVS working directory":
`/projects/snwiz/src/flj26/Sneps` (in this case),
several changes must be made before compiling can be done.

### 3.1.1 Recompile Garnet and update the paths

First, **only if ACL has been changed**, Garnet needs to be recompiled in the new ACL version, and its paths need to be redefined. The current Garnet directory has a README file that explains how to recompile Garnet. The garnet paths that need updating are:

- *sneps-garnet-directory* in `load-sneps.lisp`
- *sneps-garnet-directory* in `/projects/snwiz/bin/sneps.lisp`
  (actually, the file to which this file is linked)
- Your-Garnet-Pathname in `garnet-loader.lisp` in the garnet directory within `snwiz/Install`.

### 3.1.2 Update the SNePS path

In the file `load-sneps.lisp`, alter the pathname for the sneps directory (which is stored in the variable `*sneps-directory*`) to the current path to the CVS working directory.

**Code:**

```
(defvar *sneps-directory*
    #+unix      "/projects/snwiz/src/flj26/Sneps"
...
```

Now, the code can be compiled and tested in the CVS working directory.

## 3.2 Additional changes

### 3.2.1 Updating dates

Each SNePS source code file has a banner at the top with a copyright date in it. When updating or altering the SNePS source code, it is appropriate to update the copyright banner as well (assuming the last change was not in the same calendar year). This should be done using scripting code, because there are many files.

There is also a constant in the file `sneps/fns/snepstop.lisp` called `*copyright-years*` which must also be updated.

### 3.2.2 Updating the SNePS version number

**If this is a new release**, then the version number must be updated. The parameter `*sneps-version*` is defined in the file `load-sneps.lisp`. Whenever a user loads SNePS, this tells the user the current version. Below is the change made for the SNePS 2.6.0 release:

**Code:**

```
(defparameter *sneps-version*
    (format nil "SNePS-2.6 [PL:0 ~a]"   ;; altered code
          (let ((date-string "$Date: 1999/08/19 16:38:25 $"))
         ...
```

This sneps version is important, because the `PL` stands for *patch level* and is used later for tagging purposes (see Section 4.2).

**Note:** In the code above, there is a `date-string`. You do *not* need to set this. When the code changes are finalized and "committed" back to the CVS Sneps repository, the official timestamp for these files is set. This same timestamp is *automatically* inserted into this code!! It is actually written into the code in the CVS working directory *before* it is written to the CVS repository.

### 3.2.3 Altering the code

Now you are ready to make changes to the source code.
**Warning:** After making any change in the sneps code, **you must exit and restart ACL before reloading SNePS**. This allows the variables to be re-defined with the correct starting values.

After testing your changes, make the code accessible to the rest of the research group for additional testing. This way they can make sure their altered research work runs using this new code *before* it becomes the official release.

Any new files added to the SNePS code need to be *officially* added using `cvs add <filename>` and `cvs commit <filename>` — for more information, see the CVS web instructions at the following URL:
`http://www.loria.fr/~molli/cvs/doc/cvs_7.html#SEC61`.
Likewise, removing a file also requires special commands — `cvs remove <filename>`:
see `http://www.loria.fr/~molli/cvs/doc/cvs_7.html#SEC62`.
I have no personal experience adding or removing files.

# 4 Making the altered code the official SNePS release

## 4.1 Committing the new version into the Sneps CVS Repository

The first step to making this new code the official release is to commit it back to the CVS repository. Only actual alterations to the code are noted — code that was changed, then changed back to the original file, would not be considered as altered (even though the file date makes it seem newer than the original code). Recall the instructions about adding or removing entire files mentioned in Section 3.2.3.

During the commit process, the date/timestamp in `load-sneps.lisp` is updated; and (one at a time) each file has its changes officially noted in the CVS repository, and (if changed) the file's version number is advanced — **both in the CVS Repository and in the CVS working directory!!** This means that, if the commit process halts part way through, any already committed files will have new version numbers which will be recorded in the just-altered CVS repository.
**Caution:** If you have an empty file (e.g. A .bbl file for a .tex document that had no references in it but was BibTeX-ed, anyway), the commit process will experience a segmentation fault and exit. This will leave ALL the files locked in CVS-Repository and in the CVS working directory. To fix the problem and continue on:

1. Unlock all folders by removing any files or directories that begin with `#cvs`

2. Fix the offending problem (e.g. remove the empty file)

3. Repeat the commit procedure

**DO NOT** try to *undo* the "bad" commit (e.g. do not restore the CVS repository to its earlier version), or the altered files' version numbers in the CVS working directory will not coincide with their corresponding files' version numbers in the CVS repository. Commit only works if each altered file has the SAME version number in the CVS repository as in the CVS working directory — only *then* will the changes be noted and the version number updated. If the version numbers do NOT match, commit will exit. In this note, we are referring to the version numbers *for each file* — not the version number for the entire SNePS code.

The call to commit the code is made **from the CVS working directory**.

**Input:**

```
mira<flj26>cvs commit Sneps
```

**Output:**

```
cvs commit: Examining Sneps
cvs commit: Examining Sneps/Doc
cvs commit: Examining Sneps/Logs
cvs commit: Examining Sneps/Tests
cvs commit: Examining Sneps/Tools
cvs commit: Examining Sneps/demo
...
...
Checking in Sneps/load-sneps.lisp;
/projects/snwiz/src/CVS-Repository/Sneps/load-sneps.lisp,v  <--  load-sneps.lisp
new revision: 1.46; previous revision: 1.45
done
Checking in Sneps/logical-pathnames.lisp;
...
Checking in Sneps/packages.lisp;
/projects/snwiz/src/CVS-Repository/Sneps/packages.lisp,v  <--  packages.lisp
new revision: 1.15; previous revision: 1.14
...
```

During the commit process a text editor pops up to ask if you want to make a "log message". I entered "`Altered to load in ACL 6.X and run with a case-mode of case-sensitive-lower.`" I saved that message, and closed the editor. Commit continued, and the editor popped up again. I closed it and was given the option to continue, some other choices... and "!" to have that log be the message for ALL directory changes. I chose "!". Commit then finished ALL the updates with that log message used for each change.

## 4.2 Tagging

Next, you need to tag the new release with a tag reflecting the patch-level This should coincide with *sneps-version* as described in Section 3.2.2.

**Input:**

```
mira<flj26>cvs rtag SNePS_2_6_pl_0 Sneps
```

**Output:**

```
cvs rtag: Tagging Sneps
cvs rtag: Tagging Sneps/Doc
cvs rtag: Tagging Sneps/Logs
...
```

```
...
cvs rtag: Tagging Sneps/snip22
cvs rtag: Tagging Sneps/xginseng
```

## 4.3 Exporting to the new official release directory

### 4.3.1 Umask

Before exporting the new code to the official SNePS release directory, perform a `umask 002` to insure the files have the proper permissions.

**Input:**

```
mira<flj26>umask 002
```

### 4.3.2 Exporting

You are now ready to put the code in the official SNePS release directory in `/projects/snwiz/Install/`. After deciding the name of the new release directory, **do not make the new directory** — that will happen during the exporting procedure. When calling the `cvs export` command, use the following options:
`-r` : use revision tag
`-d` : creates the directory
**Note: DO NOT add a final "/" to the target directory name.**

**Input:**

```
mira<flj26>cvs export -r SNePS_2_6_pl_0 -d /projects/snwiz/Install/Sneps-2.6.0 Sneps
```

**Output:**

```
cvs export: Updating /projects/snwiz/Install/Sneps-2.6.0
U /projects/snwiz/Install/Sneps-2.6.0/COPYING
U /projects/snwiz/Install/Sneps-2.6.0/TAGS
U /projects/snwiz/Install/Sneps-2.6.0/load-logical-pathnames.lisp
...
...
U /projects/snwiz/Install/Sneps-2.6.0/xginseng/node.lisp
U /projects/snwiz/Install/Sneps-2.6.0/xginseng/toplevel.lisp
U /projects/snwiz/Install/Sneps-2.6.0/xginseng/xginseng.lisp
```

## 4.4 Update the SNePS path

In the file `load-sneps.lisp`, alter the pathname for the sneps directory to be the path to the new official release directory.

**Code:**

```
(defvar *sneps-directory*
    #+unix      "/projects/snwiz/Install/Sneps-2.6.0"
...
```

## 4.5 Compiling

### 4.5.1 Preliminaries

**Before compiling, make sure the Garnet version and path are correct and compiled in the current ACL version.**

Start ACL. To compile and load SNePS, load the load-sneps.lisp file from the new release directory, and select option 'e' from the menu.
NOTE: I recommend compiling in xemacs to avoid the "`Loading CLX`" problems when compiling with the 'e' option in a shell.

#### 4.5.2 Compiling example showing the commands

After starting ACL in xemacs:

**Output:**

```
==============================================================
Starting image '/apps/acl-6.1/composer'
  with no arguments
  in directory '/projects/snwiz/Install/Sneps-2.6.0/'
  on machine 'localhost'.

International Allegro CL Enterprise Edition
6.1 [Solaris] (Dec 14, 2001 13:37)
Copyright (C) 1985-2001, Franz Inc., Berkeley, CA, USA.  All Rights Reserved.

This development copy of Allegro CL is licensed to:
   [4549] SUNY/Buffalo, N. Campus

;; Optimization settings: safety 1, space 1, speed 1, debug 2.
;; For a complete description of all compiler switches given the current
;; optimization settings evaluate (explain-compiler-settings).
;;---
;; Current reader case mode: :case-sensitive-lower

cl-user(1):
```

**Now, compile and load SNePS:**

**Input:**

```
cl-user(1): :ld load-sneps.lisp
```

**Output:**

```
; Loading /projects/snwiz/Install/Sneps-2.6.0/load-sneps.lisp

Do you want to load SNePS

    a) FAST (i.e., just load compiled files)
    b) by compiling source files that are newer than their
       compiled version before they get loaded
    c) by compiling all source files before they get loaded
    d) by loading uncompiled source files only
    e) by compiling all source files before they get loaded plus
       compiling and loading all optional systems (installation option)

    Type a, b, c, d or e: e
```

**Input (in the line above, the user entered "e")**

**Output:**

```
  Compiling and loading SNePS and all optional systems....
...
...
```

*Lots of warnings, compilings, etc.*

```
...
...
;;; Writing fasl file
;;;   /projects/snwiz/Install/Sneps-2.6.0/demo/snere/arcinfo/arcinfo.fasl
;;; Fasl write complete
;     Fast loading
;        /projects/snwiz/Install/Sneps-2.6.0/demo/snere/arcinfo/arcinfo.fasl
SNePS-2.6 [PL:0 2002/07/02 20:23:15] loaded.
Type `(sneps)' or `(snepslog)' to get started.
cl-user(2):
```

### 4.6 Dates are automatically altered

As stated before (in Section 3.2.2), the timestamp is automatically updated during the committing of the files. The result can be seen when the user loads SNePS (see the end of the output above).

## 5 Final adjustments

### 5.1 Permissions

No matter how *sure* you are that everyone has permissions... they DON'T! So make sure!!

```
pollux<~/Install>chmod -R go+rX Sneps-2.6.0/
pollux<~/Install>chmod -R g+w Sneps-2.6.0/
```

Why the BIG X? "When X appears in permission it is identical to x if the target file is a directory, or if the file is of any other type but already has at least one execution bit (user, group, or other) set. In all other cases, X is silently ignored."[1]

### 5.2 The fast-loading file

I altered the fast loading file: /projects/snwiz/bin/sneps.lisp, which is linked to the new file: /projects/snwiz/bin/sneps-2_6_0.lisp. The old fast-loading file remains in sneps-2_5_1.lisp.

To do this, first make a new fast-loading file that has the new paths (to the sneps and garnet directories) and name it with the appropriate sneps version (as above). Next, remove the old sneps.lisp file linked to the old fast-loading file:
pollux<~/bin>rm sneps.lisp (called from /projects/snwiz/bin/).

Now create the file sneps.lisp and make the link:
pollux<~/bin>ln -s <target-file> sneps.lisp
where <target-file> is the file that has the new paths to sneps and garnet directories (/projects/snwiz/bin/sneps-2_6_0.lisp in this case).

### 5.3 The "Instructions for running SNePS" web page

If you have a new version (or the new instructions for running SNePS) remember to alter the "Instructions for running SNePS" web page: /web/sneps/instructions.html.

[1]From an online man page for chmod: http://www.mcsr.olemiss.edu/cgi-bin/man-cgi?chmod+1

### 5.4 Export another copy of the code to src

Export another copy of the code to an appropriately named directory in `/projects/snwiz/src/`. This is the code that others will use (by copying some or all of it from `src` or by downloading it from the SNeRG web site) to make personal research alterations.

Make sure to alter the pathname definitions in `load-sneps.lisp` so that those who use it are "reminded" to enter *their specific* pathnames.
I.E. — Change the code regarding `*sneps-directory*` and `*sneps-garnet-directory*` to look like this:

**Code:**

```
;;;;;;;;;;;;;;;;;;;; User customization section ;;;;;;;;;;;;;;;;;;;;

;; Edit the following pathname definitions according to the pathname
;; syntax of your operating system.  Do not include final directory
;; delimiters; be careful with case for Unix pathnames:

(defvar *sneps-directory*
    #+unix      "/sneps/release"
    #+explorer  "snepshost:sneps.release"
    #+symbolics "snepshost:>sneps>release"
    #+vms       "snepshost:[sneps.release"
    #+apple     "snepsdisk:sneps"
    #+mswindows "snepshost:\\sneps\\release"
    "The root of the SNePS directory tree.
Delete the supplied example paths and fill in the actual location
of the top-level SNePS directory at your site.")

(defvar *sneps-garnet-directory* "garnet"
    "The directory that contains the load files of your GARNET installation.
Ignore this variable if you do not have GARNET installed at your site.")
...
...
```

### 5.5 The download file

You now need to tar and compress this source code (located in `/projects/snwiz/src/`) and put it in the ftp sneps download file: `/ftp/pub/sneps`.

```
pollux<~/src>tar cvf Sneps-2.6.0.tar Sneps-2.6.0
pollux<~/src>compress Sneps-2.6.0.tar
pollux<~/src>mv Sneps-2.6.0.tar.Z /ftp/pub/sneps/
```

Moving the `.Z` file to `/ftp/pub/sneps` AUTOMATICALLY puts it in as downloadable from the Sneps FTP web page.

### 5.6 SNePS Manual

If this release is a new version or there are changes that affect running the code, the SNePS Manual will need to be updated. This is typically done in coordination with Stuart C. Shapiro. If a new manual is made, a link to the manual needs to be placed on the SNeRG web site.

### 5.7 Notify SNUG

Lastly, The SNePS Users Group (SNUG) needs to be notified about the changes. This is typically done by Professor Shapiro.