

CVS Download Instructions

The best way to keep up with the latest changes to our project is to download the code from CVS, and keep a constant monitor on the SCM mailing list (shown below). Here are some instructions for getting the code via CVS:

Anonymous CVS access:

This project's CVS repository can be checked out through anonymous (pserver) CVS with the following instruction set. When prompted for a password for anonymous, simply press the Enter key.

```
cvcs -d
:pserver:anonymous@cvs.drools.codehaus.org:/
home/projects/drools/scm login
(no password)
cvcs -z3 -d
:pserver:anonymous@cvs.drools.codehaus.org:/
home/projects/drools/scm co drools
```

Updates from within the module's directory do not need the -d parameter.

Developer CVS access:

Only project developers can access the CVS tree via this method. SSH1 must be installed on your client machine. Substitute <USERNAME> with the proper value. Enter your site password when prompted.

```
export CVS_RSH=ssh
cvcs -z3 -d
:ext:<USERNAME>@cvs.drools.codehaus.org:/hom
e/projects/drools/scm co drools
```

CVS via Fisheye

Fisheye provides links to download CVS HEAD in various archive formats:

- <http://cvs.drools.codehaus.org/viewrep/~tarball=zip/drools//.zip>
- <http://cvs.drools.codehaus.org/viewrep/~tarball=tgz/drools//.tgz>
- <http://cvs.drools.codehaus.org/viewrep/~tarball=tbz2/drools//.tbz2>

Secure CVS for Windows Users

To do development on windows you need to use a native CVS binary with plink for the SSH access and pageant to manage your keys and passphrase - you will also need to use puttygen to generate your keys.

1. Create a directory on your machine where you'd like to install the necessary programs. We suggest `C:\program files\cvs`. Into this directory, download the latest versions of [CVS](#), [Putty](#), [PuttyGen](#), [Plink](#) and [Pageant](#).
2. Add the new directory to your classpath. Go to `My Computer > Control Panel > System > Advanced > Environment Variables` to reach the System Variables window.
3. In the System Variables window, edit the `PATH` variable. Place your new directory at the front.
4. Create a new variable `CVS_RSH` and give it the value `plink`.
5. Create your key with puttygen.
6. Start pageant and add your key. It is preferable to create a key with a passphrase for better security. You may add pageant to your start menu if you wish.

Don't Use Cygwin's CVS

Do not use the cygwin build of CVS. If you have cygwin installed make sure your new directory is in the system path before cygwin's bin directory. The cygwin build of CVS does not work with Pageant.

Named Sessions

If you create named sessions with Putty, you can then use these names with CVS. For instance, you can make a Putty session called "drools", set it to SSH and give it your preferred login. You can then access it with:

```
cv s -d @drools:/home/projects/drools/scm co
drools
```

SmartCVS

As an alternative to the above, and if you would rather try a more graphical UI for CVS, SmartCVS <http://www.smartcvs.com/> supports SSH2, and can generate keys (or import ones). There is both a free foundation and a pay-for version, with varying capabilities.

Working on the Release Candidate Branch

Once we decide on a feature freeze in preparation for release candidates and the release, we branch the CVS repository. The release is always built from latest release-branch. Only high priority bug fixes should be committed to the release-branch. Whenever committing to the release-branch, you must *at the same time* merge those commits into the head.

What follows are the procedures for common development use-cases. In these use cases, it is assumed that the latest release branch tag is named 'branch-2_0', and its corresponding merge-point tag is named 'mergepoint-2_0'.

(More info on [branching and merging](#))

Create a release-branch

```
$ cd $home/drools/head
$ cvs update -C drools
$ maven test-all
$ cvs tag -b branch-2_0
$ cvs tag -r mergepoint-2_0
```

Checkout the branch

```
$ cd $home/drools/branch-2_0
$ cvs checkout -r branch-2_0 drools
$ cd $home/drools/branch-2_0/drools
```

Fix a bug in the release-branch

Fix the bug

```
$ cd $home/drools/branch-2_0/drools
$ cvs update
# fix the bug
```

Commit to the branch, ensuring you have the latest from the branch and that all tests pass

```
$ cvs update
$ maven test-all
$ cvs commit
```

Prepare a clean check of head

```
$ cd $home/drools/head/drools
$ cvs update -C
$ maven test-all
```

Merge from the branch to the trunk

```
$ cd $home/drools/head/drools
$ cvs update -j mergepoint-2_0 -j branch-2_0
# fix any conflicts
$ maven test-all
$ cvs commit
```

Finally, move forward the mergepoint tag in the branch

```
$ cd $home/drools/head/branch-2_0
$ cvs tag -c -F mergepoint-2_0
```

The reason for the mergepoint is to tell cvs not to consider for merging files that have already been merged previously. (This is where subversion shines. But the above is not difficult with CVS.)

If you don't 'move forward' the mergepoint, and there were conflicts resolved during the last merge, you will have to resolve them again.

Release a release-candidate or a full release

```
$ cd $home/drools/branch-2_0/drools
$ cvs update -C
# invoke appropriate maven commands
```

The main point here is to be sure that you create the release from the release-branch.

Additional CVS Info

You can learn more about CVS at <http://cvsbook.red-bean.com>.