Using the CVS

CVS for Moodle Developers

CVS is the **Concurrent Versioning System**, a commonly-used way of managing source code for large software projects. CVS keeps all versions of all files so that nothing is ever lost, and usage by different people is tracked. It also provides ways to merge code if two or more people are working on the same file. All code and all versions are stored on a central server (in the case of Moodle, at Sourceforge).

If you just want to download Moodle using CVS to run a site, then you probably don't need this page - just follow the simpler CVS instructions on the Moodle download page.

1. Joining the project as a developer

So, you've been offered CVS write access to help us develop and maintain Moodle! Welcome aboard!

To be able to write changes into Moodle's CVS archive, you first need to have an account at Sourceforge (registration is free and easy). For the examples on this page, let's assume your username is **myusername** and your password is **mypassword**. Take special note of the sourceforge instructions to create your CVS home directory - something you have to do with every new account to "enable" it for CVS. Basically you just have to use ssh to interactively connect to cvs.sourceforge.net.

Once you have a working Sourceforge account, contact Martin Dougiamas so he can set up your account with write access to particular Moodle directories.

To avoid being prompted for **mypassword** every time you run a CVS command,

follow the Sourceforge directions for using authorized keys. This step is optional, but it can make your CVS experience a lot nicer.

With that done, you should have all the permissions you need, so you just need to set up your machine and download the current sources so you can start working on them.

2. CVS Modules

Within CVS, the word "modules" refers to separate collections of code. In Moodle we have the following modules within our repository:

moodle - the main Moodle source code

contrib - user contributions and other assorted code in development

mysql - a customised phpMyAdmin to plug into Moodle for database admin

windows-cron - a small package that makes cron possible on Windows systems

docs - various extra user-contributed documentation

Most people are working on the existing features in the **moodle** module, but many are also contributing new ideas in the **contrib** modules. Once code reaches a certain level of maturity in the **contrib** area, it can be migrated over into the main **moodle** tree.

3. Basic CVS Commands

3.1 CVS on Unix

Sourceforge CVS uses ssh as a transport layer for security, so you will have to set a CVS_RSH environment variable in your Unix shell. It's best to put these commands in your .bashrc or .cshrc so you don't have to type it all the time:

```
setenv CVS_RSH ssh (for csh, tcsh etc)
export CVS_RSH=ssh (for sh, bash etc)
```

Next, you can check out the latest development version of Moodle using this (all one line):

```
cvs -z3 -d:ext:myusername@cvs.sourceforge.net:/cvsroot/moodle co moodle
```

The command is similar for other CVS modules:

```
cvs -z3 -d:ext:myusername@cvs.sourceforge.net:/cvsroot/moodle co contrib
```

Don't try to do run this first CVS command over an existing moodle installation: start fresh with a new directory.

Note that you will be prompted for **mypassword** for each command unless you set up authorized keys.

Now, you should have a new 'moodle' directory. You can rename it and move it around if you like. Go into it:

```
cd moodle
```

All the latest Moodle files should be in there. You can now change files in your copy. To compare your files and directories against the main CVS copy on the server use cvs diff, e.g.:

```
cvs diff -c config-dist.php
cvs diff -c lang
```

To fetch the latest updates from the server use:

```
cvs update -dP
```

To copy your new files back to the server you would do something like:

```
cd lang/ca
cvs commit
```

You will be prompted to add some comments (depends on your default text editor) ... add a meaningful comment and close the editor ... the files will be sent to Sourceforge and stored. Done!

To save more time you can put default arguments into a file called .cvsrc in your home directory. For example, mine contains:

```
diff -c
update -dP
```

Try 'cvs help' for more details ...

3.2 CVS on Windows

First, you need to download a completely fresh copy of Moodle using your developer account.

1. Get TortoiseCVS from tortoisecvs.org and install it, then reboot.
2. Find or create a new folder somewhere where you want Moodle to be downloaded to.
3. Right-mouse-click that folder and choose "**CVS Checkout**" from the menu. You should see a dialog box.
4. Copy this text into the CVSROOT field (using your own username!):

```
:ext:myusername@cvs.sourceforge.net:/cvsroot/moodle
```

5. Under the "Module" field, type "**moodle**" to get the latest development version of Moodle, "**contrib**" to get the contributions directory, or "**mysql**" to get the MySQL Admin module.
6. Press the button: "**OK**" and everything should be downloaded.

A dialog box should show all the files being downloaded, and after a while you should have a complete copy of Moodle. After this first checkout, you can fetch the latest updated files from the CVS server:

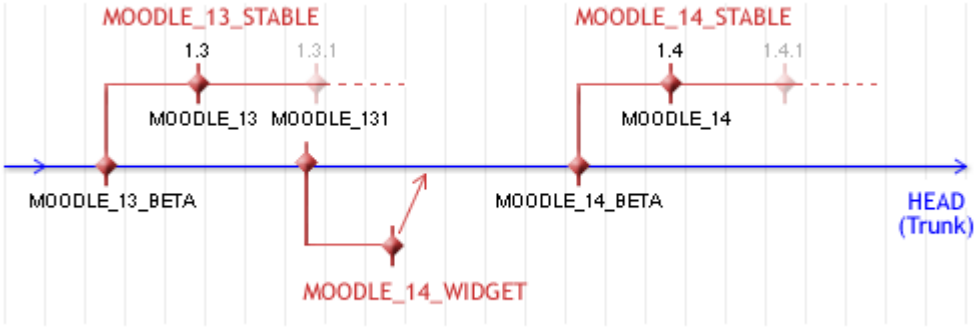
1. Right-mouse-click on your Moodle folder (or any file) and select "**CVS Update**".
2. Sit back and watch the logs scroll by. Take note of conflicts that may occur if your local code has changes that conflict with the incoming versions - you will need to edit these files and resolve the conflicts manually.

After modifying files (you will notice their icons change from green to red!), you can commit them back to the CVS server like this:

1. Right-mouse-click on your Moodle folder (or any file) and select "**CVS Commit...**".
2. In the dialog box, type a clear description of the changes you are committing.
3. Click "OK". Your changes will be sent to the server.

4. Working with Branches

This diagram shows how the main **moodle** module branches into different versions over time.



To see all the current tags and branches that are available, use this command on any old file (such as index.php in the top moodle directory):

```
cvs status -v index.php
```

Some tagging guidelines:

- Tag and branch names should always be all upper-case.
- Tags and branches should ALWAYS be applied to the **entire module** (all of Moodle). Don't tag individual files or directories.
- We don't allow renaming of tags because people may be relying on them, so get them right the first time!

4.1 Trunk development

The Trunk of CVS is the main development version of Moodle. In CVS it is also known as the **HEAD**, or default branch.

Moodle developers try to keep this stable as possible, but as it usually contains new code it probably has bugs and small instabilities.

Every now and then we decide the product has enough features to make a release. At this time, the trunk is tagged with a **MOODLE_XX_BETA** tag (in case we ever want to roll back to that point) and a new branch is formed for the release, called MOODLE_XX_STABLE.

A Beta package is also released at this point - it's for testers who don't use CVS

but want to test the latest features and report bugs.

4.2 Stable branches for each release

As soon as the stable branch **MOODLE_XX_STABLE** is created, development efforts will fork into two streams for a while. Some people may continue working on new features in the trunk for the next release, but most developers should be concentrating on using the current **STABLE** branch and fixing bugs that are found in it.

You can switch your local copy of Moodle to the STABLE version using the following command in Unix from the root directory:

```
cvs update -dP -r MOODLE_XX_STABLE
```

After that, all the commands described above will apply to that stable version. To return to the trunk version just issue:

```
cvs update -dPA
```

On Windows clients you should have a menu from which you can choose the branch.

Once the new STABLE branch really stabilises, a release can be declared. Packages are created for distribution and the branch will be tagged (by Martin) with a tag named: **MOODLE_XXX**

Periodically, bug fixes in the STABLE branch should be merged into the trunk so that they become available in future versions of Moodle. A floating tag called MOODLE_XX_MERGED will be maintained to keep track of the last merge. The procedure for such a merge is as follows:

1. Get out the very latest trunk version.

   ```
   cvs update -dPA
   ```

2. Merge everything on the branch since the last merge, into your trunk version

   ```
   cvs update -kk -j MOODLE_13_MERGED -j MOODLE_13_STABLE
   ```

3. Carefully watch the update logs for conflicts, and fix every file that you see with a conflict

4. Check the merged copy back into CVS trunk version

```
cvs commit
```

5. Go back to the branch version

```
cvs update -dPr MOODLE_13_STABLE
```

6. Update the floating merge tag so that this process can be repeated next time

```
cvs tag -RF MOODLE_13_MERGED
```

Finally, the values for *$version* in all the Moodle version.php files within the stable branch should not be updated at all if possible (except the last digit if necessary). The reason is that someone updating from a very stable version to the next very stable version could miss database upgrades that happened on the trunk.

4.3 Feature branches for large changes

Occasionally, there may be a very large feature that needs to be checked in so several people can work on it, but it is too unstable to be included in the main development trunk.

In these cases a short-term branch can be created to work on the feature, and then merged back into the main trunk as soon as possible. An example called **MOODLE_14_WIDGET** branch can be seen in the above diagram.

If you need to do this for your new WIDGET feature, follow these steps:

1. Discuss with other developers to make sure it's necessary!
2. Make a new tag on the **trunk** (for all of moodle) called **MOODLE_XX_WIDGET_PRE**

```
cvs tag -R MOODLE_XX_WIDGET_PRE
```

3. Create your branch called **MOODLE_XX_WIDGET**

```
cvs tag -Rb MOODLE_XX_WIDGET
```

4. Work in that branch until the feature is reasonably stable. Commit as necessary.

```
cvs commit
```

5. When ready, merge the whole branch into the trunk, fix conflicts, commit it to the trunk and then abandon the branch.

```
cvs update -dPA
cvs update -kk -j MOODLE_XX_WIDGET
cvs commit
```

Good luck, be careful and have fun!

Moodle Documentation

Version: $Id: cvs.html,v 1.24.2.1 2005/07/07 15:46:23 moodler Exp $