# CVSspam Documentation

For the latest version, visit http://www.badgers-in-foil.co.uk/projects/cvsspam/.

# 1. Installation

## 1.1. Installing CVSspam files

The CVSspam scripts may be located anywhere in the CVS server's filesystem. If you only have access to the CVS server using CVS, you can still install CVSspam by adding the files to the repository's CVSROOT module.

### 1.1.1. Installation Using CVS

Check out your repository's `CVSROOT`.

```
...set $CVSROOT to point at your repository...
$ cvs checkout CVSROOT
$ cd CVSROOT
$ ls
CVS           commitinfo  cvswrappers  loginfo  notify   taginfo
checkoutlist  config      editinfo     modules  rcsinfo  verifymsg
```

Place `record_last_dir.rb`, `collect_diffs.rb` and `cvsspam.rb` into this directory.

Add these three filenames into `CVSROOT/checkoutlist`

```
# The "checkoutlist" file is used to support additional version controlled
# administrative files in $CVSROOT/CVSROOT, such as template files.
#
# The first entry on a line is a filename which will be checked out from
# the corresponding RCS file in the $CVSROOT/CVSROOT directory.
# The remainder of the line is an error message to use if the file cannot
# be checked out.
#
# File format:
#
#       [<whitespace>]<filename><whitespace><error message><end-of-line>
#
# comment lines begin with '#'
record_lastdir.rb
collect_diffs.rb
cvsspam.rb
```

**`cvs add`** the three scripts to the repository, then **`cvs commit`** them, and the modified `checkoutlist`.

In `commitinfo` and `loginfo` you can now refer to the scripts with `$CVSROOT/CVSROOT/record_lastdir.rb` and `$CVSROOT/CVSROOT/collect_diffs.rb`

### 1.1.2. System-wide Installation

You want to install the scripts system-wide, rather than in a specific repository's CVSROOT. Pick a location for the three scripts (like `/usr/local/lib/cvsspam/`).

## 1.2. Configure CVS

To install CVSspam you'll need to alter the repository's configuration files.

Alter `commitinfo` to call the CVSspam script that records the directories that have been committed:

```
# The "commitinfo" file is used to control pre-commit checks.
# The filter on the right is invoked with the repository and a list
# of files to check.  A non-zero exit of the filter program will
# cause the commit to be aborted.
#
# The first entry on a line is a regular expression which is tested
# against the directory that the change is being committed to, relative
# to the $CVSROOT.  For the first match that is found, then the remainder
# of the line is the name of the filter to run.
#
# If the repository name does not match any of the regular expressions in this
# file, the "DEFAULT" line is used, if it is specified.
#
# If the name "ALL" appears as a regular expression it is always used
# in addition to the first matching regex or "DEFAULT".
```

**^*myproject /path/to*/record_lastdir.rb**

> **Warning**
>
> Users without direct administrative control over their repository please take note:
>
> An error in the initial configuration of `commitinfo` can can prevent commits to the modules your rule matches. This will cause you a major problem if you use the "ALL" or "DEFAULT" rules, as you will no longer be able to commit changes to the `CVSROOT` module to fix the problem.
>
> If you use the "DEFAULT" rule, take the precation of specifying a rule for `CVSROOT` that will never fail:
>
> ```
> # always allow commits to CVSROOT
> ^CVSROOT  /bin/true
>
> # Invoke CVSspam lastdir script,
> DEFAULT ...
> ```

**Note:** If you are using a version of CVS from the 1.12.x series, or later, the format of `commitinfo` has changed, and now requires that 'format strings' appear on the line following the name of your script. If you see messages like,

```
cvs commit: warning: commitinfo line contains no format strings:
    "/home/dave/projects/cvsspam/record_lastdir.rb"
Appending defaults (" %r/%p %s"), but please be aware that this usage is
deprecated.
```

then follow the advice, and add the example format string to silence the warning from CVS.

The resulting line should now look like,

```
^myproject /path/to/record_lastdir.rb %r/%p %s
```

Now you need to alter `loginfo` to record the log entry made by the user (and send off the email):

```
# The "loginfo" file controls where "cvs commit" log information
# is sent.  The first entry on a line is a regular expression which must match
# the directory that the change is being made to, relative to the
# $CVSROOT.  If a match is found, then the remainder of the line is a filter
# program that should expect log information on its standard input.
#
# If the repository name does not match any of the regular expressions in this
# file, the "DEFAULT" line is used, if it is specified.
#
# If the name ALL appears as a regular expression it is always used
# in addition to the first matching regex or DEFAULT.
#
# You may specify a format string as part of the
# filter.  The string is composed of a '%' followed
# by a single format character, or followed by a set of format
# characters surrounded by '{' and '}' as separators.  The format
# characters are:
#
#   s = file name
#   V = old version number (pre-checkin)
#   v = new version number (post-checkin)
#
# For example:
#DEFAULT (echo ""; id; echo %s; date; cat) >> $CVSROOT/CVSROOT/commitlog
# or
#DEFAULT (echo ""; id; echo %{sVv}; date; cat) >> $CVSROOT/CVSROOT/commitlog

^myproject /path/to/collect_diffs.rb --to me@somewhere.invalid %{sVv}
```

**Note:** The expression you use to select the project (the first thing on the line) must be the same in `commitinfo` and `loginfo`.

Commit your changes to these files. You should see a message from CVS like 'rebuilding administrative database'. You are now be ready to test the setup.

Checkout a copy of *`myproject`* and commit a change. An email should be sent to the address you specified.

## 1.3. Configuration File

You can specify CVSspam options in an external configuration file. See the example `cvsspam.conf` provided for information about the full set of options available.

CVSspam will load `$CVSROOT/CVSROOT/cvsspam.conf` or `/etc/cvsspam/cvsspam.conf`, if they exist. You can specify another location with the `--config` option to `collect_diffs.rb`.

If you want to put your config into the repository, follow the instructions above for installing files into `CVSROOT`, and remember to add `cvsspam.conf` to the `checkoutlist`.

## 1.4. Sending Email

### 1.4.1. Sendmail / SMTP

By default, CVSspam will attempt invoke an external program to send out messages. This program will normally be `/usr/sbin/sendmail`, but you can specify another using the $sendmail_prog setting in the config file. (Whatever you specify must accept the email on it's stdin, and understand the '-t' flag to cause it to take message headers from this input.)

By specifying `$smtphost` in the configuration file, you can make CVSspam contact an SMTP server directly, rather than using an external program. Only use this option if there is no MTA installed on your CVS server.

```
$smtphost = "mail.example.domain"
```

### 1.4.2. From Address

When all CVS users have real accounts on the server, the sender address in generated emails will be derived from the username of the commiter. This relies on a standard behavior of sendmail-like MTAs; we don't specify any address, so the MTA must add one.

CVS accounts are commonly aliased to a less privileged account on the server, such as 'nobody', for extra security. Unfortunately, combined with the default CVSspam configuration, this will result in all notifications appearing to originate From 'nobody@hostname', obscuring the actual commiter.

CVSspam provides a `--from` option which you can use to specify the person who is really committing. Further, CVS provides a magic $USER keyword in the `loginfo` file that will be replaced with the CVS account name.

```
# loginfo entry for aliased accounts
^myproject /path/to/collect_diffs.rb --from $USER --to me@somewhere.invalid %{sVv}
```

If you would like control over the email address used for each user, consider using a platform-specific meachanism for customising how your MTA formats user's addresses (e.g. `/etc/email-addresses`). Additionally, CVSspam will make use of `CVSROOT/users`, if this exists, to derive the email address. The format of `CVSROOT/users` is one username:address pair on each line (as documented in Open Source Development with CVS (http://cvsbook.red-bean.com/cvsbook.html#users))

### 1.4.3. Global Email Addresses

Recipient email addresses can be put in the configuration file as well as in each `loginfo` entry. For example,

```
addRecipient "code-review@somewhere.invalid"
addRecipient "project-owner@somewhere.invalid"
```

## 1.5. Debugging installation problems

**1.** No email coming from CVS commits, and no error messages on the command line

Did you specify the right email address?

Does the regular expression you specified in `commitinfo` and `loginfo` really match the project? Try changing the entry to something like

```
^myproject echo "Hello world"
```

When you commit a change to *`myproject`*, 'Hello world' should appear in your terminal. If it doesn't, verify that the expression on the left is correct.

Check that the CVS server correctly handles email. By default CVSspam invokes sendmail. Try running sendmail by hand on the CVS server machine

```
$ echo test | /usr/sbin/sendmail me@somewhere.invalid
```

**2.** Why do I see the message `No such file or directory cvs` on the console after committing?

The cvs executable is probably not in the default executable search path available to the CVSspam scripts. Tell them explicitly where to find cvs by setting the $cvs_prog option in the configuration file.

**3.** When I try to commit, I see messages like,

```
cvs commit: loginfo:32: no such internal variable $USEt
cvs commit: loginfo:32: no such internal variable $USEH
cvs commit: loginfo:32: no such internal variable $USExist
```

and other garbled $var names, but I don't use any variables like these, just the `$USER` keyword

This seems to be a bug in CVS (in at least version 1.12.9). Try upgrading the server.

# 2. Integration Options

## 2.1. Bug Tracking Software

CVSspam can generate simple links to web-based bug tracking systems. Links are formed from specially formatted text in the commit-log message.

### 2.1.1. Bugzilla

For Bugzilla (http://www.mozilla.org/projects/bugzilla/), when a CVS log comment contains text like **`Fix for bug 123...`**, the text "bug *nnn*" will become a hyper-link to that Bugzilla page in the generated email.

To enable, give your Bugzilla's URL in CVSspam's configuration file

```
$bugzillaURL = "http://bugzilla.mozilla.org/show_bug.cgi?id=%s"
```

The marker `%s` tells CVSspam where in the URL to put the bugId from the log message.

### 2.1.2. JIRA

For JIRA (http://www.atlassian.com/software/jira/), include the issueId in the log comment. JIRA issue Ids have a project name and issue number, separated by a dash. For example `JRA-1545`.

To enable, give your JIRA installation's URL in CVSspam's configuration file

```
$jiraURL = "http://jira.atlassian.com/secure/ViewIssue.jspa?key=%s"
```

The marker `%s` tells CVSspam where in the URL to put the issue Id from the log message.

### 2.1.3. RT

For systems that like to talk about *tickets*, CVSspam will make links from text in the log comment that looks like "ticket *nnn*" (where *nnn* is a number). For instance with RT (http://fsck.com/projects/rt/), supply the location of `Display.html`

```
$ticketURL = "http://localhost/rt2/Ticket/Display.html?id=%s"
```

## 2.2. CVS Web Frontends

If you have ViewCVS (http://viewcvs.sourceforge.net/), CVSweb (http://www.freebsd.org/projects/cvsweb.html) or Chora (http://www.horde.org/chora/) web-access to your repository, CVSspam can generate links to it in the emails. Links the file before and after the commit are very useful for images, as only changes to binary text files are mailed. You'll get a link to the side-by-side view of the changes as well.

You may only specify one of these three options.

### 2.2.1. ViewCVS

To enable ViewCVS support, specify the URL of the top-level ViewCVS directory in `cvsspam.conf`.

```
$viewcvsURL = "http://localhost/cgi-bin/viewcvs.cgi/"
```

### 2.2.2. CVSweb

For CVSweb, specify the URL of `cvsweb.cgi`,

```
$cvswebURL = "http://localhost/cgi/cvsweb.cgi/"
```

### 2.2.3. Chora

For Chora, specify the URL of the directory containing `cvs.php`,

```
$choraURL = "http://localhost/hord/chora/"
```

### 2.2.4. Multiple CVS repositories

If ViewCVS or CVSweb are configured for multiple repositories, you can specify which to use by defining a value for `$repository_name`. You can either hardcode the name of a repository, or use the special value `GUESS`, which will cause CVSspam to use the last segment of the CVSROOT path as the repository name.

```
# this is the top-secret repository
$repository_name = "Secret Projects"
```

Note that `GUESS` is not surrounded by quotes.

```
# our repositories are named after their containing directories,
$repository_name = GUESS
```

`GUESS` should be handy in environments like GForge (http://gforge.org/)