## Chapter 13

# Hooks, and Why They Can Be Useful

*Hooks* are commands, or scripts, that are triggered by events occurring in your repository. These events occur when you commit, push, and pull. Hooks have access to environment variables, can receive arguments containing revision information, and allow you to control whether a particular repository event can proceed or gets blocked. They live in, but are not checked in to, your repository, and a given event can have a hook that fires before, during, or immediately after said event—for example, pre-commit, commit, and post-commit. In this chapter, you'll have a look at what hooks are available in SVN, Git, and Mercurial, how you can enable hooks, and finally look at some examples of what you can do with hooks.

## SVN Hooks

If you look in the `SVN_DIR/hooks` directory you'll see a set of template hook files listed. As you `svn update` and `svn commit` away, you'll mostly be interested in the following hooks.

### start-commit

This hook runs before the commit transaction has **start**ed. It is passed three arguments: the path to the repository, the username of the person doing the commit and, since SVN 1.5, a colon-separated list of the capabilities of the SVN client making the commit.

> ***Caution*** *This list of capabilities is self-reported by the client. The `start-commit.tmpl` contains a warning that you should not make any security assumptions based on this list of capabilities, or even assume that this list has been reliably reported.*

The hook's exit status is used to determine whether the commit should be allowed to continue. To stop the commit from occurring, a non-zero exit status should be returned and any data written to `stderr` will be sent back to the SVN client.

## pre-commit

This hook runs *after* the SVN transaction has been created but *before* it is actually committed. It is passed two arguments: the path to the SVN repository and the name of the transaction about to be committed.

If the hook wants to stop the commit from continuing, it should exit with a non-zero status. As with the start-commit hook, any data written to `stderr` by the hook will be sent back to the SVN client.

What does a non-zero exit status mean? A program runs in a process. When a process finishes, it passes a small numeric value to its parent process. This exit status, also known as a return code, indicates success or failure: an exit status of zero indicates success; any non-zero exit status means an error occurred.

The pre-commit hook can be used to check for nonempty commit messages, or to check that commits to your `bugfixes` branch include a ticket number from your bug tracking system in the commit message.

## post-commit

The post-commit hook runs after a successful commit. It is passed two arguments: the path to the repository and the new revision number from the fresh commit.

The exit status of this hook is ignored, though, as it can have no influence over the commit (because it has already been committed).

As it runs after a successful commit, this hook can be used to send out e-mail notifications containing details about the commit.

## pre-revprop-change

Revision properties in SVN aren't actually versioned themselves. The pre-revprop-change hook, together with its counterpart post-revprop-change, which you'll meet very shortly, make it possible for you keep track of changes to these values using an external system, such as a log file or a database, if you need to.

> ***Note*** *When I say that the revision properties aren't versioned, it means that any successful change to them is destructive because the old value will be lost forever. It is recommended that this hook be used to back up the old value somewhere.*
>
> *Because of the destructive nature of changes to revision properties, this hook must exist for these changes to go through. If an attempt is made to change a revision property without the pre-revprop-change hook being enabled, SVN acts as though the hook is there and has returned a non-zero status.*