FishEye User Manual

Version 1.3

1. Introduction

1.1. Starting Points

Your source code repository contains an abundance of valuable information, but it is not always easy to extract. Let FishEye tame your repository. Put your repository to work.

The best way to get an overview of FishEye's features is to take the **Guided Tour**.

Read the **Quick Start Guide** if you are installing FishEye for the first time.

For FishEye troubleshooting information, see the **FAQ** or the **Online Forums**.

2. FishEye

2.1. Using FishEye

The sections below describe the different screens in FishEye and the different information you can retrieve from them. Each Tab/View has a number of panes, and each pane is described seperately.

2.1.1. Header

The header along the top of the FishEye screen will not change as you browse through the different screens.

On clicking the FishEye icon in the top left hand corner, this will take you to the list of FishEye repositories. Underneath the FishEye icon is the directory that is currently being browsed.

The user currently logged in will be shown at the top right hand corner or it will say 'Guest' if nobody is logged in. If a user is logged in then they can change FishEye settings such as passwords, notifications and display settings by clicking on the Profile link

The 3 tabs at the top right hand corner are described below:

- Browse Tab
- Changelog Tab
- Search Tab

2.1.2. Browse Tab

The Browse Tab is the first screen you see when you sign into FishEye. Click on the Browse Tab in the top right hand corner to get to this screen.

The Browse View lets you explore the revisions, files and directories in your repository. This screen helps you quickly navigate to the file you are looking for, as well as presenting useful information about the directory you are looking at.

Recent Changelog pane

The top of the right-hand column shows the most recent changesets for this directory subtree. The RSS icon allow you to subscribe to an RSS feed of the recent changes in this subtree.

Files pane

The list of files is shown in the filename page below the changelog pane on the right hand side. You can sort the filename pane by name, age or author.

Line History Graph pane

This graph shows the total line-count of MAIN or trunk over time for this directory subtree. This line-count does not include binary files, but does include every other file. If you have a branch-constraint specified, then the line-count history of that branch is also shown.

Constraint pane

You can specify a constraint that controls the information that is shown in the Browse View.

Branch	Will only show files and recent-changes on that branch.
Author	Displays the most-recent revision in each file that was checked in by the given author. Only shows recent-changes checked in by the given author.
Tag	Only shows files/revisions that are tagged with the given tag.
Date	Only shows revisions and changesets that were created on or before that date. Must be of the form `YYYY-MM-DD`, `YYYY-MM` or `YYYY` (you can use `/` instead of `-`).

Sub Directories pane

This is a list of the different folders under the repository. It is ordered by default in alphabetical order but can be ordered by last-commit or first-commit entries.

2.1.3. Changelog Tab

The Changelog allows you to browse the changes made to your repository chronologically. It provides a calendar in the left-hand column to allow you to navigate to any time in the history of your repository. You can also drill-down into a subdirectory using the directory tree in the left-hand column.

The changesets are shown in the right-hand column. They are ordered most-recent-first.

You can move forward/backward in time using the controls at the top (and bottom) of the right-hand column. A timeline is also shown on the calendar highlighting the range of

changesets displayed on this page. Clicking on the calendar will allow you to navigate to the changesets relevant to that period of time in the calendar

Constraint Mode

You can specify a constraint that controls the information that is shown in the Changelog.

Branch	Will only show changesets on that branch.
Author	Only shows changesets checked in by the given author.
Tag	Only shows changesets that contain revisions tagged with the given tag.
Date	Only shows changesets that were created on or before that date. Must be of the form `YYYY-MM-DD`, `YYYY-MM` or `YYYY` (you can use `/` instead of `-`).

2.1.4. File History View

The File History View shows the different revisions of a file.

Branch History

This is a diagram of the branches and revisions of this file. Clicking on the diagram will focus the window to that branch/revision.

Line History Graph

This graph shows the total line-count over time for this file. If you have a branch-constraint specified, then the line-count history of that branch is also shown.

Arbitrary Diffs

This allows you to request a diff between any two revisions of the file. You can use revision numbers or tag-names.

2.2. User Profile

Users can change FishEye settings such as passwords, notifications and display settings. To view your own user profile, log into FishEye, choose a repository and on the top right-hand corner of the page, click on the link **Profile.**

Note:

> Always click `save` after making any changes.

Detailed below is a description of each tab, and its contents.

2.2.1. Display Settings Tab

The options in this tab allow you to amend the display settings.

General

Length of tag list	Default is Medium. The option to show the list of tags for a file. This can be changed to show none or all(long).
Show Linecount History Graph	Default is Yes. This is the graph that appears on the left hand side of the Browse and Changelog screen.
Show hidden directories	Default is no. Do not show the hidden directories within any folder lists.
Show empty directories	Default is yes. The option to see any empty directories within any folder lists.
Timezone	Default is the timezone of the FishEye server.

Changelog

Changesets per page	The default is 30 per page.
Maximum files shown in a changeset	Default is 5.

Diff View

Truncate long diffs	Default is yes. Only show part of the diff if, the diff contains many lines of code.
Diff mode	Default is Unified. Can be changed to 'Side-by-side' diffs. Implemented in 1.3.1
Line wrapping	Default is None. None is when long lines will never word-wrap. Soft is when long lines will word-wrap. Implemented in 1.3.1.

Source View

Default annotation mode	Default is age. It can be changed to Author or none.
Tab width	Default is 8. Can be changed to between 1 and 10. Implemented in 1.3.1

2.2.2. Email Tab

The settings in this tab allow you to change your email address and your display name.

Display Name	Name displayed for the user currently logged in.
Email address	The address all email notifications will be sent to.
Email Format	Default is text. Can be changed to be sent as HTML

2.2.3. Change Password Tab

Option to be able to change your password if required.

Note:
The passwords are case sensitive

2.2.4. Author Mapping Tab

This functionality is currently not used in FishEye.

2.2.5. Watches Tab

By adding a Watch, it allows the user to receive emails about changes being made to the repository. This tab allows the user to change whether an email is sent every time a change is made, or a daily email detailing these changes. The default is set to send 'immediately'.

The option to add a watch may only be available if the administrator has enabled watches for the repository.

2.3. Overview

FishEye allows the user to search through the repository to find particular changesets or files.

There are 3 ways to search:

- Quick Search
- Simple Search tab
- Advanced Search

2.3.1. Quick Search

Quick Search is the search box in the top right hand corner of the Fisheye screens.

You can search for:

- Authors
- Branch names
- Commit comments
- Changeset ids
- Filenames/paths - Ant Globs can be used

2.3.2. Simple Search tab

The simple search screen (by clicking on the tab in the top right hand corner of the Fisheyes screens) can be used to retrieve a list of changesets/files using the filters that are available. You can search using 1 or more of the below filters:

- Commit comments
- Contents of files - non-binary files, less then 5mb and on the trunk/head
- Filenames/paths - Ant Globs can be used
- Authors
- Branch names
- Tag names
- Revision Dates Before and After

Results can be grouped by:

- Changeset
- Revision
- File
- Directory

The results are shown in a standard html view. It is possible to show the results in a tabular format by using the Tabular/CSV checkboxes, and if required can be saved to a CSV file. The fields to be shown are:

- Path
- Revision
- Author
- Date
- Comment

- Changeset
- Total lines
- Total lines added
- Total lines removed

2.3.3. Advanced Search

In some circumstances the results of a simple search may not be specific enough. Using the advanced search the user can create their own complex searches using FishEyes powerful query language called EyeQL. To do an advanced search click on the link - **Switch to Advanced Search** found on the top lefthand side of the simple search screen.

TIP: It is possible to flick between Simple to Advanced Search and the EyeQL statement will contain the basics of the statement and then the user can expand on it as required.

3. Administration

3.1. System Requirements

Java Runtime	A JDK or JRE version **1.5** or greater. You can download a Java Runtime for Windows/Linux/Solaris On MacOSX the JVM is available as part of the OS install. here. **Note:** There appeared to be a problem with the first release of the JRockit 5.0 JVM (R25.0.0-75) when using FishEye. If you use the JRockit JVM we recommend you use a later release.
CVS	If you are using CVS, FishEye needs read-access your CVS repository **via the file system** (it does not support protocols such `pserver` at the moment).
Subversion (server)	FishEye can communicate with any repository running Subversion 1.1 or later.
Subversion (client)	FishEye uses the native Subversion client installed on your system. which must be version 1.2 or later and include the JavaHL bindings. Please see **Subversion Client Setup** for more information.
Perforce (client)	FishEye needs access to the `p4` client executable. Due to some problems with earlier versions of the client, we recommend version 2006.2 or later.
Operating System	FishEye is a pure Java application and should run on any platform provided the above requirements are satisfied.

3.1.1. Caveats

- Currently, the `$Log` RCS expansion keyword is not handled correctly by FishEye. Some diff results (and line numbers in diffs) may appear incorrect in files were `$Log` is used.
- When indexing the contents of files, FishEye has an internal limit on the number of tokens/words in the file it can index. Any text past the one-millionth token/word in a file is ignored.

3.1.2. Future SCM support

At this time, FishEye only supports CVS (and CVS-NT), Subversion and Perforce.

Support for other version control systems (such as ClearCase) will be added to FishEye in the future. Let us know what SCM you would like to see supported by posting to the forums

3.2. Quick-Start Guide

This guide will explain how to get FishEye installed and running as easily as possible. For more advanced installation topics, see the **installation guide**.

3.2.1. Install FishEye

1. Download the FishEye zip file and extract it. This document assumes you have extracted FishEye to `/FISHEYE_HOME/`.
2. Ensure you have installed an appropriate Java runtime, see **Requirements**. Ensure that `java` is in the PATH, or that the `JAVA_HOME` environment variable is set.
3. If you intend to use FishEye with Subversion, please ensure you read about the **Requirements**, **Subversion Client setup**, and **granting permission to FishEye** to scan your repository.
4. If you intend to use FishEye with Perforce, please ensure you read about the **Requirements** and **Perforce Client setup**.
5. Copy your `fisheye.license` file to `/FISHEYE_HOME/`. You can download a trial license here.

3.2.2. Run FishEye

> **Note:**
> We recommend you run FishEye as a user that has only read access to your repository.

You can start FishEye immediately with the following:

```
$ cd /FISHEYE_HOME/bin
$ ./run.sh
```

(Use `run.bat` on Windows)

Once started, FishEye will run its own HTTP webserver on port 8080. You can access FishEye immediately by going to `http://HOSTNAME:8080/` in a browser.

> **Note:**
> By default, FishEye will listen on port 8080 for HTTP requests. It also listens on 127.0.0.1:8079 as a control port. You can configure both of these in the admin pages, or by editing `/FISHEYE_HOME/config.xml` and restarting FishEye.

3.2.3. Setup FishEye

The first time you access FishEye from a browser, you will be required to enter an administrator password. This password will give you access to the FishEye admin pages.

Once you have setup an administrator password, you can access the admin pages at `http://HOSTNAME:8080/admin/`. One of the first steps will be to **add a repository**.

3.2.4. Using FishEye

Once you have added a repository, you can view it in FishEye at `http://HOSTNAME:8080/`. FishEye needs to build an index and cache of the contents of your repository, so some information will not appear in FishEye until this is complete.

3.2.5. Stopping FishEye

Use `stop.sh` (or `stop.bat` on Windows) to stop the FishEye server:

```
$ cd /FISHEYE_HOME/bin
$ ./stop.sh
```

3.3. Upgrading FishEye

Before upgrading, you should always read the **changelog**.

The first time you run a new version of FishEye, it will automatically upgrade its data. This may involve a complete re-index of your repository.

Your upgrade procedure depends on whether you are using a separate `FISHEYE_INST` directory. (Read more about `FISHEYE_INST` in the **Install** documentation.)

3.3.1. Using FISHEYE_INST

Simply extract the new FishEye version to a directory, leave your `FISHEYE_INST` environment variable set to its existing location, and start FishEye from the new installation.

You will also need to follow any of the version-specific instructions below.

3.3.2. No separate FISHEYE_INST

You will need to copy some files from your old FishEye installation to your new one.

1. Extract the new FishEye instance into a directory such as `/NEW_FISHEYE/`. Delete the `/NEW_FISHEYE/var` directory.

2. Shutdown the old FishEye instance if it is running.
3. Copy `/OLD_FISHEYE/config.xml` to `/NEW_FISHEYE/`.
4. Copy (or move) the `/OLD_FISHEYE/var` directory to `/NEW_FISHEYE/var`.
5. Copy your `fisheye.license` to `/NEW_FISHEYE/`.
6. Follow any of the version-specific instructions below.

3.3.3. Upgrading 0.x to 1.0

There are some important changes that occurred between 0.10 and 1.0RC1 that you should be aware of when upgrading:

- The FishEye scripts (fisheyectl, start, stop, etc) have been moved from `/FISHEYE_HOME/` to `/FISHEYE_HOME/bin/`
- You can now split part of your FishEye installation into an "instance" directory `FISHEYE_INST`. This makes upgrades much easier.

3.4. FishEye Changelog

NOTE: As of 1.3, FishEye now requires a JVM version 1.5 or later. Previously, 1.4+ was required.

FishEye 1.3 contains many bugfixes and improvements, and adds support for Perforce.

NOTE: Upgrading from 1.2.5 (or earlier) or 1.3beta8 (or earlier) will force a complete re-index of CVS repositories.

For changes prior to 1.3, see the **1.2.x Changelog**.

3.4.1. New features in 1.3

- Support for the Perforce version control system.
- SVN properties are now shown.
- Quicksearch now searches for changeset ids.
- New "mixed" chart on annotation pages, showing author-over-time breakdown.
- Side by Side diffs (1.3.1)

3.4.2. From 1.3.1 to 1.3.2

Bug fixes, changes and improvements

- Fix potential XSS vulnerability in quick-search page.
- Fix problem sending watch emails where the commit message contains a tab character.
- [SVN] Add support for requesting a rescan between given revisions.
- [SVN] Improve scan performance, and better handle add operations from outside

FishEye's view of the repository.
- [SVN] Improve scan performance by not fetchings diffs for binary files.
- [SVN] Timeout settings now configurable via Admin screens.
- [SVN] Display SVN properties at the directory level.
- [P4]] Address problem fetching labels for files.
- Fix Javascript problem in IE when logging into the Admin screens.

3.4.3. From 1.3 to 1.3.1

Bug fixes, changes and improvements

- The truncate diff setting should now work in Internet Explorer
- Fix issue with duplicate paths in tarball generation
- [P4]An empty <p4-config> element would cause an error
- Unknown repos now return a 404 status rather than 500
- [SVN]Handle empty content files when using SvnKit
- [CVS]Allow $ in author names.
- FishEye now uses the tabwidth setting in each user's profile
- [SVN]Fix issue where FishEye incorrectly states that no username was supplied
- Fix IE7 directory spacing problem
- Implement side-by-side diffs

3.4.4. From 1.3beta9 to 1.3

Bug fixes, changes and improvements

- Various improvements when scanning Perforce repositories.
- [SVN] fix for problem with diff hyperlinks to re-added files.
- Fix problem where some paths were not correctly html-escaped.
- Fix "NoSuchFieldError deferredExpression" problem on some platforms (due to a 3rd-party library being included twice).
- Ensure LDAP connections are closed in all situations.

3.4.5. From 1.3beta8 to 1.3beta9

NOTE: Upgrading to 1.3beta9 will force a complete re-index of CVS repositories.

Bug fixes, changes and improvements

- Upgrade JVM requirement to 1.5+.
- Upgrade embedded HTTP engine (Jetty). This fixes some bugs and improves performance under load.
- Fix a performance problem (esp. under load). "Recent Changes" pages should return

much faster now.
- Fix a very slow memory leak when FishEye is under load (for example, when it is being crawled by a web spider).
- Fix a problem where daily-emails would break after a backup was performed.
- [CVS] Fix an error introduced when FishEye builds its repository cache. This requires a full re-scan of CVS repositories.
- [CVS] Fix a problem where FishEye could not parse in RCS files author names that were only numerical digits.
- [CVS] Fix bug when creating tar/zip files from a branch constraint.
- [SVN] FishEye will now timeout long running SVN connections that have blocked.
- [SVN] Fix problem where FishEye was not storing SVN properties correctly.
- [SVN] Fix a bug when entering a revision beyond the current last revision in quick search.

3.4.6. From 1.2.5 to 1.3beta8

Bug fixes and improvements

- [SVN] When importing a repository from a given start revision, you can now nominate if it should import the state of the repository at that revision, or just import changes made after that revision.
- [CVS] Fix a bug where FishEye would send out watch emails for historical changesets after a re-index.
- Performance improvements to changeset page when one of the files in the changeset has a very large history.
- [SVN] Some changes that improve the speed of the initial-scan for some SVN repositories.
- Fix a bug when FishEye generates RSS feed urls constrained by author, when the author has an "@" in their name.
- [SVN] Fix a bug when a tag is deleted (as part of a move).

3.5. Configuration

3.5.1. Installation

This guide details the advanced installation options that can be used when installing Fisheye. For a quick install see the quickstart guide

FishEye Prerequisites

1. Download the FishEye zip file and extract it. This document assumes you have extracted FishEye to `/FISHEYE_HOME/`.

2. Ensure you have installed an appropriate Java runtime, see **Requirements**. Ensure that `java` is in the PATH, or that the `JAVA_HOME` environment variable is set.
3. If you intend to use FishEye with Subversion, please ensure you read about the **Requirements**, **Subversion Client setup**, and **granting permission to FishEye** to scan your repository.
4. Obtain your `fisheye.license` file. You can download a trial license here.
5. **Note:** We recommend you run FishEye as a user that has only read access to your repository.

FishEye Layout

By default, FishEye will run self-contained within the `/FISHEYE_HOME/` directory. The FishEye directory layout looks like this:

`/FISHEYE_HOME/config.xml`	Configuration file.
`/FISHEYE_HOME/fisheye.license`	FishEye license.
`/FISHEYE_HOME/var/`	Directory under which FishEye stores its data.
`/FISHEYE_HOME/var/data/`	Persistent information.
`/FISHEYE_HOME/var/cache/`	Caches and indexes.
`/FISHEYE_HOME/var/log/`	Log files.
`/FISHEYE_HOME/var/tmp/`	Temporary files.
`/FISHEYE_HOME/bin/`	Scripts for controling FishEye.
`/FISHEYE_HOME/lib/`	FishEye's dependant libraries.
`/FISHEYE_HOME/` ...	Remainder omitted for brevity.

However, this self-contained layout results in tedious copying of files each time you upgrade FishEye. Also, if you want to run multiple instances of FishEye, you need multiple `/FISHEYE_HOME/` installations. These two issues can be avoided by setting a `FISHEYE_INST` ("FishEye Instance") location.

Note:

Using a separate `FISHEYE_INST` location is recommended for production installations of FishEye.

When the `FISHEYE_INST` environment variable is set, FishEye's directory layout becomes:

`$FISHEYE_INST/config.xml`	
`$FISHEYE_INST/fisheye.license`	

`$FISHEYE_INST/var/`	All permanent and temporary data is found under `$FISHEYE_INST/var/`
`$FISHEYE_INST/lib/`	Site-specific Java libraries (.jars) that FishEye should load on startup. (Do not copy the dependant `/FISHEYE_HOME/lib/` files into here.)
`/FISHEYE_HOME/lib/`	FishEye's dependant libraries.
`/FISHEYE_HOME/bin/`	
`/FISHEYE_HOME/` ...	Remaining files are found under `/FISHEYE_HOME/`.

The rest of this document will refer to `$FISHEYE_INST/`, but if you have not set `FISHEYE_INST` then it defaults to `/FISHEYE_HOME/` (the directory where you extracted FishEye).

Initial configuration

FishEye runs its own HTTP webserver, and additionally listens on a socket for administration/shutdown commands. These default to `:8080` and `127.0.0.1:8079`, respectively. You can change both these addresses before starting FishEye by editing `config.xml`.

The first time you run FishEye, when you access the FishEye webserver you will be asked to enter an administrator password. This password controls access to the FishEye administration pages.

You can disable FishEye's administration pages by setting `admin-hash=""` in the `<config>` element of `config.xml` before starting FishEye.

If you need to reset the administrator password, delete the `admin-hash` attribute in the `<config>` element. You will be prompted to enter a administrator password next time you start FishEye.

To run FishEye for the first time, simply do the following:

```
$ cd /FISHEYE_HOME/
$ ./run.sh
```

(Use `run.bat` on Windows)

Further Configuration

Once started, FishEye will run its own HTTP webserver (by default on port 8080). You can

access FishEye immediately by going to `http://HOSTNAME:8080/` in a browser.

Once you have setup an administrator password, you can access the admin pages at `http://HOSTNAME:8080/admin/`. One of the first steps will be to **add a repository**.

You will also want to read about the **command line scripts** for controling FishEye.

3.5.2. Adding a repository

Adding a repository to FishEye is a simple matter. Further configuration options are available once a repository is added.

FishEye needs to build an index and cache of your repository. This begins when you first **enable** a repository, and may take some time to complete.

FishEye currently supports the following SCM systems:

- CVS
- Subversion
- Perforce

Common fields

Name	A name for this repository. The name may contain alphanumeric, underscore, "-" or "." characters. Use "cvs" or "svn" if you can't think of a better name.
Description	A short description of this repository.
Enable immediately	Controls whether FishEye will immediately enable this repository, which starts the inital scan. If you wish to do some further configuration before this scan starts, then select "No". You can enable a repository later from the repository list.

CVS

Note:
To add a CVS repository, FishEye must have filesystem access to the repository.

CVS dir	The path to the CVS repository. This is often `/usr/local/cvsroot`. This is a path in the server's filesystem.

Charset	The character set used to interpret and display text files.

Subversion

Note:

When adding a Subversion repository, you should also read about **Subversion Client setup** and **granting permission to FishEye** to scan your repository.

Note:

It is **particularly important** that you correctly setup the correct branch and tag structure for your Subversion repositories. If FishEye does not know which files are tags and branches, it will treat all files as trunk files, which can significantly increase the effective size of your repository. **This will increase initial slurp time and impact runtime performance.** Please refer to the **tag and branch configuration information.**

SVN URL	The Subversion URL to your repository, such has `svn://svn.foo.com/` or `file:///var/svn`
Path	The sub-tree within your repository FishEye should display. If this value is `.` (or empty), then the whole repository will be shown.
Block Size	Controls how many revisions FishEye will pull down from the repository in one batch. Larger values can reduce the time it takes for FishEye to scan your repository for changes, but use more memory. Smaller values can reduce the amount of memory FishEye uses during scans. The default is 400.
Svn Operation Timeout	Sets the timeout value that FishEye imposes on Subversion operations. Operations which exceed this value are terminated. The default for most operations is 1 hour. It can be changed to a different interval, for example to: 2 days, 10 hours, 20 minutes
Throttle connections-per-sec	If set, this allows FishEye to throttle how many connections it makes per-second to the SVN server. Many systems use `inetd`/`xinetd` to service the svnserve protocol. `xinetd` has, by default, an incoming connection limit which can cause FishEye to disrupt other svnserve-based connections. The default is blank (do not throttle).

Charset	The character set used to interpret and display text files.
Access Code	The Access Code for the `fisheye.access` property on the server. See also Subversion fisheye.access.
MD5 Access Code	The MD5 sum of the above Access Code. See also Subversion fisheye.access. (This only appears if Access Code is set.)
Set Access Property Command	The Subversion command to set the `fisheye.access` property to grant FishEye access if nescessary. See also Subversion fisheye.access. (This only appears if Access Code is set.)
Start Revision	If set, the revision number from which FishEye will start indexing the repository. The default is to start scanning from the first revision in the repository.
Initial Import	When a Start Revision is set, this setting controls how FishEye establishes the initial state of the repository. Selecting "Do not import" means that FishEye will only process the revisions from the start revision onwards. The repository state prior to this revision is ignored. If you select "Import without tag information", FishEye will import the repository content as it existed one revision prior to the start revision. FishEye will create a single synthetic revision to hold the initial state. The comment associated with this revision will be "Created by FishEye for initial repository import". Tags created prior to the start revision are ignored.
Username/Password	The credentials to use if your repository requires authentication.
trunk/branch/tag structure	Determines how FishEye attempts to understand the tag and branch structure of your Subversion repository. For more information **read this**.

Perforce

Perforce Host	The name of the server which provides the Perforce repository
Port	The port the server is listening on. This field is optional and FishEye will default to the standard Perforce port (1666) if you do not specify a value here
Path	The path within the Perforce depot that you wish to have FishEye index. You would normally put the depot path here, e.g. //depot/ but you may also use a more specific path to restrict FishEye to a subset of the depot.
Perforce Host	The name of the server which provides the Perforce repository
Block Size	Controls how many changelists FishEye will fetch from the depot in one batch. Larger values can reduce the time it takes for FishEye to scan your repository for changes, but use more memory. The default is 400.
Filelog limit	FishEye uses the p4 filelog command to gather information about the files in changesets. The list of files generated can be very large. Setting a limit here will cause FishEye to batch up filelog operations into groups. This is useful with some versions of the Perforce client which may have trouble with large output. In general you should only set this field if you have a 2005 client or earlier. Lower values will degrade scanning performance.
P4 Operation timeout	Sets the timeout value that FishEye imposes on P4 operations. Operations which exceed this value are terminated. The default for most operations is 10 minutes.
Throttle connections-per-sec	If set, this allows FishEye to throttle how many connections it makes per-second to the Perforce server. The default is blank (do not throttle). You may enter fractional values such as, e.g. 2.5
Charset	The character set used by the server.
Unicode Server	This field indicates whether the Perforce Server

	is running in internationalized mode.
Case Sensitive	This field indicates whether the Perforce Server metadata is case sensitive. You should set this to false for servers running on Windows platforms.
Username/Password	The credentials to use if your repository requires authentication.

3.5.3. Subversion client setup

FishEye can communicate with any *server* running Subversion 1.1 or later, but it needs to use a Subversion *client* to do so. You must configure FishEye to use **one** of the two clients specified below, either the Native **or** the SVNKit client.

If you do not have all the necessary components (see below) of the Native client installed, you may find it easier to use the SVNKit client.

Note:

Using the file:// protocol to access your Subversion repository can be much faster than the other network protocols. We recommend using the file:// protocol if possible.

Native client

FishEye can use a native Subversion client installed on your system, but your client needs to be version 1.2 or later, and **must include the JavaHL bindings**. FishEye can use all of the protocols supported by your native client.

The JavaHL bindings include a Java `.jar` file (typically named `javasvnhl.jar`) and a dynamic library (such as `libsvnjavah-1.so` or `libsvnjavahl-1.dll`). FishEye must be configured so it can find **both** the `.jar` and dynamic library.

If the JavaHL dynamic library is in your "library path" (such as `%PATH%` on Windows), then FishEye will automatically find it. Otherwise you can tell FishEye where it is (with one caveat, see below), or set the `FISHEYE_LIBRARY_PATH` environment variable before starting FishEye.

Pre-compiled native clients are available from the Subversion site.

Native client configuration

You can configure your Subversion client in the **Server Settings** section of the FishEye admin screens, or by editing the <svn-config> section of your `config.xml`. If you change

these settings, you need to restart FishEye.

JAR	The path to the JavaHL `.jar`.
Dynamic library	The path to the dynamic library, if it is not already on your system's library path. Note: due to a bug in earlier versions of the JavaHL bindings, setting this value is ineffective unless you are using a Subversion client 1.2.3 or later.

SVNKit client

If you have difficulty acquiring a native Subversion client which contains the JavaHL bindings, you can try to use SVNKit (which is a 100% Java Subversion library).

Note: prior to SVNKit version 1.1.0, SVNKit was called JavaSVN. We recommend using the 1.1.0 version or later, as it is much improved over earlier releases.

To use SVNKit:

- Disable the native client, by clearing the "JAR" and "Dynamic Library" fields described above (or remove the <svn-config> element from your config.xml file).
- Download SVNKit from the above url.
- Unzip the SVNKit download, and copy all the `.jar` files to `$FISHEYE_INST/lib`

Note:
SVNKit supports the file:// protocol for FSFS repositories only.

SVNKit sometimes has problems working with Subversion servers which are running older versions, such as 1.1.x. If you see exceptions in FishEye's log such as

- SEVERE: assert #B
- Checksum mismatchwhile reading representation:

You will need to either swap to native JavaHL layer or upgrade your subversion server to 1.3 or later.

3.5.4. Subversion fisheye.access

The `fisheye.access` property allows an administrator/commiter to control FishEye access to a directory in the repository. FishEye queries this property to decide whether it will continue to access the repository. If the property does not exist or does not match with that configured in FishEye, FishEye will immediately disconnect from the repository.

Note:
By default, FishEye will have access to your repository.

Setting fisheye:allow

FishEye can operate in one of three accessibility modes:

Mode	Accessibility	Subversion repository property: `fisheye.access`
Allow	any FishEye server	"`allow`" or no property set
Access Code	only FishEye servers configured with the correct Access Code	e.g. "`md5:dc0c08df1f3e80b599c90f53d7dd05ec`
Deny	no FishEye server	"`deny`"

If you would like to restrict FishEye access to your repository, you must set the `fisheye.access` property. This property must be set on the "URL + path" you have configured in FishEye.

Access Code

The repository must be configured with the MD5 sum of the Access Code that is configured in FishEye. The MD5 sum and even the svn command to set the property will be generated for you by FishEye when you configure it using the FishEye Adminstration page. See Adding a repository

For example, if you have configured FishEye with a URL of `svn://foo.com/`, a path of `.` and an Access Code of "fisheye", then you would need to do something like this:

```
$ svn checkout -N svn://foo.com/ tmpworkspace
$ cd tmpworkspace
$ svn propset fisheye.access "md5:4d0c5db8382f80c58e7b0619ae5767a7" .
$ svn commit -m "grant fisheye access" .
```

Deny - deny access to all FishEye instances

To deny all FishEye instances access to the repository, it must be configured with the fisheye.access property of "deny".

For example, if you have configured FishEye with a URL of `svn://foo.com/` and a path of `.` (or you have left path empty), then you would need to do something like this:

```
$ svn checkout -N svn://foo.com/ tmpworkspace
$ cd tmpworkspace
$ svn propset fisheye.access "deny" .
$ svn commit -m "disable fisheye access" .
```

If you configured a path of `some/dir` then use:

```
$ svn checkout -N svn://foo.com/some/dir tmpworkspace
$ cd tmpworkspace
$ svn propset fisheye.access "deny" .
$ svn commit -m "disable fisheye access" .
```

3.5.5. Subversion Symbolic Setup (tag/branch structure)

Since tags and branches in Subversion are implemented via directory copies, they are not really first-class concepts. You can describe what your tag/branch structure looks like, and FishEye will display that information as it would for CVS. These settings can be edited in the "Add Repository" or "Edit Repository" pages in the FishEye Admin screens.

For more information on tag/branch layout, see Repository Layout in the Subversion documentation.

The symbolic setup tells FishEye how to classify each path it encounters in the repository. Each path is classified as either a trunk, branch, tag or root path. The root category is used when a path does not match any of the given trunk/branch/tag settings and is mostly treated in the same way as trunk paths.

Note:

The symbolic settings do not exclude any paths from consideration by FishEye. To exclude paths you should set up appropriate "allow" rules. If your symbolic setup does not match a path, that path will be classified as a root path and processed by FishEye accordingly.

Note:

If you change these trunk/branch/tag settings, you will need to do a complete re-scan of the repository. You can do this from the Maintenance section.

Common layouts

There are two common repository layouts that you can choose from in FishEye. These layouts are described in Repository Layout.

The first is where there are top level `trunk`, `branches` and `tags` directories. This is called "/trunk/..., /branches/NAME/..., /tags/NAME/..." in FishEye.

The second is where the `trunk`, `branches` and `tags` directories are one level down, under each top-level project directory. This is called "/project/trunk/..., /project/branches/NAME/..., /project/tags/NAME/..." in FishEye.

Custom layouts

You can describe to FishEye any custom tag/branch structure you have. If you want to use

one of the common layouts as a basis, first select it from the dropdown, then select "Custom" to edit/add rules.

When looking at a file on a branch, or a file that was tagged, FishEye needs to determine a "name" for the branch/tag. FishEye does this by matching a regular expression against the file's path, and extracting the name based upon the match. FishEye also needs a "name" for files on the trunk (in effect, this is the name of the trunk "branch").

For any file that matches a trunk/branch/tag regular expression, a "logical path" is calculated. Two different files with the same "logical path" are considered to be related. For example, using the second type of common repository layout:

- The file `project1/trunk/dir1/foo.txt` would have a logical path of `project1/dir1/foo.txt`.
- The file `project1/tags/BUILD123/dir1/foo.txt` would have a logical path of `project1/dir1/foo.txt`, and the name of the tag would be `project1-BUILD123`.
- Both these files have the same logical path, and so are considered related. By looking at what revision the directory-copy for `project1/tags/BUILD123/dir1/foo.txt` occurred, FishEye can determine to what revision the tag `project1-BUILD123` applies.

You can add as many rules as you need, for any given file the first rule that matches is used.

Regex	The regular expression used to match against the start of the path. The trailing part of the path that does not match the regex is called the "tail".
Name	An expression used to extract a tag or branch "name" from the regex.
Logical Path Prefix	This is an expression used to construct the logical path. The logical path is the concatenation of the result of this expression, and the "tail" of the regex.

3.5.6. Perforce client setup

FishEye can communicate with any Perforce *server*, but it needs to use the `p4` command-line *client* to do so.

By default, FishEye looks for the `p4` executable in the current path. You can specify the exact path the the `p4` executable you wish to use in the "Server Settings" in the FishEye Admin screens.

Some users have reported errors where FishEye considers some files to be binary when they are not. It appears this may be a limitation of earlier P4 clients. If you can upgrade to a recent P4 client (2006.1 onwards), this will fix this issue. You do not need to update the P4 Server. If you are unable to update, the admin UI allows you to set a limit on the size of filelog commands sent to the server. Setting this to something around 100 will fix the issue. It will, however, also impact performance significantly.

3.5.7. Repository Settings

FishEye has various configuration options for each repository. To access these, click on the name of the repository in the Admin Menu or by clicking on 'view' in the repository list within Admin. To change settings that will affect all repositories, choose Repository Defaults from the Admin menu instead.

Repository management

Note:

Some setting changes will require the repository to be restarted, while others will require the repository to be re-indexed. FishEye will advise you if this is the case when you make the change. You can do this from the Repository List.

Indexer

The following actions can be triggered manually by the Administrator.

Full scan (CVS only)	Scan the whole repository for any changes since the last scan.
Re-index Repository	Deletes the current cache and re-indexes the repository from the beginning.
Rescan Revision Properties (SVN only)	In Subversion it is possible to allow non-versioned properties to be updated by committers (for example: the check-in comment). When this happens, FishEye will not automatically pick up the updates. By rescaning specific revisions, FishEye will rescan the non-versioned properties and amend the entry in FishEye accordingly.

Updater (CVS)

FishEye will monitor your CVS "history" file (`CVSROOT/history`) to determine what in your repository has changed. FishEye will also periodically scan the whole repository.

CVS is not always configured to create a history file. Talk to your CVS administrator.

The default values should be fine for most repositories. (Leave a value blank to use the default value.)

History file	The location the CVS history file. If relative, then it is relative to the CVS directory specified for this repository. Defaults to `./CVSROOT/history`.
Full scan period	How often FishEye will do a full scan of the repository. Defaults to 15 minutes. Specify using an interval, such as "15 min", "2 hours", etc. A value of "0" disables the periodic full-scan (you can still use `fisheyectl fullscan` to cause a full-scan to occur).
Strip prefix	Prefix to strip off files found in the history file, to make them relative to this repository's CVS directory. Necessary if the CVS directory specified is not the "root" of the CVS repository. For example, your CVS is located at `/usr/local/cvsroot`, but you specified `/usr/local/cvsroot/foo/bar` as the CVS directory of this repository. You will need to give the history file as `../../CVSROOT/history` and set a strip prefix of `foo/bar`.

Updater (SVN)

Poll Period	How often FishEye will check if there have been any new commits into the SVN repository. The default is 60 seconds. It is possible to set the period by units, for example: 10second, 1week. Valid units are "second", "minute", "hour", "day", "week", "month", "year". The default unit is days if only a number is added.

Linkers

FishEye can detect special substrings in commit messages, and hyperlink those substrings to other systems. This is particularly useful if you use an issue tracking system, and put the issue identifiers into your commit messages.

Any linkers defined in the repository defaults are added to each individual repository.

Here are some example simple linkers:

- Regex: `[a-zA-Z]{2,}-\d+`
  Href: `http://jirahost:8080/browse/${0}`
  Link any occurrence of a Jira-style issue to Jira.
- Regex: `^BUG: (\d+)`
  Href: `http://bugzilla/bugzilla/show_bug.cgi?id=${1}`
  Links bug numbers that occur at the start of a line to Bugzilla.

Permissions

Anonymous access to FishEye is allowed by default. You can disable anonymous access at a global and per-repository level. See here for more information about how Fisheye handles security and users.

Watches

FishEye has a watch notification system that allows users to receive email notifications when commits are detected. This functionality can be disabled on a per-repository basis.

> **Note:**
> Watch functionality requires a valid SMTP server to be configured.

Allow (process)

By default, FishEye will cache and index your whole repository, and presents all of this information to users. You can control what parts of your repository FishEye will access in the **Allow** section.

Includes defines what subtrees of your repository fisheye will access. It defaults to "everything". If you specify some include directories, then only those directories (and all their subdirectories) will be included by FishEye.

Excludes allows you to specifically exclude files and directories that may have been included. Each exclude is is an Antglob. Examples:

- `/CVSROOT/**` (or just `/CVSROOT/`): excludes `/CVSROOT` and all its sub-children.
- `*.OBJ`: excludes any OBJ files.

> **Note:**
> Changes to Includes & Excludes do not take effect until a full re-slurp of your repository is performed.

Hidden Dirs

You can mark unused(deprecated) directories as "hidden", so that they do not appear in the FishEye user interface unless the user has specifically toggled "Show hidden directories". FishEye will still index and cache these directories.

This can be useful if have old directories that you don't want cluttering the UI by default.

Tarball Settings

FishEye contains a feature that will build an archive of a directory tree. This feature is disabled by default. Tarball settings in the Repository Defaults can be over-ridden on a per-repository basis.

You can set a limit on the number of files that a tarball can contain.

You can selectively disable tarballs creation for certain directories, or directory trees.

Properties

These properties allow you to customize the behavior of FishEye specifically removing the graph and calendars from certain screens.

3.5.8. Properties

Overview

Properties allow you to customize the behavior of FishEye. A property may be set either per repository, or globally as a repository default. A repository default property, is inherited by all repositories. A default property may be overridden at the repository level.

The following properties are supported:

Name	Possible Values	Default Value	Description
`show-changelog-calendar`	`true`, `false`	`false`	If set to false, the calendar is disabled on the changelog page. This may be required for performance reasons. The revision totals displayed per calendar day, month and year may be expensive to calculate.

			For repositories with a lot of historical data, disabling the calendar can result in significant performance improvements when viewing the changelog page.
`enable-line-history`	`true`, `false`	`true`	Allows you to disable (hide) the line-count history graph on the Browse and Changelog pages. This may be desirable if you have a large repository and generating the linegraphs takes a long time.

3.5.9. The FishEye Web Server

Note:
You need to restart FishEye for any changes to these settings to take effect.

HTTP Bind	The address the FishEye webserver will bind to. Can be just a port number, or an address and port number. If no host is specified, then FishEye will bind to all available interfaces. Examples are: `:8080`, `hostname:8080`, `10.0.0.11:80`. At least one of 'AJP13 Bind' or 'HTTP Bind' must be set.
Web context	By default, the FishEye application can be accessed via `http://HOST:PORT/` (where HOST and PORT are defined as above). You can force the FishEye application to be hosted on a different "context" or "path" by specifying a value here. For example, if you specify a web context of "fisheye" then FishEye will be accessable from `http://HOST:PORT/fisheye/` instead of `http://HOST:PORT/`.
Proxy scheme	Can be set if you are forwarding through to FishEye from another webserver.

Proxy host	Can be set if you are forwarding through to FishEye from another webserver.
Proxy port	Can be set if you are forwarding through to FishEye from another webserver.
AJP13 Bind	The bind address for ajpv13. If no host is specified, then FishEye will bind to all available interfaces. Examples are: `:8009`, `hostname:8009`, `10.0.0.11:8009`. At least one of 'AJP13 Bind' or 'HTTP Bind' must be set.
Remote API	Enables/Disables the FishEye's Remote API. Clicking on the help link will take you to the API doc.
Server timezone	The timezone to use within FishEye. This timezone is used when displaying dates, and parsing EyeQL date expressions. If blank, then the timezone of the underlying host is used.
Site URL	This is the base URL for this FishEye instance. If not specified, FishEye will attempt to determine this value.

See **this page** for more information on **Subversion Client settings**.

3.5.10. SMTP Settings

Configuring SMTP

To configure SMTP, go to the Server Settings section of the FishEye Admin. The following parameters can be entered:

From Address	The from email address used when FishEye sends an email. e.g. `fisheye-noreply@example.com`
SMTP hostname	The hostname of the SMTP server.
Enable debug logging	*Optional.* Turn this on to enable debug logging from the mail server, useful in tracking down mail server connectivity problems.
SMTP host port	*Optional, defaults to 25.* The port to connect to on the SMTP host.
Username & Password	*Optional.* Username and Password for

	authenticated SMTP access.

Once you have configured SMTP, you can use the "Send Test Email" link on the Server Settings page to send a test email from FishEye to confirm the SMTP connectivity is functioning correctly.

3.5.11. Users/Security

Users and Security

Overview

You can implement access-control using FishEye's user list. FishEye can maintain a set of users, or you can have FishEye look in an *external authentication source* for users, passwords and permissions.

Note:

FishEye provides a pluggable architecture to allow arbitrary forms of Authorization and Authentication to be used.

Anonymous access to FishEye is allowed by default. You can disable anonymous access at a global and per-repository level. To change go to Admin -> Global Settings -> Users/Security.

External Authentication sources

Although FishEye always maintains a list of users internally, you can have FishEye authenticate and authorize users against an external authentication source.

FishEye currently supports:

- **LDAP authentication**.
- **Host-based authentication**. This is implemented using PAM on Linux/Solaris/OS-X, and Local/Domain Accounts on Windows.
- **AJPv13 authentication**.
- **Custom authentication**.

To set one of the above authentications, go to Admin -> Global Settings -> Users/Security. Only one authentication can be set at one time. However each repository can have the option as to whether to use the authentication or not.

To change authentications you will need to remove the settings that are already configured. Just click on the "Remove" link. You will then be presented with the option to add a different authentication.

LDAP Authentication

Global Settings

Global LDAP settings are:

URL	The URL of the ldap server, e.g. `ldap://localhost:389`.
Base DN	The base search space for users, e.g. `dc=example,dc=com`
User Filter	The LDAP search for locating users, e.g. `uid=${USERNAME}`. The `${USERNAME}` variable is expanded to the username of the individual being authenticated. You can use a more complicated LDAP filter to only allow a subset of users, such as: `(&(uid=${USERNAME})(group=fisheye))`.
UID Attribute	The name of the username attribute in objects matching the filter.
Email attribute	(optional) The name of an attribute giving the user's email address.
Cache TTL (positive)	How long FishEye should cache permission checks. Example values are: `0 secs`, `5 mins`.
Auto-add	FishEye can automatically create a user it has not previously encountered if the user can successfully authenticate against LDAP.
Initial bind DN and password	(optional) If your LDAP server does not allow anonymous bind, then you need to specify a user FishEye can use to do its initial bind.

Per-repository Settings

You can give FishEye an LDAP filter that will be used to check if a user has access to individual repositories. You can specify this per-repository, or just specify it in the repository-defaults:

LDAP restriction	An LDAP filter used to check if a given user can access a given repository, e.g. `(&(uid=${USERNAME})(group=${REP}))`. The `${REP}` variable is replaced with the name

	of the repository in question.
Match Type	One of 'user' (default) or 'any'. This setting modifies the meaning of LDAP restriction. If set to 'user', then FishEye expects the filter to match the exact DN of the current user. If it does match, then the user has access to the repository. Commonly, if your user object contains the list of groups the user has access to, then you would use a 'user' match. If set to 'any', then the filter just needs to match one result for the user to have access to the repository. Commonly, if your group object contains the list of uid members, then you would use an 'any' match. In such a case, your LDAP restriction filter may look like: `(&(uniquemember=${USERNAME})(dn=${REP},ou=groups,`. That is, return the group of which the current user is a member.

Active Directory

To have FishEye connect to an Active Directory server, use settings such as the following:

URL	`ldap://HOSTNAME:389`
Base DN	`DC=corp,DC=example,DC=com`
User Filter	`sAMAccountName=${USERNAME}`
UID Attribute	`sAMAccountName`
Email attribute	`mail`
Initial bind DN	`corp.example.com/Users/SomeUser`

Host-based Authentication

Host-based Authentication uses the user account mechanism of the underlying operating system on which FishEye is running. FishEye currently supports PAM-based authentication on Linux/Solaris/OS-X, and NT-based authentication on Windows.

For more details on configuring Host-based authentication on your operating system, see further below.

Group restrictions

FishEye can be configured to check if a user belongs to a group (or groups) before allowing access. You can list one group name, or join several group names into a boolean expression like `group1 & (group2 | group3)`.

If your group name contains spaces or non-ASCII characters, then you need to use quotes. For example: `"Power Users" | Administrators`.

Windows

Note:

If you are using Active Directory, you can configure FishEye to use LDAP as an alternative to using host-based authentication.

If the computer FishEye is running on is **not** a member of a domain, then then Domain attribute is ignored.

When the computer is a member of a domain, you need to enter the full DNS name of the domain. For example, `corp.example.com`. If you enter the short version of the domain (e.g. `corp`), then group-based restrictions may fail.

Once you have configured your settings, we recommend you use the Test function to ensure your access-control performs as you expect.

PAM

On Linux, Solaris and OS-X, host-based authentication uses PAM (Pluggable Authentication Modules) to check users' passwords.

FishEye needs to be configured with the *service name* to use when conversing with PAM. You can create a new service name in the PAM configuration (typically `/etc/pam.conf` or `/etc/pam.d/`), or configure FishEye to use an existing service name (such as `other`, `login` or `xscreensaver`).

Some general operating-system specific tips are given below, but you should consult the PAM documentation for your operating system.

Once you have configured your settings, we recommend you use the Test function to ensure your access-control performs as you expect.

Linux

On many Linux distributions, you may need to create a `/etc/pam.d/fisheye` file containing:

```
auth       required     pam_stack.so service=system-auth
```

Mac OS-X

On a default OS-X installation, you may need to create a `/etc/pam.d/fisheye` file containing:

```
auth       sufficient     pam_securityserver.so
auth       required       pam_deny.so
```

Solaris

If your are using the default `pam_unix_auth` PAM configuration on Solaris, then you may need to add a line like this to your `/etc/pam.conf` file:

```
fisheye auth requisite          pam_authtok_get.so.1
fisheye auth required           pam_unix_auth.so.1
```

If you test this and it does not work, it is probably because when using `pam_unix_auth` on Solaris, the process doing the password check needs read access to `/etc/shadow`. Giving the FishEye process read access to this file may solve this problem, but using permissions other than `0400` for `/etc/shadow` is not recommended. You should discuss this with your system administrators first, and possibly change to a PAM module other than `pam_unix_auth`.

Global Settings

Global settings are:

Domain/Service name	**Windows:** the name of the domain. Leave blank to use the local computer. **PAM:** the service name in your PAM configuration to use. If blank, `fisheye` is used.
Required group:	The group or groups a user must belong to in order for them to be able to login.
Cache TTL (positive)	How long FishEye should cache permission checks. Example values are: `0 secs`, `5 mins`.
Auto-add	FishEye can automatically create a user it has not previously encountered if the user can successfully authenticate with the host.

Per-repository Settings

You can give FishEye an group restriction that will be used to check if a user has access to individual repositories. You can specify this per-repository, or just specify it in the repository-defaults:

Required Group	A group (or groups) used to check if a given user can access a given repository. For example: `cvsusers & cvs${REP}` The `${REP}` variable is replaced with the name of the repository in question.

AJPv13 Authentication

Overview

AJP Authentication expects that requests are pre-authenticated via an external server before arriving at FishEye.

Typically, this would be a web server (e.g. apache) configured to perform password and role checking for a given URL. If successful the server forwards the request to the FishEye server via the AJPv13 protocol.

FishEye Configuration

For FishEye to use AJP authentication the following two values must be configured:

- The AJP Bind Address must be set per FishEye instance. See also Server Settings
- The users Auth Type must be set to 'ajp'.

Apache Configuration

Here is one example of how to configure Apache Httpd so that all requests to Apache Httpd for the path `/fisheye` are forwarded to a FishEye instance on the same machine with an AJP Bind Address of `localhost:8009`.

Add these lines to your apache configuration:

```
LoadModule jk_module modules/mod_jk.so

JkWorkersFile /path/to/workers.properties
JkLogFile /var/log/mod_jk.log
JkLogLevel debug
JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "
JkMount /fisheye/* worker1
```

Then create a file under `/path/to/workers.properties` and add:

```
worker.list=worker1
worker.worker1.type=ajp13
```

```
worker.worker1.host=localhost
worker.worker1.port=8009
```

Custom Authentication

Overview

To implement an arbitrary form of Authentication and Authorization for FishEye you need to provide a class which extends `com.cenqua.fisheye.user.plugin.AbstractFishEyeAuthenticator` . More information regarding custom FishEye Authorization can be found in the JavaDoc or the Zip archive. .

For FishEye to use the Authenticator, it must be compiled, placed in a jar archive and then put in the `$FISHEYE_INST/lib` directory. If other 3rd party libraries are required by your Authenticator, they must also be in the `$FISHEYE_INST/lib` directory.

Global Configuration

After implementing a custom Authenticator, the next step is to configure FishEye to use it. Click the "Setup Custom authentication" link on the "Users/Security" page. You will be presented with a form containing the following fields to be set:

Classname	The fully qualified classname of your AbstractFishEyeAuthenticator. e.g. `com.cenqua.fisheye.user.plugin.ExampleFishEyeAuth` .
Cache TTL (positive)	How long FishEye should cache permission checks. Example values are: `0 secs` , `5 mins` .
Auto-add	FishEye can automatically create a user it has not previously encountered if the user can successfully authenticate against your Authenticator.

Properties	Any properties your Authenticator requires. These will be passed to it's `init()` method. This field should comply with the java.util.Properties format. e.g. `# comments`
	`name1=value1`
	`name2=value2`

Per-Repository Constraint Configuration

You may also require a per-repository constraint to restrict access to specific repositories using your custom authenticator. If a custom authenticator is set, then the Permissions Summary table will display the constraint per repository and a link to enable you to edit it.

Note:
The Authentication Test page allows you to enter a user's credentials and to test the user's authentication. It will also test which repositories the user is authorized to access.

3.5.12. Backing up and Restoring FishEye Configuration Data.

Overview

A zip archive of all FishEye configuration files can be created via the FishEye Admin interface or by using the `fisheyectl` script.

The FishEye backup and restore procedure requires you to use the `FISHEYE_INST` system variable. (Read more about `FISHEYE_INST` in the **Install** documentation.)

A backup and restore allows you to move your fisheye instance to another location or host. It also allows you to upgrade to another version of fisheye without losing any configuation or user data.

Backup

The following files will be backed up:

- `config.xml`
- `fisheye.license`
- `var/data/data0.bin`

Note:
No repository cache data will be backed up.

Backup via the Admin Web UI

The "Create Archive" button creates a .zip file in the `$FISHEYE_INST/backup` directory.

Backup via the Command line

The `fisheyectl` script takes a `backup` command and an optional `filename` for the backup archive. see: fisheyectl backup

Restore

To restore a backup, stop the FishEye server and then unzip the file created above into the `$FISHEYE_INST` directory. For example, say you have a `backup_20060101120000.zip` in `/tmp` and you have stopped FishEye, the restore procedure would be something like:

```
$ cd $FISHEYE_INST
$ unzip /tmp/backup_20060101120000.zip
```

3.5.13. Command-line Options

A FishEye instance can be managed using the `fisheyectl` script. Before running this script you either need to ensure you have set the `JAVA_HOME` environment variable, or that `java` is on the path.

Unix usage:

```
/FISHEYE_HOME/bin/fisheyectl.sh command [options]
```

Windows usage:

```
\FISHEYE_HOME\bin\fisheyectl.bat command [options]
```

The *command* parameter can be one of `run`, `start` or `stop` (see below). You can also find convenience scripts for running each of these commands (for example, `run.sh` or `run.bat`).

run

The `run` command starts FishEye. This command runs FishEye in the foreground (it does not fork a background process).

Options:

`--config path`	Load configuration from the file at `path`. Default is `$FISHEYE_INST/config.xml`.
`--quiet`	Do not print anything to the console.
`--debug`	Print extra information to the debug log.
`--debug-perf`	Print performance related information to the debug log.

The following options can be used, but will be removed at a later date:

`--Xtab-width nchars`	Specifies the number of spaces to use to represent a tab character. The default is 8.

`--Xdisable-dirtree-empty-checks`	When rendering the directory tree on some pages, FishEye calculates if each directory subtree is "empty". For massive repositories, this calculation can cause the page to take a long time to render. This option disables the calculation that determines emptiness.
`--Xdisable-content-indexing`	Disable the generation of a full-text index for file content. This prevents further indexing, but does not delete any existing full-text indexes. FishEye will not warn you if you specify this option but still try to do a content-search. This option is useful if you do not use content-search and you are finding FishEye is taking a long time to index your content.

start

This command has the same options as `run`, but starts FishEye in the background.

Windows: FishEye will be run in a separate `cmd.exe` window.

Unix: FishEye will be run with `nohup`, and the console output will be redirected to `$FISHEYE_INST/var/log/fisheye.out`.

stop

The `stop` command stops a running FishEye instance.

Options:

`--config path`	Load configuration from the file at `path`. Default is `$FISHEYE_INST/config.xml`.

fullscan

Usage:

```
fisheyectl fullscan [options] [repname ...]
```

Requests a full-scan of the given repositories, or all repositories if no repository name is given

Options:

`--config path`	Load configuration from the file at `path`. Default is `$FISHEYE_INST/config.xml`.

backup

Usage:

```
fisheyectl backup [filename]
```

Creates a zip archive containing important FishEye configuration files.

Options:

`filename`	Store the backup.zip to `filename`. Default is `$FISHEYE_INST/backup/backup_yyyyddMMHHmmss.zip`.

3.5.14. Environment Variables

JAVA_HOME

The JAVA_HOME environment variable is used by FishEye to select the Java Virtual machine to be used to run FishEye. If this environment variable is not set, FishEye will use whatever Java executable is available on the path. In Linux systems, this may sometimes be the gcj based which has some problems running FishEye.

FISHEYE_OPTS

FishEye uses the FISHEYE_OPTS environment variable to pass parameters to the Java Virtual Machine (JVM) used to run FishEye. This is typically used to set the Java heap size available to FishEye. WIth a Sun JVM, for example, you would use:

```
FISHEYE_OPTS=-Xmx256m
```

This would give FishEye a 256 MByte heap.

It is possible to put other JVM options into the FISHEYE_OPTS environment variable. For example, the -Xrs options should be used if running FishEye as a service under Windows to prevent the JVM closing when an interactive user logs out.

FISHEYE_ARGS

FISHEYE_ARGS are the arguments which will be passed to FishEye when it is started. You can set this to --debug, for example, if you always want to have FishEye debugging put into the FishEye log files

FISHEYE_LIBRARY_PATH

The FISHEYE_LIBRARY_PATH environment variable tells FishEye where it should look to load any additional native libraries

FISHEYE_HOME

FISHEYE_HOME is the location of the FishEye application. By default FishEye will set this to the directory above the fisheyectl script

FISHEYE_INST

The FISHEYE_INST variable tells FishEye where to store its data. If you wish to separate FishEye's data from it application files in FISHEYE_HOME, you should use this variable.

3.5.15. Tuning FishEye

Java Heap Size

The heap size of the FishEye Java Virtual Machine is controlled by the FISHEYE_OPTS environment variable. The best heap size to use is dependent on a number of factors including:

- The SCM system being used. Subversion scanning typically uses more memory than CVS, for example.
- The complexity of operations in the repository. Processing changesets which affect many files will use more memory
- The amount of physical RAM in the system. If the Java heap is too large, it may induce swapping which will impact performance

FishEye will reserve a portion of the available heap for caching of database data, so, in general, the more memory you can supply, the better.

If you do run into OutOfMemory errors, you will need to increase the heap size and restart FishEye

For Subversion respositories, it is also possible to reduce FishEye's memory footprint by reducing the BlockSize parameter

3.5.16. Integration with other web servers

FishEye has a built-in web server, but commonly runs in an environment that has its own web server. You can easily proxy-through to FishEye from this primary web server, so that it appears FishEye is part of the primary web server.

In most situations, FishEye can determine the host and port of the primary web server automatically. This is usefull when you have multiple virtual-hosts proxied through to the one FishEye instance.

If it appears FishEye is having trouble automatically detecting the primary web server's host and port, you will need to set the **Proxy host** and **Proxy port** parameters. If the primary web server is running on `WEBHOST:80` and FishEye is running on `FEHOST:8080`, then you can set FishEye's **Proxy host** and **Proxy port** parameters to `WEBHOST` and `80`.

If the primary web server is using SSL, then you should set **Proxy scheme** to `https`.

You will probably want FishEye to appear in a "subdirectory" of the primary server. In that case, you need to set FishEye's **web context** parameter. The rest of the page assumes you have set this value to `fisheye`.

Then configure your primary web server as follows.

Apache

The easiest way to proxy through to FishEye is using the `ProxyPass` directive (which requires the `mod_proxy` module). Add this section to your Apache configuration:

```
ProxyPass /fisheye  http://FEHOST:8080/fisheye
```

If you want Apache to serve FishEye's static content, then you can do something like this instead:

```
<Directory "/FISHEYE_HOME/content/static" >
    Allow from all
    AllowOverride None
</Directory>
Alias /fisheye/static /FISHEYE_HOME/content/static
ProxyPass /fisheye/static/ !
ProxyPass /fisheye  http://FEHOST:8080/fisheye
```

Note:

An alternative to using `ProxyPass` is to use `mod_rewrite` with the `[P]` flag.

AJP

FishEye also supports AJPv13 connectivity. For more information, please see ajp13.

3.5.17. ViewCVS URL Compatibility

FishEye contains a URL-compatibility mode with the ViewCVS and CVSWeb tools. For example, a ViewCVS URL of the form `http://host/viewcvs.cgi/x/y/z` can be

viewed in FishEye at `http://fisheyehost/viewcvs/x/y/z`.

FishEye can be configured as to exactly how it provides this compatibility mode. In particular, you can configure how to map ViewCVS repository names (`cvsroot` or `root` in the query parameter) to FishEye repository names.

The **Default Mapping** can be used to configure which repository to use if no repository is specified in the URL. If a repository name is given in the URL, you can tell FishEye how to translate that to the name of a FishEye repository. Otherwise, FishEye will attempt to use the repository name given in the URL directly.

3.5.18. Custom Content

FishEye Enterprise License users have access to the HTML/JSP source of FishEye and can customize FishEye's look and feel.

FishEye source edition

To use custom HTML/JSP content, you must be using a build of FishEye that contains the JSP source. These builds are named `fisheye-1.x.y-jspsource.zip` instead of the normal `fisheye-1.x.y.zip` bundle.

If you have a FishEye Enterprise License and need access to the JSP source build, please send an email to FishEye support.

Customizing Content

You can modify any of the files in `FISHEYE_HOME/content/`. However we strongly recommend you use separate `FISHEYE_HOME` and `FISHEYE_INST` directories (as described **here**), and that you store your modified files in `FISHEYE_INST/content` instead.

If you use `FISHEYE_INST/content`, you only need to keep in there your modified JSP/HTML files. This has the advantage of simplifying upgrades.

When you make changes to content, your changes should appear when you next refresh the page in your browser. If they do not then login to the FishEye Admin screens, go to the Content page and follow the instructions there.

4. Reference

4.1. Antglobs

FishEye supports a powerful type of regular expression for matching files and directories (same as the pattern matching in Apache Ant). These expressions use the following wildcards:

?
Matches one character (any character) (not including path separators)
*****
Matches zero or more characters (not including path separators)
******
Matches zero or more *path segments*

Remember that Ant globs match *paths*, not just simple filenames. If the pattern does not start with a path separator (a / or \), then the pattern is considered to start with `/**/`. If the pattern ends in a `/` then `**` is automatically appended. A pattern can contain any number of wildcards. (Also see the Ant documentation.)

4.1.1. Examples

***.txt**
Matches `/foo.txt`, `/bar/foo.txt`; but not `/foo.txty`, `/bar/foo.txty/`.
/*.txt
Matches `/foo.txt`; but not `/bar/foo.txt`.
dir1/file.txt
Matches `/dir1/file.txt`, `/dir3/dir1/file.txt`, `/dir3/dir2/dir1/file.txt`.
****/dir1/file.txt**
Same as above.
//dir1/file.txt**
Same as above.
/dir3//dir1/file.txt**
Matches `/dir3/dir1/file.txt`, `/dir3/dir2/dir1/file.txt`; but not `/dir3/file.txt`, `/dir1/file.txt`,
/dir1/**
Matches all files under `/dir1/`.

4.2. Date Expressions

FishEye supports a wide variety of date expressions. A date has the general form of either `DATE[+-]TIMEZONE[+-]DURATION` or `DATECONSTANT[+-]DURATION`. The `TIMEZONE` and `DURATION` parts are both optional.

`TIMEZONE` can be an offset from GMT `HHMM` or `HH:MM`, or simply the letter `Z` to denote GMT. If no timezone is given, then the FishEye server's configured timezone is used.

`DATE` can be either of the following:

`YYYY-MM-DDThh:mm:ss`
Specifies a time and date (separated by a `T`). The seconds part may contain a fractional component. A `/` can be used instead of `-` as a separator.
`YYYY-MM-DD`
Specifies 00:00:00 on the given date. A `/` can be used instead of `-` as a separator.

`DATECONSTANT` can be any of:

`now`
This very instant (at the time the expression was evaluated).
`today`
`todaygmt`
The instant at 00:00:00 today (server-time* or GMT)
`thisweek`
`thisweekgmt`
The instant at 00:00:00 on the first day of this week (Sunday is considered the first day) (server-time* or GMT)
`thismonth`
`thismonthgmt`
The instant at 00:00:00 on the first day of this month (server-time* or GMT)
`thisyear`
`thisyeargmt`
The instant at 00:00:00 on the first day of this year (server-time* or GMT)

* The timezone used for server-time is part of the FishEye configuration

The syntax for `DURATION` is similar to the XML Schema duration type. It has the general form `PnYnMnDTnHnMnS`. See Section 3.2.6 of the XML Schema Datatypes document for more details.

4.2.1. Examples

`2005-01-02`
The start of the day on January 1, 2005 (server's timezone)

2005-01-02-0500
The start of the day on January 1, 2005 at GMT offset -0500 (New York)
2005-01-02T12:00:00Z
Midday, January 1, 2005 GMT
today-P1D
Yesterday (start of day)
today+P1D
Start of tomorrow
thismoth-P1M
Start of last month
thisyear+P1Y
Start of next year
now-PT1H
One hour ago
now+PT1H2M3S
One hour, two minutes and three seconds from now.

4.3. EyeQL

FishEye contains a powerful query language called EyeQL. EyeQL is an intuitive SQL-like language that gives you the ability to craft arbitrary queries.

EyeQL allows complex searches to be done either within the Advanced Search or they can be incorporated within scripts from the FishEye API.

***query*:**
select revisions
(**from** (**dir**|**directory**) *word*)?
(**where** *clauses*)?
(**order by date**)?
(**group by** (**file**|**dir**|**directory**|**changeset**))?
(**return** *return-clauses*)?
***clauses*:**
clause ((**or**|**and**|**,**) *clause*)*
Notes: "and" binds more tightly than "or". "," means "and"
***clause*:**

(*clauses***)**

not *clause*

path (**not**)? **like** *word*
Notes: *word* is an Ant-glob.

date in (**(** | **[**) *dateExp*, *dateExp* (**)** | **]**)
Notes: The edges are inclusive if **[** or **]** are used, and exclusive if **(** or **)** is used.

date *dateop dateExp*
Notes: *dateop* can be <, >, <=, >=, =, == or != .

author = *word*

author in (*word-list***)**

comment matches *word*
Notes: does a full-text search

comment = *string*
Notes: matches *string* exactly. Most comments end in a newline, remember to add \n at the end of your string.

comment =~ *string*
Notes: *string* is a regular expression

content matches *word*
Notes: does a full-text search, but at this time searches are restricted to HEAD revisions.

(**modified** | **added** | **deleted**)? **on branch** *word*
Notes: Selects all revisions on a branch.
modified excludes the branch-point of a branch.
added selects all revisions on the branch if any revision was added on the branch.
deleted selects all revisions on the branch if any revision was deleted on the branch.

tagged *op*? *word*
Notes: *op* can be <, >, <=, >=, =, == or != and defaults to == if omitted. These operators are "positional" and select revisions that appear on, after, and/or before the given tag.

between tags *tag-range*

after tag *word*

before tag *word*

is head (**on** *word*)?
Notes: this selects the top-most revision on any branch, if a branch is not specified

is (**dead** | **deleted**)
Notes: means the revision was removed/deleted.

is added

Notes: means the revision was added (or re-added).

***tag-range*:**
(**(** | **[**) T1:*word* **,** T2:*word* (**)** | **]**)
A range of revisions between those tagged T1 and T2. The edges are inclusive if **[** or **]** are used, and exclusive if **(** or **)** is used. You can mix edge types, these are all valid: `(T1,T2)`, `[T1,T2]`, `(T1,T2]` and `[T1,T2)`.
***word*:**
any *string*, or any non-quoted word (that does not contain whitespace or any other separators)
***string*:**
A sequence enclosed in either **"** (double quotes) or **'** (single quotes). The following escapes work: \' \" \n \r \t \b \f. Unicode characters can be escaped with \uXXXX. You can also specify strings in "raw" mode like **r"foo"** (similar to Python's raw strings, see Python's own documentation).
***dateExp*:**
See here for more information on date formats.
***return-clauses*:**
return-clause (**,** *return-clause*)*
A return clause signifies that you want control over what data is returned/displayed.
***return-clause*:**
(**path** | **revision** | **author** | **date** | **comment** | **csid** | **totalLines** | **linesAdded** | **linesRemoved**)
(**as** *word*)?
The attribute to return, optionally followed by a name to use for the column.

4.3.1. Examples

The following examples demonstrate EyeQL to extract information from your repository.

Find files removed on the Ant 1.5 branch:

```
select revisions where modified on branch ANT_15_BRANCH and is dead group by changeset
```

As above, but just return the person and time the files were deleted:

```
select revisions where modified on branch ANT_15_BRANCH and is dead return path, author, date
```

Find files on branch and exclude delete files:

```
select revisions where modified on branch ANT_15_BRANCH and not is deleted group by changeset
```

Find changes made to Ant 1.5.x after 1.5FINAL:

```
select revisions where on branch ANT_15_BRANCH and after tag ANT_MAIN_15FINAL group by changeset
```

Find changes made between Ant 1.5 and 1.5.1:

```
select revisions where between tags (ANT_MAIN_15FINAL, ANT_151_FINAL] group by changeset
```

As above, but show the history of each file separately :

```
select revisions where between tags (ANT_MAIN_15FINAL, ANT_151_FINAL] group by file
```

Find Java files that are tagged ANT_151_FINAL and are head on the ANT_15_BRANCH: (i.e. files that haven't changed in 1.5.x since 1.5.1)

```
select revisions from dir /src/main where is head and tagged ANT_151_FINAL and on branch ANT_15_BRANCH and path like *.java group by changeset
```

Changes made by conor to Ant 1.5.x since 1.5.0

```
select revisions where between tags (ANT_MAIN_15FINAL, ANT_154] and author = conor group by changeset
```

5. Miscellaneous

5.1. Frequently Asked Questions

5.1.1. Questions

1. **General**
   - Can't find an answer here?
   - How does FishEye calculate CVS ChangeSets?
   - Why do I need to describe the branch and tag structure for Subversion repositories?
2. **Troubleshooting**
   - I have installed FishEye, but there is no data in the Changelog. Why?
     I have installed FishEye, and the inital scan is taking a long time. Is this normal?
   - After I commit a change to my CVS repository, it is taking a long time before it appears in FishEye. Why?
   - On my Red Hat Linux system, after running for several days FishEye freezes and does not accept any more connections. How can I fix this?
   - Adding a new repository and on the initial scan, I am receiving messages similar to this in the logs: `org.tigris.subversion.javahl.ClientException: svn: Java heap space`
   - Can FishEye be run as a Windows service?
3. **Subversion**
   - I use the svn:// protocol to access my Subversion repository and FishEye fails to connect to the repository after a short time of successful operation.
   - How can FishEye help with merging of branches in Subversion?
   - I'm using SVNKit and I see errors in the FishEye log such as `SEVERE: assert #B`
     or `Checksum mismatch`

5.1.2. Answers

1. General

1.1. Can't find an answer here?

Try our Online Forums, or contact us directly.

1.2. How does FishEye calculate CVS ChangeSets?

FishEye's goal is to allow changesets to be seen as a consistent stream of atomic commits.

Revisions are collated into the same changeset so long as:

- They have the same commit comment.
- They are by the same author.
- They are on the same branch.
- The changeset does not span more than 10 minutes.
- The same file does not appear in a changeset more than once.

1.3. Why do I need to describe the branch and tag structure for Subversion repositories?

In Subversion, branches and tags are defined by convention, based on their path within a repository, and not directly defined by the repository. There are a few different layout alternatives commonly used. It is also possible that you are using your own custom layout. As a result you need to describe to FishEye which paths in your repository are used as branches and tags.

It is very important that you correctly define in FishEye the layout you are using. If you do not, FishEye will not know which paths represent tags and branches. This will prevent FishEye from relating different versions of the same logical file across separate paths within your repository. It will also mean that FishEye's cache will be much larger as each tagged path will be indexed separately. This will result in an increase in the initial slurp time and may reduce runtime performance.

2. Troubleshooting

2.1. I have installed FishEye, but there is no data in the Changelog. Why? I have installed FishEye, and the inital scan is taking a long time. Is this normal?

When you add a repository, FishEye needs to scan through the repository to build up its index and cache. **This scan can take some time**. As a guide, FishEye should be able to process about 100KB-200KB per second on an averaged-size PC.

If FishEye is accessing the repository over the network (e.g. over a NFS mount), then you should expect the initial scan to take longer.

2.2. After I commit a change to my CVS repository, it is taking a long time before it appears in FishEye. Why?

If possible, FishEye will monitor and parse the `CVSROOT/history` file in your repository to quickly work out what has changed. You may want to check with your CVS administrator to ensure this feature of CVS is turned on.

If you do not have a `CVSROOT/history` file, then a commit will not appear in FishEye until after it has done a periodic full-scan of your repository. You can configure the period of this scan in the admin pages.

2.3. On my Red Hat Linux system, after running for several days FishEye freezes and does not accept any more connections. How can I fix this?

On some Linux systems (particularly RH9), there are socket problems between the JVM and the kernel. To fix this, you need to set the `LD_ASSUME_KERNEL` environment variable before starting FishEye. Add this to the script that starts FishEye:

```
export LD_ASSUME_KERNEL=2.4.1
```

2.4. Adding a new repository and on the initial scan, I am receiving messages similar to this in the logs: org.tigris.subversion.javahl.ClientException: svn: Java heap space

The Java heap space needs to be increased to an acceptable size. See the FishEye Tuning documentation for more information.

2.5. Can FishEye be run as a Windows service?

To run FishEye as a service you can either use SRVANY and INSTSRV to run java.exe or create a Java Service Wrapper. A mechanism to run FishEye as a service will be incorporated at a later stage. In the meantime example wrapper files written by FishEye users can be found on this FishEye forum thread. To install on windows:

1. Unzipping it into the FISHEYE_HOME directory.

2. Run Fisheye-Install-NTService.bat found in FISHEYE_HOME/wrapper/bin.

3. Start the Fisheye Service under control panel.

3. Subversion

3.1. I use the svn:// protocol to access my Subversion repository and FishEye fails to connect to the repository after a short time of successful operation.

On Unix systems, the svn:// protocol is usually handled by inetd or xinetd. These daemons apply, by default, a connection per second limit to incoming connections. Any connections above this rate are rejected by the server. You may either have your system administrator increase the connection rate allowed for svn connection by updating the xinetd configuration or specify a connection per second limit in your FishEye repository definition to prevent FishEye from exceeding the xinetd limits.

3.2. How can FishEye help with merging of branches in Subversion?

FishEye gives you a logical view of your branched files so you can see activity on a single file across multiple branches/trunk.

For merge management the main advantages with FishEye come from its searching features. If, when you check in a merge result, you record the revisions merged, you can find this info in FishEye easily for the next merge operation.

As an example, let's say you have a branch dev created at revision 1300 from trunk. Development has proceeded on both trunk and dev. At some point you wish to add the latest trunk changes into the dev branch. Let's say that is at revision 1400. When you check in the results of this merge, you would use some standard format checkin comment such as:

```
merge from trunk to dev 1300:1400
```

When you come to do the next merge, say at rev 1500, you can use FishEye search to find this checkin comment and know what the starting point for the merge should be. You can then check this in as:

```
merge from trunk to dev 1400:1500
```

Merges back to trunk from the dev branch are managed in the same way.

3.3. I'm using SVNKit and I see errors in the FishEye log such as SEVERE: assert #B or Checksum mismatch

SVNKit may have problems with older version Subversion servers - versions 1.1.x and prior. If this is the case you should either use the native JavaHL layer or upgrade your Subversion serv er to a more recent version.