SmartCVS 7.1 Reference

syntevo GmbH, www.syntevo.com

2010

Contents

Chapter 1

Introduction

SmartCVS is a graphical CVS client. Its main purpose is to manage a number of related files in a directory structure, to control access in a multi-user environment and to track changes to the files and directories. Typical areas of application are software projects, documentation projects or website projects.

Acknowledgments

We want to thank all users, who have participated in the Early Access Program of SmartCVS and in this way helped to improve it by making feature suggestions or reporting bugs.

Used Notations

Throughout this help, features/operations which are only available in the Professional version will be marked by"(Pro Only)".

Chapter 2

Installation

Generally, SmartCVS can be launched on any operating system which has full support for Java Runtime Environment ("JRE") 1.5 or newer. For other systems, e.g. OS/2, which only support Java 1.4, SmartCVS might run, but we don't offer any support for it. It requires a screen resolution of at least 1024x768 pixels.

SmartCVS saves its settings by default outside of the installation directory, so your settings will be kept when updating the SmartCVS applcation. You can find out the actual settings path by opening the About dialog and taking a look at the **Information** page. But before updating, you always have to exit SmartCVS (**ProjectExit**). Closing just the last window is sufficient any more, because SmartCVS still lives in the tray.

2.1 Microsoft Windows

If you want to get a self-contained application which runs out of the box, download the "Windows-bundle with JRE". If you are sure, you already have the right JRE installed on your system, you can use the "Windows-bundle without JRE" instead. The installer requires administrator privileges and installs SmartCVS in the common places, so all accounts on the machine will be able to use it. This implies, that a quickstart icon cannot be installed automatically (it is user-based), so you have to copy the desktop icon yourself to the quickstart icons area.

2.2 Linux and Unix

Download the "generic bundle" and unpack it using the command line

```
tar xzf smartcvs*.tar.gz
```

to your preferred location. Ensure, that an up to date JRE from SUN is installed on your system. Some systems, e.g. Fedora, ship with "GCJ (GNU libgcj)" or beta version of "OpenJDK" which are not sufficient. On Ubuntu systems use Synaptic to install the Java Runtime Environment 1.6.0 from SUN (package "sun-java6-jre" from the "multiverse" repository). Depending on your system configuration, it might be necessary to uncomment the following line

```
#JAVA_HOME=/usr/lib/java
```

in the file `bin/smartcvs.sh` and set the correct path for the installed JRE. Finally, launch the `bin/smartcvs.sh`.

2.3 Mac OS X

Download the "Mac bundle" and double click it. Move the `SmartCVS` application to your preferred location, usually `/Applications`. If you prefer, use the usualy drag and drop to create an icon in the dock.

Chapter 3

Major Features

3.1 Change Sets (Pro Only)

A Change Set is a group of files with an assigned log message and might be known as "prepared commit" from other version control systems. Optionally, files assigned to a change set are not shown in the project structure (see Section 4.4.3). Starting with SmartCVS 7 change sets can be configured to not just contain committable files, but files with other file states as well, e.g. unchanged files. You also can configure, whether the change set should be kept or deleted automatically once it becomes empty. These options allow, for example, to put files into change sets and hence remove them visually from the project structure, which will be changed temporarily but should not be committed.

Change Sets are displayed in the Directory Tree (see 4.1), but below the normal project directory structure. To assign files to a new or existing change set or to remove them from a change set, select the files and invoke **Change Set|Move to Change Set** (see 4.4.5). When you are ready to commit, you need to select the Change Set in the directory structure and invoke **Modify|Smart Commit** (see 4.4.4). This will add unversioned files, remove missing files and finally commit all files in the Change Set.

Another convenient way to assign files to Change Sets is the Change Report (see 4.4.7) using the **Local changes** option. When the project directory structure is selected (as opposed to a Change Set), deactivating the **View|Files Assigned to Change Set** (see 4.4.3) option will give a better overview of changed files not already assigned to a Change Set.

Note	A file can only be assigned to one Change Set.

3.2 Local State

Each displayed file has a local state. The local state is defined by different locally available information, e.g. whether the file exists physically on disk, the corresponding entry in the `CVS/Entries` file of the parent directory, the file's modification time and the state of its parent directories. Each local state is also represented by the icon (see Section 4.2.2) in front of the file name in the File Table (see 4.2).

3.3 Remote State (Pro Only)

Beside the local state (see 3.2) all files also can have an assigned remote state. In contrast to the local state, the remote state is determined by repository content and hence needs a connection to the repository to be refreshed. It shows what will happen when performing a (recursive) Update command (see 4.4.4), e.g. "Will be updated" or "Will be removed".

Use the Refresh Remote State command (see 4.4.7) to manually refresh the remote state. Alternatively, it also can be optionally refreshed after executing some commands, e.g. after the Update command (see 4.4.4).

Following remote state values exist:

Name	Description
Unchanged	you have the latest revision locally
Non-CVS	there is no corresponding revision available in the repository yet (when adding files)
Needs Update	there is a newer revision available in the repository; you will not be able to commit your changes for this file; use the Update command (see 4.4.4) to get the changes from the repository
Needs Update (Conflict)	there is a (newer) revision available in the repository, but you either have marked a new file for adding or an existing for removal
Will be Removed	there is a newer, but removed revision in the repository; you will not be able to commit your changes for this file; using the Update command (see 4.4.4) will delete this file locally or will produce a conflict (if the file is modified locally)
Sticky	the file has a sticky tag or date and will not be changed by an Update command (see 4.4.4)
Needs Refresh	the remote state needs to be refreshed, it is out-of-date

Tip	To keep your remote state in sync with the repository, you can configure the refresh-options in **Edit**|**Preferences** (see 4.4.2).

3.4 Repository Profiles

Repository profiles are a group of settings to access a repository. When SmartCVS executes a CVS command, it reads the `CVS/Root` file from the selected directory and tries to find a matching repository profile. If none could be found (e.g. you checked out a project using another CVS client), an appropriate repository profile will be automatically created. Use the menu item **Project**|**Repository Profiles** (see 4.4.1) to manage the repository profiles.

To tell SmartCVS, it should use a different profile, you need to change the content of the `CVS/Root` files. The best way to do this, is to use **Tools**|**Change Used Repository Profile** (see 4.4.9).

3.5 Password Store

All passwords and passphrases you enter for repository profiles (see 3.4) and select for storing on the hard disk will be saved in a encrypted *Password Store* file. When it does not exists and it should be accessed, you will be asked for a master password to secure the password store. You have the option to choose not to use a master password, but this should be only used in single-user environments for security reasons. Use **Tools|Change Master Password** (see 4.4.9) to change it.

Chapter 4

Main Window

SmartCVS works completely project-centered. Each opened project (see 4.4.1) is displayed in a main window and might have some belonging secondary windows (see 5). When the project is closed, the belonging secondary windows will be closed automatically. To work with multiple projects at the same time, you will need to have to open multiple main windows. You can open main windows manually using the menu item **Window|Open New Window** (see 4.4.11). If you open a project and in the current main window a project is already loaded, SmartCVS will ask you whether to open the project in a new window. It automatically opens a new window, when commands are processed in the current main window.

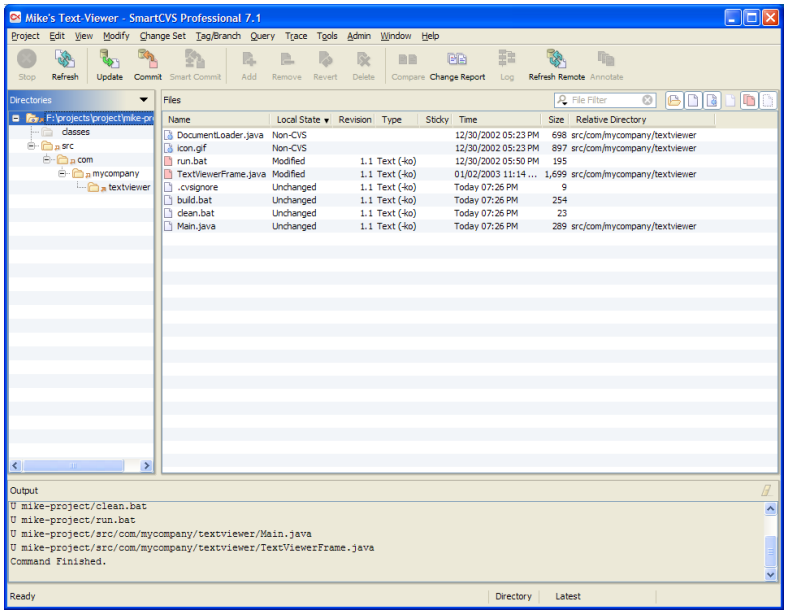

Figure 4.1: The main window with the directory structure to the left and the recursive file table to the right.

The main window is similar to the Windows Explorer: the directory tree (see 4.1) to the left shows the project directory structure and the file table (see 4.2) to the right shows the files with their various attributes. In contrast with Windows Explorer, the right part

in SmartCVS only shows files, no directories. But it can be configured to show even files from subdirectories.

Below the directory tree and the file table you can find the **Output** view, where the output of the executed CVS-commands is displayed.

At the bottom a status bar (see 4.3) informs about the selected files count, the project state and the progress of the operations.

4.1 Directory tree

The directory tree displays the directory structure of your project. Below that, change Sets (see 3.1) are shown.

Select a directory to see its files in the file table. If the recursive mode is active (**View**|**Files From Subdirectories**, see Section 4.4.3), also the files from all subdirectories are listed in the file table.

4.1.1 Used Icons

Icon	Description
	This directory is under CVS control and every file below this directory is unchanged.
	This directory is under CVS control and it contains changed files or new/missing/corrupt subdirectories.
	This directory is under CVS control, contains no changed files nor new/missing/corrupt subdirectories itself, but at least one of its (direct or indirect) subdirectories contains changed files or new/missing/corrupt subdirectories.
	This directory is not under CVS control and not ignored (Maybe a new directory that should be taken under CVS control?).
	If this is the child directory of a versioned one, then this icon indicates an ignored directory. Otherwise it indicates an unversionable directory, e.g. an unversioned parent directory of a versioned directory.
	This directory is considered to be under CVS control, but it does not exist locally. This happens, if the directory was deleted.
	This directory is under CVS control, but its CVS information are not fully valid (maybe its CVS subdirectory is missing or has invalid entries). It is easy to mess up your CVS information, when you copy, move or rename CVS controlled directories!
	This directory does not exist locally, but on the CVS server. Use the Update command (see 4.4.4) to fetch the new files from this directory.

Icons with a blue dot ( ,  and ) determine directories which are the top-most directory of a CVS-working copy (root CVS-directory). Icons with a green dot ( , 

and ) indicate nested modules (which do not form a clean directory structure).

Between the directory icon and the directory name up to two arrows can be shown. A red arrow indicates outgoing changes (commitable changes to the working copy, even Non-CVS files), a green arrow indicates incoming changes (which will be fetched by a recursive Update command (see 4.4.4)). If the arrow is filled, the changes are for this directory, otherwise for a subdirectory.

4.1.2 Project Pop-up

To open, switch or add projects, click the black triangle button in the right top corner of the directory tree. When a project is already open in the main window and you've choosen to open another project, you will be asked whether it should be opened in the current or a new main window. If currently a command is processed, a new main window will be opened automatically.

4.2 File table

The file table displays all files in the selected directory and – if option **View|Files From Subdirectories** (see Section 4.4.3) is selected – the files from all subdirectories as well.

4.2.1 Table columns

Column Title	Description
Name	Shows the file name and the icon for the local state (see below for the icon description).
Local State	Shows the local state (see 3.2) of the files.
Remote State	Shows the remote state (see 3.3) of the files. This column is only available in the Professional version.
Editors	Shows the editors of this file, that are reported by the Refresh Remote State command (see 4.4.7) when **Reserved Edits** are selected in the Project Settings (see 4.4.1).
Revision	Shows the revision of the file, if it is under CVS control.
Type	Shows, whether the file is a text or binary file (for CVS). If it is a text file, the keyword substitution option is also displayed.
Sticky	Shows the sticky tag, date or revision of the file, if it is under CVS control.
Time	Shows the time of the locally existing file.
Size	Shows the size of the locally existing file.
Relative Directory	Shows the path of the file relative to the selected directory. This column only displays useful information, if the recursive mode is active.
Editable	Shows, whether the locally existing file is writable.
Ext.	Shows the extension of the file. Note: everything behind the right most period is considered as extension.
Last Author	Shows the author of the displayed revision. Use the Refresh Remote State command (see 4.4.7) to refresh this column. This column is only available in the Professional version.

Use the **View|Table Columns** (see 4.4.3) menu item to show or hide table columns or change their order. You also can rearrange the table columns by moving them with the mouse.

4.2.2 Used icons in the Name column

Icon	Description
	This file is under CVS control and it is unchanged.
	This file is under CVS control and its file time (and most likely the content) has been changed. Use the Commit command (see 4.4.4) to store the local changes to the repository.
	This file is not under CVS control and not ignored. Use the Add command (see 4.4.4) to put it under CVS control or the Ignore command (see 4.4.4) to ignore this file.
	This file is not under CVS control and it is ignored. Nonetheless, you can use the Add command (see 4.4.4) to put it under CVS control, if necessary.
	This file is considered to be under CVS control, but it was deleted locally. Use the Remove command (see 4.4.4) to remove it from CVS control or the Revert command (see 4.4.4) to restore it if it was accidently deleted.
	This file is marked for adding, but not yet committed. Use the Commit command (see 4.4.4) to put it under CVS control or the Revert command (see 4.4.4) if it was accidently added.
	This file is marked for adding, but not yet committed and later was physically deleted. Use the Revert command (see 4.4.4) to undo the Add operation.
	This file is marked for removal, but not yet committed. Use the Commit command (see 4.4.4) to put it under CVS control or the Revert command (see 4.4.4) if it was accidently removed.
	This file is under CVS control and has conflicts from the last Update or Merge command. You need to fix the conflict, before you commit the file to the repository.
	This file is under CVS control and had conflicts from the last Update or Merge command, but its timestamp is modified (hopefully the conflicts are solved). Use the Commit command (see 4.4.4) to store the local changes to the repository.
	This file does not exist locally, but in the repository. Use the Update command (see 4.4.4) to fetch it.

Read-only files have an additional lock badge ( ).

Use the options in the **View** menu to show or hide files with a certain state. To show just the files which match a special file pattern (e.g. `*.txt`), use the **Edit**|**File Filter** (see 4.4.2).

The table can be sorted by multiple columns. Just click them in the preferred order using the left mouse button. The sort indicator arrow looks darker or brighter depending on the sort order. To clear the sorting, click at the table header using the right mouse button.

4.3 Status Bar

The status bar consists of 4 areas, the *Message Display*, the *Selection Display*, the *Project State Display* and the *Progress Bar*.

The *Message Display* shows the progress messages while processing a command. When you skim through the menus you can see the tooltips of the menu items.

The *Selection Display* shows, whether you have selected a directory or files to operate on.

The *Project State Display*, which only is available in the Professional version, shows the Project State. For example, when "Needs Update" is shown, new revisions are available in the repository and running an Update command (see 4.4.4) on the project root directory will fetch them. This part of the status bar is also used to show ongoing refreshes of the meta-cache (see Section 4.4.7) which are performed in the background to allow you to continue other operations.

The *Progress Bar* tries to give a hint about how long an operation will take. For some commands, like the Checkout command (see 4.4.1), SmartCVS cannot display a useful progress, because the file count is unknown.

4.4 Menu Items

4.4.1 Project

Projects are checked out modules ("sandbox", "local working copy") with specific settings. Each project has an associated root directory. By default, the project's name indicates the root directory, but you are free to change the name.

Most options of a project you can change in the project settings (see 4.4.1). The view-settings (recursive, show ignored files, show non-CVS files and show unchanged files) are also project specific, but they can be customized through checkable menu items in the View-menu.

Create from Directory

Use this command to create a project (see 4.4.1) from an already checked out working copy. Just specify the root directory of the working copy and the name for the project.

Check Out

Checks out a module (or multiple modules) and creates a new project. To *export* a module without the `CVS` subdirectories, first check it out using this command and then export a backup (see 4.4.9) later.

Page 'Repository' Choose the repository profile (see 3.4) to determine from which repository you want to check out files. If you do not find the desired profile in the combobox, use the **Manage** button to open the Repository Profiles dialog (see 4.4.1).

Page 'Modules' When switching to this page, the repository will be scanned. The repository structure is displayed in one tab, all module definitions at another tab. To further explore the repository structure, click the **[+]** icon left to a repository directory. Select the directory, file or module you want to check out. To find directories in large repositories, click in the repository structure tree and start typing the directory name. You can select multiple directories or files.

To check out a specific part of a module, select the **Enter Path** radio button and enter the path (relative to the repository root without leading slash).

Page 'Target Directory' Enter the directory path where the module should be checked out to in the **Local Directory** text field or choose it using the directory button. You may enter a non-existing path; SmartCVS will create it automatically during the checkout.

If you have selected to check out a directory, SmartCVS automatically selects the **Check out into alternative path** option and fills the **Alternative Path** text field with the name of the selected directory. If the checkbox is unselected, SmartCVS will check out into the full directory path.

Page 'Checkout Options' Choose one of the following options to specify what you want to check out.

- **Default (keep sticky)**: Use this option to get the latest revisions on the main trunk.
- **Main Trunk's Head (reset sticky)**: Use this option when checking out over an already existing working copy and you want to ensure that you will get the latest revisions on the main trunk.
- **Retrieve Tag/Branch (new sticky)**: Use this option to check out files with the specified tag or the latest revisions in the specified branch. Clicking the periods button right beside the input field lets you select the tag or branch. Note, that the periods button only is enabled, if you have selected to check out exactly one directory.
- **Retrieve Revision (new sticky)**: Use this option to check out the files with a specified revison. This option only makes sense when you want to check out only a certain file.

Select the **Before Date (new sticky)** option, if you don't want to fetch the latest revisions on a branch or the main trunk, but at the specified date. This option makes no sense in combination with retrieving revisions or tags.

Select **Recurse Into Subdirectories**, to also fetch files from subdirectories of the selected directory or module.

Select **Prune empty directories**, if SmartCVS should automatically remove empty, CVS-controlled directories.

Note	CVS only offers the possibility to remove files (not directories!) from the repository using the Remove command (see 4.4.4). This means, that directories – if they were added to the repository – remain there forever (unless you delete them directly in the repository). Therefore this option was introduced for the Checkout and the Update command.

Page 'Project Settings' After a checkout, SmartCVS also creates a new project. Here you can specify the project settings, which also have influence for the checkout command itself.

If you store text files in the repository using the UTF-8 encoding, you need to select the appropriate option. For detailed explanation of the settings, refer to the Project Settings (see 4.4.1).

Page 'Confirmation' Review the choices you have made in the previous pages. Use the **Back** button to modify your choices. Press **Finish** to start checking out your selected directory, module or file.

Create Module

Use this command to store a new local file structure (project) in the repository. In contrast with the Import command (see 4.4.10), this command initially just puts the project root directory under CVS control and the files (and the subdirectories in the project) should be added later using the Add command (see 4.4.4).

Page 'Local Directory' Enter or choose the directory for which you want to create the module. Since it will be taken under CVS control, you must not specify a directory which is already under CVS control.

Page 'Repository' Choose the repository profile to determine in what repository the module should be created. If the desired repository is not listed, use the **Manage** button to open the Repository Profiles dialog (see 4.4.1).

Page 'Module Name' On this page you define how the new module in the repository should be named and where it should be created. By default the module will be created at the root level. If you want to place it somewhere deeper in the repository structure, use the periods button and select the parent location of the new module.

If, for example, you have selected the **Root Path** "/projects" and entered the **Module Name** "MyProject", the module will be created at "/projects/MyProject".

Page 'Project Name' Creating a module also creates a new SmartCVS project automatically. Here you can specify the name for the new project.

Page 'Confirmation' Review the choices you have made in the previous pages. Use the **Back** button to modify your choices. Press **Finish** to start creating the module.

Open

Lets you open an already defined project. You have the option to choose whether to open the project in a new or the current main window. The **In current window** option is not available, when the current window was processing a command while opening this dialog.

Close

Closes the current project. If multiple main windows are open, the current window will be closed as well.

Project Manager

The Project Manager helps you managing projects (see 4.4.1).

Use the **Add** button to create a new project from an already checked out local working copy. To change the name or root path of a project, use the **Edit** button. Changing the root path is only possible, when the project is not open. To delete projects, select them in the list and click the **Delete** button. Only the SmartCVS project will be deleted, not the files and directories. You are free to arrange your projects as you like using the **Move Up** and **Move Down** buttons or drag and drop.

Repository Profiles

Use this dialog to create, remove or edit repository profiles (see 3.4).

Click the **Add** button to create a new repository profile (see Section 4.4.1). You also can create a new repository profile based on an existing one by clicking the **Copy** button, while the original one is selected. If you want to edit a repository profile, select it and click the **Edit** button. Note, that you only can change these settings (name of the profile, authentication details) which does not change the CVS-location of the repository profile. To delete repository profiles, select them and click the **Delete** button. To order the repository profiles, use the **Move Up** and **Move Up** buttons or drag and drop. To show the stored passwords or passphrases temporarily, select **Show Passwords and Passphrases** from the context menu and enter the master password (see 3.5). Press the **OK** button to make your changes permanent.

Add Repository Profile First, choose the Access Method:

- **pserver** for a password-authenticated CVS server,
- **sserver** for a secure, password-authenticated CVS server,
- **ext (SSH1, SSH2)** for a SSH-based access to the CVS server.

Enter the **User Name** (login name), the **Server Name** (or IP-address) and the **Repository Path** (the path on the CVS-server, where the repository is located). If the server is running on a non-default port, select **Non-Default** and enter the port number. To enter a full CVS-location (looks like `:pserver:anoncvs@cvs.netbeans.org:/cvs`), click the **Enter CVS-Location** button. If the access to the repository should be done via the proxy (see Section 4.4.2), select the **Use Proxy** checkbox. Click **Next** to switch to the **Configuration** page.

When you've selected the **pserver** or **sserver**, enter the password and select, whether SmartCVS should store the password on disk, so you will not be asked again. When you've selected the **ext (SSH1, SSH2)** access method, you need to select the authentication to use. If you select **Password**, enter the password. If you select **Public/Private Key**, enter the path to your private key file and the passphrase. The options **SSH-Type to Use** and **Preferred Public Key** are only necessary to change for non-default SSH server setups. If you are having problems to connect the server, ask your CVS server administrator about these options.

When the option **Verify connection when pressing Next** is selected, SmartCVS tries to establishes a connection to the CVS server when clicking the **Next** button. If you did not provide a password or passphrase, SmartCVS will ask you for it if necessary. Click **Next** to switch to the **Name** page.

By default the name of the repository profile is equal to its CVS-location. Alternatively, you can assign a name to your profile by **Use This Profile Name**.

Settings

Use this action to edit different settings of the currently opened project.

Global Options Select **Concurrent Edits**, if the members of your team should be able to modify the project files at the same time. Select **Reserved Edits**, if the team members by default have read-only files and need to run the Edit command (see Section 4.4.8) if they need to modify them.

Select **Use compression for faster client-server-transfer**, to compress the data sent to and received from the CVS server. This will speed up the transfer, especially on a dialup connection.

Note	Some newer GNU CVS servers (1.12.9 < version <= 1.12.13) us a buggy zlib-library and have problems when this option is selected.

Select **No history logging** to tell the CVS server not to log your actions. This option typically is unselected.

Text File Encoding Select, which encoding your local text files have. This option is important for, e.g., the built-in file compare to display correctly non-US-ASCII characters.

If you have some UTF-16 encoded text files in your project, select the checkbox.

Note	This option only has influence on local operations like built-in file compare or the conflict solver. UTF-16 encoded files need to be stored as binaries in the repository (SmartCVS detects them as binaries automatically), otherwise the CVS server will destroy the content when doing merges!

Define, whether the text files of your project should be stored as ASCII or as Unicode (UTF-8) in the repository. If all of your team members are working on the same operating system, you can use the **ASCII** option. If you use text files that contain unicode characters (e.g. Chinese) or your team members are working on different operating systems, you should select the **Unicode (UTF-8)** option, before adding/importing any text file in the repository! Even if other SmartCVS users access this module from other operating systems using other encodings, they will get the right characters.

Warning!	Be sure, that all of the users who access the module use the same options!

Text File Options Select **Don't convert tabs** when SmartCVS should not change the tab characters in your text files. Select **Convert inner-line tabs to spaces** to change tab characters to spaces, which are not the leading characters in the lines of your text files. Most often inner-line tabs will be inserted by not-so-smart text editors or IDEs and should be converted to spaces, because they make the files look ugly with different tab size values. Select **Convert tabs to spaces** to convert all tab characters in your text files to spaces. For the last two options you also need to set the **Tab Size** to the value you are using in your text files.

Select **Strip trailing whitespace** to remove spaces and tabs from the ends of the lines of your text files.

Note	The above two options do not influence your local files directly, they only change the way, how your files are stored in the repository (or you get them during an Update (see Section 4.4.4) or Checkout (see Section 4.4.1) command).

Select one of the options from **Use These Line Separators** to tell SmartCVS, what line separators to use, when text files are fetched from the CVS server and should be stored locally.

- **Automatic (platform-dependent)**: uses the line separators of your operating system
- **Mac**: uses Mac line separators (\r)
- **Unix**: uses Unix line separators (\n) and
- **Windows**: uses Windows line separators (\n\r)

Use one of the non-platform-dependent options, when you explicitely need to work with – for example – Unix-files on Windows.

Note	This option does not change your files immediately. The files will be changed, when they are updated from the server.

If text files with Windows line separators accidently were committed with a Unix CVS client, selecting the option **Ignore erroneous CarriageReturn-characters** will cause an update command to fetch the files correctly from the repository.

4.4.2 Edit

Stop

Use this action to stop CVS commands in a save way. It can take a few moments to find a secure point to stop the currently executing command.

Open File

Use this action to open the selected file with an external tool. You can define these external tools in the Preferences dialog (see 4.4.2).

Open Directory

Use this action to open the selected directory with the configured external tool. You can define the external tool in the Preferences dialog (see 4.4.2).

Reveal in Finder (Mac OS X and Pro only)

Use this action to select the file or directory in the Finder.

File Filter

Use this command to show just the files which match a certain file pattern (e.g. `*.txt`). Enter the file pattern in the text field in the right top corner of the file table. If you prefix the file pattern with an exclamation mark, you can define what files should *not* be shown (e.g. `!*.gif` hides all gif-files). Click the left magnifier button to enable regular expressions as file filter or to save or restore patterns. Click the right cross button to clear the filter and to show all files.

Select

Use this action to select files with a specified pattern. Enter the pattern for files to select in the **File Pattern** input field. Valid wildcards are "*" (any character) and "?" (one arbitrary character). Files from the file table that match the pattern, are added to the file table's selection.

Deselect

Use this action to deselect files of a specified pattern. Enter the pattern for files to deselect. Valid wildcards are "*" (any character) and "?" (one arbitrary character). Files from the file table that match the pattern, will be deselected.

Select All

Use this action to select all (visible) files.

Select Committable Files

Use this action to select all committable files.

Select Directory

Use this action to select the deepest directory that is the common parent to all selected files. If only one file is selected, its immediate parent directory is selected.

Copy Selected Information

Use this action to copy the visible information of the selected files to the clipboard. The copied information depends on the visible table columns and their order (see **View|Table Columns**, Section 4.4.3).

Copy File Path

Use this action to copy the full path of the selected file to the clipboard.

Preferences

Use this dialog to edit the project-independent preferences.

On Start-up

- **Open last project** Select this option, if SmartCVS should automatically open the last project on start-up.
- **Show 'What do you want to do'-dialog** Select this option, if SmartCVS should show dialog on start-up, which lets you reopen existing projects or create new ones by checking out or creating new modules.
- **Do nothing** If SmartCVS should do nothing on start-up, use this option.

Select **Show tip of the day**, if you want to see the "Tip Of The Day"-dialog during start-up. Unselect it, if you don't want to see the tips any longer.

Select **Remove obsolete projects**, if SmartCVS should search for projects with unavailable root paths. If obsolete projects were found, you will see a confirmation dialog on start-up.

User Interface When a file is double clicked, you can specify what should happen: **Open the file** using the assigned External Tool (see Section 4.4.2) or **Compare the file** to invoke the Compare command (see 4.4.7).

You also can specify the maximum line count in the CVS Output Console as well as the maximum number of projects shown in the Project Pop-up (see 4.1.2).

Here you can define how file filtering (see Section 4.4.2) and quick search (start typing in the file table or directory tree) works.

On some operating systems, like Microsoft Windows, using a Java 6 or newer allows to show an icon in the system tray. If the option **Show icon in the System Tray** is selected, SmartCVS can stay in the background, even when all windows are closed. Starting a new SmartCVS instance will automatically connect to the already running SmartCVS instance and open a new main window.

Independent of your system locale, you can customize the date and time formats used by SmartCVS. Changing this option requires a SmartCVS restart.

User Interface—Accelerators Use this page to customize the accelerators (shortcuts) of main and secondary window menu items.

To set or change an accelerator, double click the menu item line, press the key combination and click **Assign**. To remove an existing accelerator, select the menu item line and click **Clear**.

User Interface—Popup Menus Use this page to customize the popup menus of the main window.

First select the **Popup Menu** to change. The you will find all available menu items on the left and the current popup menu structure on the right. Use the **Add** button add a left-selected menu item before the right-selected item. You also can a **Add Separator** or **Add Menu** before the right-selected item. Each (sub) menu contains a gray placeholder at the end to allow adding items to the end of that (sub) menu. Use the **Remove** button to remove right-selected menu item, separator or sub menu.

User Interface—Toolbar Use this page to configure which toolbar buttons are shown in the project window and how they are ordered.

Project When opening a project with the **Project**|**Open** menu item (see Section 4.4.1) or with the Project pop-up (see 4.1.2), you can specify the default behaviour: either open the project **In current window** or **In new window**. Select **Ask when using the Project-pop-up** to let SmartCVS ask for the window to open, when opening a project by the shortcut in the directory tree.

Select the option **Confirm closing** if SmartCVS should ask you when you clicked the close button of a main window with a loaded project.

Project—Default Project Use these options to define the default project settings. They are used when creating new projects.

Actions—Add Select the default text file keyword substitution. This option is used in the Add command (see 4.4.4) and the Smart Commit command (see 4.4.4).

Actions—Check Out Select under what conditions the repository browser in the Checkout Wizard (see Section 4.4.1) should show the `CVSROOT`-directory.

Actions—Commit Select the option **Remind me to enter a commit message**, if SmartCVS should notify you when you try to Commit (see 4.4.4) or Smart Commit (see 4.4.4) without entering a log message.

Furthermore, you can specify how many commit messages SmartCVS should remember for the corresponding commands.

Actions—Conflict Solver Decide, whether you want to use the built-in or an external Conflict Solver (see Section 4.4.9). If you want to use an external tool, enter or choose the **Command** and enter the necessary **Arguments**. To insert the patterns, that are replaced by the appropriate file names of the files to merge, use the triangle button.

Actions—Edit/Unedit If you need the reserved checkout commands (see Trace menu), you have to set up the external commands, that make files read-only or writable.

Actions—Refresh Select, whether SmartCVS should scroll to the file table's top or should keep the same files visible when invoking the Refresh action (see Section 4.4.3).

Select **Refresh on frame activation**, if the Refresh (see Section 4.4.3) should be automatically performed, when the corresponding project's frame gets activated. On Windows this automatic refresh is very fast and recommended.

Actions—Remote State At this page you can define, whether the Remote State should not just be loaded after opening a project, but also refreshed. Additionally you can define, whether the Remote State should be refreshed autmatically after tag operations (add/remove tags, create branches).

Text Editors Use this card to customize the font, colors or keyboard behaviour for all built text editors.

File Comparators Here you can define different file-name-patterns with corresponding file comparators. SmartCVS searches from top to down to find a matching file pattern for the file to compare. For text files you can use the built-in text file comparator, too.

Click the **Add** button to add a new file comparator definition, the **Edit** button to change the selected definition or **Delete** button to delete the selected definitions. You can use drag and drop or the **Move Up** and **Move Down** buttons to rearrange the definitions.

Enter the file pattern (wildcards "*" and "?" are allowed) and select, whether you want to use the built-in text file comparator or an external tool. If you want to use an external file comparator, enter the command (without parameters!) and the parameters using the `${leftFile}` and `${rightFile}` placeholders. If necessary, you also can pass

the used file encodings to your external file comparator as command line parameters. You also can use file viewers which will be invoked two times for the two files to be compared. This is especially very useful to compare graphic files.

External Tools Here you can define file-pattern-to-program associations for the Open File command (see 4.4.2). SmartCVS searches from top to down to find a matching file pattern for a file to open. If it finds a matching entry, the appropriate application (editor or viewer) is invoked with the full path of the file as the parameter-variable `${filePath}`. If no file pattern matches, the internal file viewer is used instead.

Click the **Add** button to add a new external tool definition, the **Edit** button to change the selected definition or **Delete** button to delete the selected definitions. You can use drag and drop or the **Move Up** and **Move Down** buttons to rearrange the definitions.

You can use the system specific edit or open association or any command. When entering the external tool, be sure to only enter the path to the executable in the **Command** text field and put the necessary command line parameters to the **Arguments** input field. Surround parameters with spaces with quotes, e.g. `"-d *.txt"`. To insert a quote in the command line parameter, use two quotes, e.g. `"-execute ""print"""`.

Tip	You are able to define multiple comma-separated file patterns for one association, e.g.“*.txt, *.xml”.

External Tools—Directory Command Here you can specify the command which is used, when you invoke the Open Directory command (see Section 4.4.2). With that it is possible to open, e.g., a console window or Explorer for the specified directory.

Connection If you have problems with the execution of a CVS command or you think it does not work correctly, select this option. The CVS connection protocol between the client (SmartCVS) and the server is logged into files in the directory `<settings-directory>/connecti` Sending these two files *and a detailed report* about what went wrong or is expected to work different and the log-file (`<settings-directory>/log4j-log.txt`) to us (support@smartcvs.com) helps us to fix bugs.

The **Connection Timeout** defines, how long SmartCVS should wait for answers from the CVS server. If you have really large projects, it might be necessary to increase this value.

Select the **SSH2 Connection Caching** option if the authentication phase to your SSH2-based CVS server takes a long time. The **Disconnect After** value defines, after what period of idle time SmartCVS should close the connection to the CVS server.

Select **Use this proxy** and enter the right options, when (some of) your Repository Profiles need to use a proxy to connect to the repository.

Check for New Version On this page you can configure the interval in which SmartCVS should check the `http://www.syntevo.com` website for new versions. Use **Help|Check for New Version** to perform the check manually. If a new update is available, you will get a notification dialog at start-up.

If your machine is behind a HTTP proxy, select the option **Use proxy server to connect to the internet** and configure the settings according to your browser settings.

Note	These proxy settings only are used for the check for new versions, not for executing CVS commands!

4.4.3 View

Table Columns

Use this action to define what table columns should be visible and how they should be ordered.

Use the **Add** button to make the (left) selected table column visible and the **Remove** button to hide the (right) selected table column. With the **Move Up** and **Move Down** buttons you can define the order of the table columns. You also can use drag and drop move table columns between the lists or change their order.

Select "Make this configuration the default" to use this table column layout for new projects.

Refresh

Use this action to force SmartCVS to reread the directory structure below the selected directory.

Tip	If you want SmartCVS to perform a refresh always if SmartCVS gets activated, select the appropriate option in **Edit**|**Preferences** (see Section 4.4.2).

Files From Subdirectories

If this option is selected, the file table shows all files below the selected directory, even those from subdirectories. If this option is unselected, the file table shows only the files within the selected directory (as the Windows Explorer does).

Tip	Keep this option selected helps you not to forget to add new or remove deleted files to/from the repository respectively.

Unchanged Files

Select this option, if you want to see unchanged files in the file table. This options is selected by default.

Non-CVS Files

Select this option, if you want to see non-CVS (not ignored) files in the file table. This option is selected by default.

Ignored Files

Select this option, if you want to see ignored files in the file table. This option is unselected by default.

Files Assigned to Change Set (Pro Only)

Select this option to see also files already assigned to a Change Set (see Section 3.1). This options is selected by default. Disabling it makes it easier to see files not assigned to a Change Set.

Remote-Only Files (Pro Only)

Select this option, if you want to see files which are not locally available but only in the repository and will occur when you would perform an Update command (see 4.4.4). This option is selected by default.

4.4.4 Modify

Smart Commit (Pro Only)

Use this command to add, remove and commit files at once. It either can operate on committable files and on Change Sets (see Section 3.1). When invoked on a Change Set, the Change Set's Log Message will be the default for this dialog.

Click the **Show Changes** button to open the Change Report window for the selected files (see Section 4.4.7).

Standard Options Select all non-CVS files that you want to add, all missing-files that you want to remove and all changed files you want to commit (you can use **Edit|Select Committable Files**, Section 4.4.2). After entering the commit message and pressing **OK**, the selected non-CVS files will be added, the missing-files will be removed and finally all selected files will be committed.

Note	The non-CVS files will be added with the default keyword-substitution (see Section 4.4.2).

You can achieve the same functionality with the Add command (see Section 4.4.4), Remove command (see Section 4.4.4) and Commit command (see Section 4.4.4), but less comfortable.

Select **Mark unchanged, touched files as unchanged files** so touched, but unchanged files (only file time changed) will occur as unchanged after the commit.

Advanced Options Select **Refresh Remote State**, when SmartCVS should refresh the Remote State after the commit.

Select **Refresh Editors** (which only is enabled, if the Editors table column is visible), when the editors information should be refreshed after the commit.

Commit

Use the commit command to commit your local changes into the repository.

Click the **Show Changes** button to open the Change Report window for the selected files (see Section 4.4.7).

Standard Options Select **Recurse into subdirectories** when committing directories, to tell SmartCVS not only to commit the files in the selected directory, but those in subdirectories, too.

In **Log Message** you can enter a describing message about what you have done. To specify a clear message (e.g. "fixed null-pointer exception") is a good idea to better track your changes. The messages for revisions that have already been committed can be viewed with **Query|Show Transactions** (see 4.4.7) or **Query|Log** (see 4.4.7). Press <*Ctrl*>+<*Space*> to complete file names.

Note	If there exists a `Template` file in the `CVS` subdirectory of the selected directory, the content of this file is read into the **Log Message** input field. To set up such a template file, consult the documentation of the `CVSROOT/rcsinfo` file in the repository.

Advanced Options Select **Force commit even if there are no changes** to create a new revision of the file in the repository, even if the file hasn't changed.

Select **Don't run the module program on the CVS-server**, if you do not want the CVS-server to run a special command after your commit, e.g. for email notification about the change.

Select **Refresh Remote State**, when SmartCVS should refresh the Remote State after the commit. Select **Refresh Editors** (which only is enabled, if the Editors table column is visible), when the editors information should be refreshed after the commit.

If you want to add a tag to the committed file revisions, select the similar named checkbox and enter the commit tag.

Tip	In general, it is often advisable to commit files as soon as possible. This reduces the risk of conflicts (changes made by co-workers at the same pieces of code) significantly. Of course it is also adviced to commit only changes, that compile, especially in the whole project.

Reset Unchanged, Touched Files (Pro Only)

Use this command to mark files which only were changed by time as unchanged.

Update

Use this command to fetch the latest changes from the repository. It is intented for daily use. If you need more Update options, please use **Modify|Switch (Special Update)** command (see Section 4.4.4) instead.

Select **Recurse into subdirectories** when updating a directory, if you want to process the selected directory and subdirectories, too.

Select **Create new directories** when updating a directory, if you want to get new directories in the repository automatically created.

Select **Prune empty directories** when updating a directory, if you do not want empty CVS-controlled directories remaining.

Note	CVS does not support the removing of directories. You only have the possibility to remove files from directories. But you can let SmartCVS automatically remove empty, CVS-controlled (sub-)directories while updating. This is a little bit odd, but it works.

Select **Refresh Remote State** if SmartCVS should refresh the Remote State after performing update command.

Switch (Special Update)

Use this command to perform advanced update operations like switching to other branches or tags, reverting files. For day-to-day use we recommend to take the simple Update command (see 4.4.4). To merge explicitely between branches, use the separate Merge command (see 4.4.4).

Standard Options Use the **Default (keep sticky)** option to not switch to a different tag, branch or revision.

Use the **Main trunk's head (reset sticky)** option to fetch the latest revisions in the main trunk with their default keyword-substitution mode. Usually this option is used after working in a branch to switch back to the main trunk.

Use the **Retrieve Tag/Branch (new sticky)** option to either switch to a tag or branch. Clicking the ellipsis button right beside the input field, lets you open the Tag Browser (see 4.4.6) to select the tag or branch to switch to.

Use the **Retrieve Revision (new sticky)** to retrieve individual revisions of your files. This option is useful in combination with the **Revert To** option on the **Advanced Options** page to revert back to old revisions (see description below for further details).

Select the **Before Date (new sticky)** option to get the project state at the specified time (several date-formats are supported by the CVS server, the most straight-forward ist "yyyy-mm-dd hh:mm:ss"). To get the project state at a certain time in a branch, you also need to select the **Retrieve Tag/Branch** option above and enter/select the wanted branch.

Select **Recurse into subdirectories** when updating a directory, if you want to process the selected directory and subdirectories, too.

Select **Create new directories** when updating a directory, if you want to get new directories in the repository automatically created.

Select **Prune empty directories** when updating a directory, if you do not want empty CVS-controlled directories remaining.

Note	CVS does not support the removing of directories. You only have the possibility to remove files from directories. But you can let SmartCVS automatically remove empty, CVS-controlled (sub-)directories while updating. This is a little bit odd, but it works.

Select **Don't change anything (check what would happen)** to test, what files will be added, removed or updated. No local change will be made with this option selected.

Advanced Options Select **Get clean copy (revert local changes)** if you want your locally changes overridden with a plain copy from the repository.

Select **Revert To (to overwrite revisions in the repository)** if you have committed erroneous revisions into the repository and want to overwrite them easily (makes sense only in combination with the sticky Date/Revision or Tag option from the Standard Options' page). If this option is selected, the files are fetched from the repository without changing the local revision or options (so, as you would have modified the file's content locally). This option even works for binary files and whole directories.

Example
You have accidently committed a revision 1.4. The latest valid revision is 1.3.

- Ensure, that you have the latest revision of the branch or the main trunk (you may use the Update command with the selection **Main trunks head (reset sticky)** to get the latest revision of the main trunk).
- Than run another Update command with option **Kind of Revision to Get** set to set to **Revision (new sticky)** (or **Tag (new sticky)** or **Date (new sticky)** depending on how to obtain the latest valid revision from the repository) *and* the checked option **Revert To (Keep Revision/Sticky Options)**. In our example you need to get the content of the revision 1.3, but it will occur as you would have made the changes to the latest revision.
- Now you just need to commit the modified revision 1.4 (that actually has the content of revision 1.3) to get the new valid revision 1.5.

If you need to override the file's default keyword substitution mode for your local project, enable **Override default keyword substitution** and select the keyword substitution mode you need.

Merge

Use the merge command to merge changes between branches.

Select, whether you want to merge changes from a brach/tag or a certain revision. If you already have merged changes and want to merge more changes, select **Starting With** and choose between a tag and a special revision.

Select **Recurse into subdirectories** when merging a directory, if you want to process the subdirectories, too.

Select **Create new directories** when merging a directory, if you want to have new directories in the repository automatically created.

Select **Prune empty directories** when merging a directory, if you do not want empty CVS-controlled directories remaining.

Select **Don't change anything (check what would happen)** to test, what files will be added, removed or merged. No local changes will be made with this option checked.

Revert Local Changes

Use this command to revert local changes. Modified files will be replaced by clean copies from the CVS server (keeping the revision number). Added (but not committed) files will become non-CVS (unversioned) files. Removed (but not committed) files will become unchanged files.

Add

Use the add command to add new files to the repository. To undo removed files before commit or to undo erroneously added files, use the **Modify**|**Revert Local Changes** command (see 4.4.4).

CVS handles text and binary files differently. The **File Type** option specifies, how the selected files should be added:

- **Automatic detection of text and binary files**: this automatically detects, whether the specified files are binary or text files,
- **Treat files as text files**: this forces to treat the files as text files,
- **Treat files as binary files**: this treats the specified files as binary files.

Note	SmartCVS uses the selected local encoding (see Section 4.4.1) for the binary/text file detection. If it finds at least one character from the 0x00..0x1F range (except the 0x09 tab-, 0x0A new-line- and 0x0D line-feed-ASCIIs), the file is considered to be a binary file.

The **Keyword Substitution for Text Files** option determines, what keyword substitution mode should be used for the text files. The default value can be changed in **Edit**|**Preferences** (see Section 4.4.2).

Note	You can not and need not to add directories with SmartCVS. Simply add the files within the unversioned directory and the directory will be added automatically.

Ignore (Pro Only)

Use the ignore command to put an entry in the global or local `.cvsignore` file. This causes the specified file(s) or directory to be ignored in SmartCVS.

Depending on whether the command was invoked on a directory or a file, different dialogs are shown.

Ignore Files With the **How to Ignore** option you specify, whether the file(s) should be ignored explicitely using the full file name(s) or by a pattern. The pattern may contain the well-known wildcards "*" (0 or more characters) and "?" (one character) to form the pattern.

With the **What .cvsignore File to Use** option you decide, whether the global or local `.cvsignore` file should be used for adding the new entry. The global `.cvsignore` file is located in your home directory and influences any project and directory. The local `.cvsignore` file is located in the directory, the selected file is located in. It only influences files in this directory. If the global or local `.cvsignore` file isn't present yet, it will be created automatically.

Ignore Directories Click the **Ignore** button to ignore the directory. Directories are always ignored explicitly (no patterns possible) and locally.

Remove

Use this command to remove the selected CVS-controlled files from CVS control. To undo added files before commit or to undo erroneously removed files, use the Revert Local Changes command (see 4.4.4).

Select **Delete local files if they still exist** if you want the local copy to be deleted physically. You should leave this option unselected, if you want to keep your local file.

Delete Locally

Use this command to delete local files (or unversioned directories) without any influence to the repository (you can use the Explorer or command line for getting the same effect).

Warning! The files will be deleted without the trash-can, so better think twice!

Apply Patch (Pro Only)

Use this command to modify your working copy according to the change information provided in a patch file. You can create a patch file from within the Change Report window (see 5.2).

Page 'Select Patch File' Select the patch file and click **Next** to read the file.

Page 'Select Files to Patch' After reading the patch file, all files to patch will be shown. If you only want to patch some files, unselect the files you don't want to patch. In the table column **Switch to Revision** you should see the revision to which the patch will be applied. Click **Next** to continue.

Page 'Change Set' On this page you can define a change set which will be assigned the patched files. Using this feature is especially useful when you already have modified files in the working copy. Click **Finish** to start patching. The files will be updated to the given revision and then the patch will be applied.

4.4.5 Change Set (Pro Only)

Move to Change Set (Pro Only)

Use this command to change the assigned Change Set (see 3.1) of selected, committable files.

To move the selected files to a new Change Set, select **New Change Set** and enter the Log Message of the new Change Set. Press <*Ctrl*>+<*Space*> to complete file names. You may also select **Delete this Change Set once it gets empty** to automatically delete this Change Set once it gets empty. This may for instance be convenient to automatically remove Change Sets after they have been committed (and are empty therefore).

To move the selected files to another, already existing Change Set, select **Existing Change Set** and choose the **Target Change Set**.

To remove the selected files from their currently assigned Change Set, select **Remove from Change Set**.

Move Up (Pro Only)

Use this command to move the selected Change Set (see 3.1) one position up (when having multiple Change Sets).

Move Down (Pro Only)

Use this command to move the selected Change Set (see 3.1) one position down (when having multiple Change Sets).

Delete (Pro Only)

Use this command to delete the selected Change Set (see 3.1). This will only affect the Change Set assignment, not the physical files.

Edit Properties (Pro Only)

Use this command to change the log message and other properties of the selected Change Set (see 3.1).

4.4.6 Tag/Branch

Add Tag

Use this command to add a tag (a named label) to file revisions. This is very useful to mark and later retrieve all files at a certain project state (e.g. when developing applications a certain program version). It also makes understanding changes in the Log window (see 5.3) more easy.

Enter a valid tag name in the **Tag Name** input field. Pressing the ellipsis button allows you to choose a tag from the Tag Browser (see 4.4.6).

Select **Revisions specified by local files** to add the tag to those revisions, that your local files have.

Select **Latest revisions on main-trunk** to (blindly) add the tag to the latest main-trunk-revision of all files in the repository (below the selected directory).

Select **Revisions specified by another tag** and enter an existing tag to set the new tag at the same revisions. You can use this option to set the tag to all latest revisions in a branch by entering the branch tag. To "rename" tags, first add the new tag to all revisions of the old tag (no branch-tag!) and then remove the old tag from all files in the repository.

Select **Revisions specified by a date** to tag all (main-trunk) revisions, which existed at a specified date.

Select **Check that not modified** to ensure that all locally modified files are committed before tagging.

If files in the repository already contain the entered tag, you can determine, what should happen:

- **Don't move tag**: will not change the tags in files, which already contain the tag.
- **Move tag to current revision**: will move the tag from locally available files to the current revision. Removed files which also contain this tag, will not be changed.
- **Ensure that no other files keep this tag**: will remove the tag from all files (even removed) and then tag the current revisions. If you continuesly tag your project with a tag like "all-tests-run-successfully", this option should be used. Otherwise removed files will still contain the tag and when updating the project with this sticky tag, beside the last files you also will get removed files which this tag, which most likely is not what you want.

Even if you want to tag the files with a new tag, the Tag Browser can be of great value when the tags should follow a common naming scheme, because you often only need to change the version number in the tag name.

Example
To tag the files of program version 1.8, select the existing tag "version-1_7" from the Tag Browser and change the 7 to 8.

Remove Tag

Use this command to remove tags from the selected file(s).

Enter the tag name you want to remove in the **Tag Name** input field or click the ellipsis-button to open the Tag Browser (see 4.4.6).

Select **Remove tag also from not locally available files** will remove the tag from all files (below the selected directory), even from those, which are not available locally, e.g. because they were removed. This option only is available for directories.

Select **Check that not modified** to ensure that all files are committed before removing the tag from them.

If you need to remove a branch tag, select the **Force To Remove Branch Tag** option.

Note	Removing a branch tag does not remove any revision from the repository, but it makes getting revisions from this branch very hard. So better think twice before removing a branch tag.

Create Branch

Use this command to create a branch for the selected files.

Enter a valid branch (tag) name in the **Branch Name** input field. Pressing the ellipsis button allows you to choose a tag from the Tag Browser (see 4.4.6).

Select **Revisions specified by another tag** to create the branch at the revisions of all files, which have the specified tag.

Select **Revisions specified by local files** to create the branch at the revisions specified by the local files.

Select **Check that not modified** to ensure that all locally modified files are committed before tagging.

Select **Update Branch** to automatically update with this branch, so you can start working in it.

Tag Browser

The Tag Browser helps you to easily get an overview of the tag-structure of your project. When it is invoked from a tag-input-field, you can use it to select tags or branches.

To the left you can see the branch structure and to the right all tags. To refresh the displayed tag-structure, press the **Refresh** button. Select **Show also tags from subbranches** to make SmartCVS also display the tags from subbranches of the (left) selected branch. An icon with a exclamation mark badge is used, when the branch-structure is not non-ambiguous. For example this might happen, when files on the main-trunk and a branch have the same tag.

4.4.7 Query

Compare

Use the compare command to compare a local file with any revision of the same file from the repository.

Decide, whether you want to compare the local file with

- the **Latest Revision** from the repository,
- the **Same Revision** from the repository as the local revision or
- **Other** revision, which can be specified by **Date**, **Revision** or **Tag**.

If this command is invoked on a locally modified file, **Same Revision** automatically is preselected. If this command is invoked on an unmodified file, **Latest Revision** automatically is preselected.

When the command is performed, a Compare Window (see 5.1) will be opened.

Compare Two Files (Pro Only)

Use this command to compare two selected local files. This can be very useful to easily merge some changes from one file to the other.

Change Report (Pro Only)

Use the Change Report to see a compact list of local or repository changes for a set of files.

Decide, what changes should be shown. To be able to assign files to change sets, you need to use the **Local changes** option. After clicking **OK**, the Change Report Window (see 5.2) will be opened.

Show Log

Use this command to see the history tree for the selected file or compare between past revisions.

After a few moments, the Log Window (see 5.3) will be shown.

Show Status

Use this command to see the plain text output of the status command.

Refresh Remote State (Pro Only)

Use this command to refresh the Remote State (see 3.3) and/or editors of the selected directory.

If your project is running in Reserved Edits mode (see Section 4.4.1), additionally an "editors" command is issued and its output is redirected to the "Editors" table column.

Annotate (Pro Only)

Use this command to get an annotated view of the selected file's history.

After a few moments, the Annotate Window (see 5.4) will be shown.

List Repository Files (Pro Only)

Use this action to either list all files in the repository or those which have a certain tag.

Tab 'List Repository Files Options' Select, whether you want to get all files or just those which have the entered tag. After clicking **OK**, the List Repository Files Window opens:

Tab 'Refresh Options' Select **No refresh** to directly search in the SmartCVS meta cache. Of course this only makes sense, when the cache was refreshed before.

Select **Fast refresh** to let SmartCVS perform a 'cvs history' command to detect changed files and then perform a 'cvs rlog' command to just refresh the needed directories. Note, that with this option selected, only changed files can be detected, but not newly tagged (because of a CVS server limitation).

Select **Deep refresh** to tell SmartCVS to perform a 'cvs rlog' command. Executing such a command is very expensive, but cannot "forget" to refresh newly tagged files.

After the command executed, the List Repository Files window (see 5.5) opens.

Compare Repository Files (Pro Only)

Use this action to compare project states in the repository using tags to define these states.

Tab 'Compare Repository Files Options' Enter (or choose using the Tag Browser (see 4.4.6) by clicking the ellipsis buttons) two tags, for which you want to compare the repository files. If you enter "HEAD" (without the quotes) as one tag, you compare against the latest revisions in the main-trunk. If you choose a branch-tag, the latest revision in the branch will be taken for comparison.

Tab 'Refresh Options' See Section 4.4.7 for details.

When pressing **OK**, the Compare Repository Files window (see 5.6) opens.

Show Transactions (Pro Only)

Use this action to display the (last) transactions (commits) in the repository.

Transaction Options Choose the period, for which the actions should be displayed. If you choose the **From Tag** option, SmartCVS first searches all revisions with this tag and uses the earliest time of these revisions.

When you have selected a subdirectory of your project, use the last two options to select, whether all the displayed transactions should show also files from parent directories or just from the selected or subdirectories.

Tab 'Refresh Options' See Section 4.4.7 for details.

When pressing **OK**, the Transactions window (see 5.7) opens.

Create Patch (Pro Only)

Use this command to create a patch file containing the changes of the selected, changed files.

An alternative to create patches is to use the Change Report (see 5.2).

Refresh Meta-Cache

Use this action to refresh the meta cache, which is used for detecting the Remote State (see Section 4.4.7), for Listing (see 4.4.7) or Comparing Repository Files (see 4.4.7), for detecting the tag structure (see Tag Browser, Section 4.4.6) or Showing Transactions (see 4.4.7).

Such a refresh might be necessary, when someone has added or removes tags or branches in the repository, because the 'cvs history' command, which is used for (relatively) fast detecting changed files does not provide enough information about tag and branch creation/removal.

4.4.8 Trace (Pro Only)

Reserved Edit (Pro Only)

Use this command to prepare files for modification in a reserved-checkout environment (see Section 4.4.1).

Select **Try to edit if possible** if SmartCVS should just start editing this file, if nobody else is already editing it. Select **Force edit even if other users are editing** if SmartCVS should start editing even when other users have marked it for editing. Select **Edit only, if latest revision(s)** to start editing only, if you already are using the latest revision (the Remote State shows Latest). Select **Refresh editors** to let SmartCVS automatically refresh the Editors table column, if displayed.

If the edit command is successful, the selected files become writable and the CVS server notifies other users that you are planning to modify the files.

Unedit (Pro Only)

Use the unedit command to tell other users, that you do not want to make modifications on the selected files any more. SmartCVS makes the file read-only and the CVS server notifies other users.

After performing this command, the "Editors" column of the affected files will be changed to reflect the change.

Refresh Editors (Pro Only)

Use this command to refresh the Editors table column independent of the Remote State table column (see **Query|Refresh Remote State**, Section 4.4.7).

Watch (Pro Only)

Use the watch command to either add a watch on files or to remove watches on files.

Choose what temporary watch you want to set for the files:

- **Watch On**: to enable watching (this updates your files read-only and you need to invoke an Edit command before changing them)
- **Watch Off**: to disable watching
- **Add Watch**: adds you to the CVS server's list of users who receive notification about work going on the selected files;
- **Remove Watch**: removes you from the CVS server's notification list for the selected files;

Choose from **Watch action**:

- **All**: to be notified if another user starts editing a file (Edit), commits changes or abandoned editing files;

- **None**: you will not be informed about file changes;
- **Edit**: you will be informed about other users start editing the file;
- **Unedit**: you will be informed about other users finishing edit;
- **Commit**: you will be informed about other users commits.

Watchers (Pro Only)

Use the watchers command to list all users who are watching the selected files.

Lock (Pro Only)

Use the lock command to lock the current revisions.

When performing this command and your user account does not have admin privileges in the repository, you will get an error message like this: `cvs [admin aborted]: usage is restricted to repository administrators` In this case, ask your CVS administrator to enable admin privileges for you.

If you are the CVS administrator, do folloging (taken from CVSNT documentation): In the `CVSROOT` directory on the CVS server, create a file named `admin`, which, is not checked into CVS. Each line in this file contains the user name of each user account with admin privileges. If SystemAuth=no (`CVSROOT/config` file), the names are those that appear in the `passwd` file.

Unlock (Pro Only)

Use this command to unlock previously locked revisions.

Make Read-Only (Pro Only)

Use this command to make the selected files read-only without any communication to the CVS server. You might use the command line or the Windows Explorer as well to get the same result.

Make Writable (Pro Only)

Use this command to make the selected files writable without any communication to the CVS server. You might use the command line or the Windows Explorer as well to get the same result.

Release (Pro Only)

Use this command to unedit all files and finally delete the project.

4.4.9 Tools

Export Backup

Use this command to export the selected files including their directory structure to either a zip-file or a directory.

To export all files below a directory of your project, select the directory and invoke this command. To export only some files (e.g. only modified files), select the root directory to use and the files you want to backup and invoke this command.

To store the files/directories into a zip-file, select **Into Zip-File** and enter or choose a valid file name.

To store the files/directories into another directory, select **Into Directory** and enter or choose a directory. Check **Wipe Directory Before Copying** if you want to ensure to get a clean copy of the exported files without any old file hanging around there. This option is especially useful, if you always backup into the same directory.

Note	This command does not store the local `CVS` directories, because it is meant to be a better replacement for CVS' export command.

Conflict Solver (Pro Only)

Use the conflict solver to visually solve conflicts in conflict files. If you haven't choosen to use an external merge tool at **Edit**|**Preferences** (see 4.4.2), the Conflict Solver window (see 5.8) opens.

Change Keyword Substitution Mode (Pro Only)

Use this command to change the keyword substitution mode of files *in the repository*.

Select the keyword substitution mode to use and press **OK**.

Pin Revision (Pro Only)

Use this command to pin the current revision, so (temporary) modifications will not committed.

A pinned revision is marked with a pin-icon in the "Revision" table column.

Unpin Revision (Pro Only)

Use this command to unpin a previously pinned revision to allow changes to be committed.

Change Sticky Tag (Pro Only)

Use this action to either remove sticky options from files and directories or to set a sticky tag.

Decide, whether you want to remove sticky options or whether to set a sticky tag to your local files.

Example
Imagine following situation. You know, you have send a coworker a copy of your project. Unfortunately you have forgotten to tag it at this time. Now (s)he sends you the modified files of the project and you need to merge his/her changes in the project.
This is very easy. Update your project with the (sticky) date, when you send him/her the project. Now remove the sticky options from your project (which occured because of the update with sticky options) using this action. Then overwrite your local files with your coworker's changes. Add new files and remove obsolete files and run a simple update command without sticky options. CVS will merge for you automatically.
If you would not have removed the sticky options of your local files, CVS would not merge the changes.

Change Used Repository (Pro Only)

Use this command to change the used repository (see **Project|Repository Profiles**, Section 4.4.1), e.g. when switching from pserver access to SSH-access.

Choose the new repository from the combobox. If it does not yet exist, use the **Manage** button to open the Repository Profiles (see 4.4.1).

Change Master Password

Use this dialog to change or set the master password for the Password Store (see 3.5). To change the master password, you will need to enter the current master password. When *setting* (and not *changing*) a new master password, all stored repository profile passwords and passphrases will be lost.

4.4.10 Admin (Pro Only)

Track 3rd-Party Sources (Import) (Pro Only)

Use this command to import 3rd party sources into the repository.

Page 'Directory to import' Enter the directory you want to import or choose it by clicking the directory button.

When clicking **Next**, the directory is scanned. This may take some seconds.

Page 'File Types' In this table all found file patterns from your scanned local directory are listed. SmartCVS detects automatically for each file pattern, whether the matching files are text or binary files. Sometimes you do not want to import some files or want to import text files as binaries. To do so, select the file patterns and click the appropriate button.

Press **Rescan** to perform a rescan of you import-directory (e.g. if you created/deleted some files in the mean time externally). All your made settings about text/binary file or about ignoring the pattern, will be lost!

Page 'Keyword Subst. Mode' Select, what keyword substitution mode should be used to import the text files.

Page 'Repository' Choose the repository you want to import into from the combo-box.

If you do not find the desired repository in the combo-box, use the **Manage** button to open the Repository Profiles (see 4.4.1).

Page 'Module Name' Enter the name of the module you want to create in the repository with this import. If you want to import a newer version of an already imported module, choose the same name as when importing the first time.

Page 'Import Options' Enter an import message (a message to describe your import), the vendor branch, a vendor tag and a release tag. If you don't know, please contact the CVS manual.

Page 'Confirmation' Review the choices you have made on the previous pages. If you feel the need to change them, use the **Back** button.

Press **Finish** to start the import.

User/Password Manager (Pro Only)

Use this manager to add new user accounts, remove them or change their passwords for pserver-controlled repositories.

Page 'Choose Repository' Enter or choose the locally available (e.g. mounted) path to the repository thats users or user's passwords you want to edit.

Note	The specified directory must contain a `CVSROOT` subdirectory.

Page 'Edit Users' To add new users, select the last (empty) table row and enter the new login name of the new account into the **User Name** input field. Type the appropriate password twice, one into the **Password** and one in the **Password (retype)** input field. The password may be empty, if the user may login without a password. Enter a system user name in the **System User Name** input field to specify the system user that should be used to access the repository when logging in with this new user. The system user name may be empty, depending on your CVS-server configuration/requirements. Press the **Add** button.

To change an account's password or it's system user name, select it in the table, edit the appropriate input fields and press the **Apply** button.

To delete accounts, select them press and the **Delete** button.

Page 'Line Separators' Select the line separator to use in the password file. Be sure to specify the line separator the matches your CVS server's operating system.

Press **Finish** if you want the changes to be stored in the repository.

SSH-Key-Pair Generator (Pro Only)

Use this action to create key-pairs for public-private-key authentication with a SSH2-protected CVS server.

Select the type of the key and the public key format. If you are unsure what to use, contact the documentation of your SSH server or ask the administrator.

Type in the passphrase to secure the private key and retype it to prevent input errors.

Enter or choose the file name of the private key file. For the public key file name, `.pub` is automatically appended.

Note	The generated private key file should never leave your machine. Upload the public key file to your SSH server and register it according to the SSH server manual. For a Cygwin OpenSSH server, for example, create a file `$HOME/.ssh/authorized_keys` and copy the generated public key file's content into this file. Then prefix this line with a 'from="*" ' (without single quotes) and save the file.

4.4.11 Window

Open New Window

Use this action to open a new SmartCVS main window to be able to work in different projects simultaneously.

Chapter 5

Secondary Windows

Beside the project main window, secondary windows will be shown as the result of different executed commands. These windows rely on an open project main window, for example, they can select files in the project main window or they use settings from the project. If you close a project, its secondary windows will be closed as well.

5.1 Compare Window

If binary files were detected, instead of the (undisplayable) content the file size and SHA hash codes are shown.

Use the **Refresh** tool bar button to reload the file(s) from disk. Use the **Goto Prev.** and **Goto Next** tool bar buttons (or corresponding menu items) to navigate to the previous or next change. If the right text area is editable, you can easily take changes over from the left side by **Take Left** or **Take Right** tool bar buttons (or corresponding menu items). Additionally you can use the small buttons in the center area, where the links between the left and right blocks are shown.

If you need to show long lines, select the menu item **View|Left Above Right**.

View|Settings

The **Inner Line Comparison** option determines, how inner-line changes should be detected. Select the option, which suites your file type. All other options should be self-explanatory.

Select the **Make default** checkbox to remember these options as defaults.

5.2 Change Report Window

In the upper part you see all changed files with their changes. The lower part shows the comparison result for the currently selected file.

Use the tool bar buttons to navigate through all changes, refresh the report or open a file compare window (see 5.1) for the selected file, e.g. to be able to edit the file. The

available options (**View|Settings**) are similar to those from the Compare Window (see Section 5.1). To create a patch for the selected files, select **File|Create Patch**.

When the **Local changes** change report was opened, you will find a table column **Change Set**. Use the **File|Move to Change Set** menu item to change the Change Set (see 3.1) assigned to a file (see Section 4.4.5). With **File|Log Message** you can edit the log message of the current file's Change Set.

5.3 Log Window

The revision structure contains two types of entries: green, rectangular revisions and cyan, rounded-rectangular branches. To see more revision details at the bottom, select the revision. Same is displayed in a tool tip after a few moments, when hovering over a revision.

You can select revisions by clicking on them. As usual, use the <*Ctrl*>-key to select multiple revisions.

If one revision is selected, you can bring up an Annotate window (see 5.4) for the specified revision.

If one revision is selected, you can compare this revision with the local file (see Compare Window (see 5.1)).

If two revisions are selected, you can compare them (see Compare Window (see 5.1)).

You also have the ability to save the selected revision in a new file, to open it or to update/check it out with the revision marked sticky.

5.4 Annotate Window

You have the possibility to color the file by

- **Revision**: All lines, which where changed in or after the selected (threshold-) revision will be marked.
- **Age**: Depending on the age or revision numbers (lower revision numbers are meant older), the lines are colored from blue (old) over green to red (new).
- **Author**: All lines of the same author are colored the same.

5.5 List Repository Files Window

If you have choosen to list all files, dimmed file icons indicate files, which have been removed in the main-trunk. Dimmed directory icons indicate directories, which are empty or contain only removed files.

To show the remote log (rlog) of a file, select it and click the right tool bar button. With the help of the Log Window (see 5.3) you easily can reactivate removed files.

5.6 Compare Repository Files Window

The icons in the file table have following meanings:

Icon	Description
	Both tags are assigned the same revision (the file is unchanged).
	Both tags are assigned different revisions (the file was modified).
	The file only contains the second tag (the file was added).
	The file only contains the first tag (the file was removed).

To show the remote log (rlog) of a file, select it and click the right tool bar button.

To compare the two tagged revisions of a file, select it and click the mid tool bar button.

5.7 Transactions Window

In the above table, you can find the transactions, sorted by time in reverse order (newer transactions are on top). Selecting transactions, shows their file structure in the below directory tree and file table. If just one transaction is selected, its commit message is displayed in the top right. Transactions marked with an asterisk (*) are those which are not locally available. Transactions marked with an exclamation mark (!) are those which partly are not locally available. Use the **Filter Branch** and **Author** comboboxes to filter the shown transactions.

Use the **Search Author and Commit Messages** text field to limit the transactions to those, which contain the entered text in the author or commit message. Click the **Show All** button to display all transactions.

Double clicking a transaction displays the Change Report (see 5.2).

Use the **Transaction**|**Export as XML-File** menu item to create an XML file which contains all information (including file paths) of the selected transactions. If you already have a XSL file available, you can specify that to transform directly to your preferred output format.

Use the **Transaction**|**Copy Date** menu item to copy the transaction's date for usage in the Switch (Special Update) command (see 4.4.4).

Double clicking a modified file displays the Compare Window (see 5.1) for the previous and new revision. Use **File**|**Show Log** to display the log of the selected file (see Log Window (see 5.3)). You also have the possibility to select the corresponding local file in the main window of SmartCVS using File—Select Local File (if it exists locally).

5.8 Conflict Solver Window

With the left two tool bar buttons you can navigate from conflict to conflict. The right two tool bar buttons allow to take either the left or right solution of the conflict. If necessary, you can also edit directly in the resulting document at the bottom. Use the first four **View** menu items to define where the three displayed file contents should be placed.

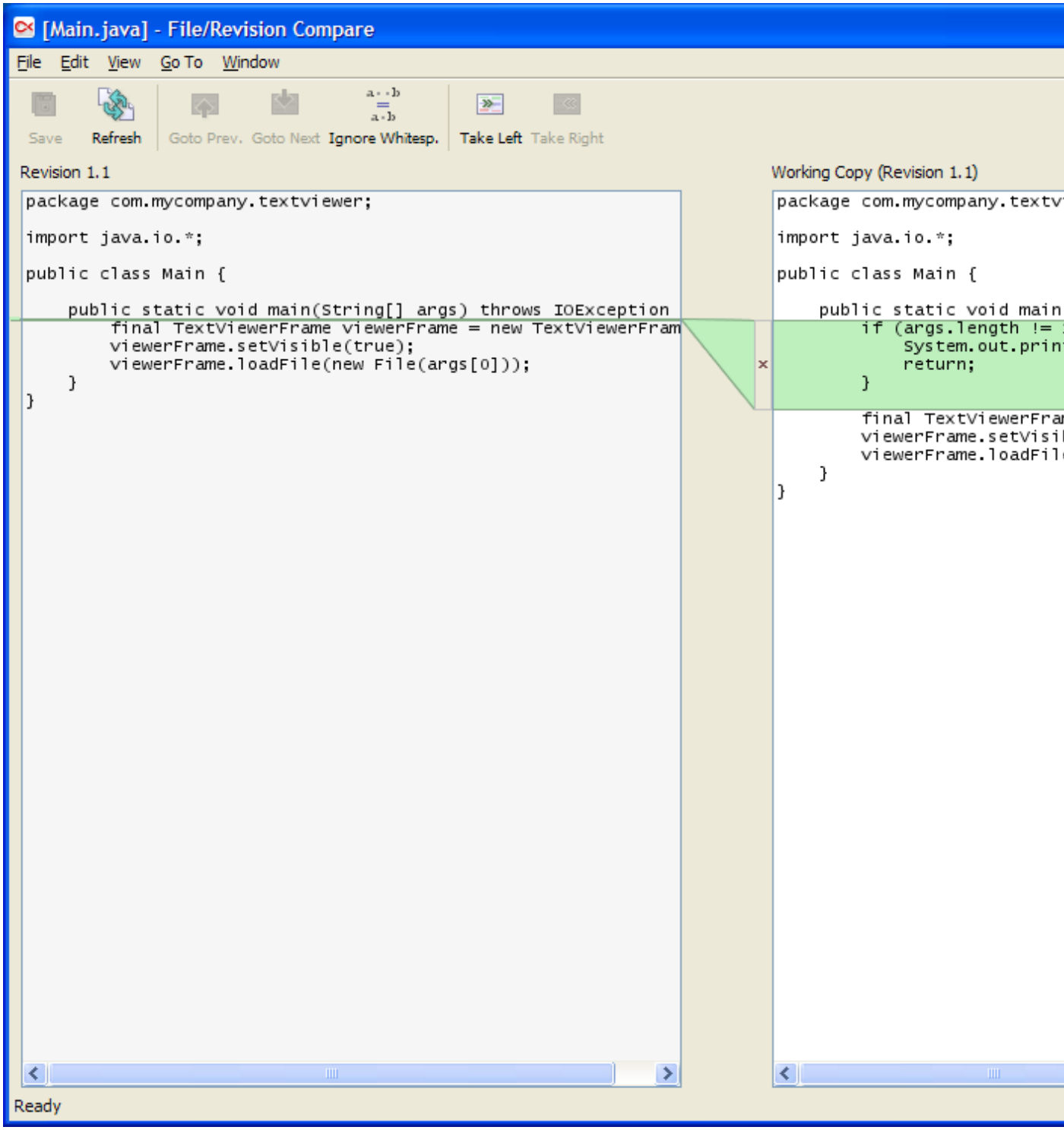

Figure 5.1: The Compare window.

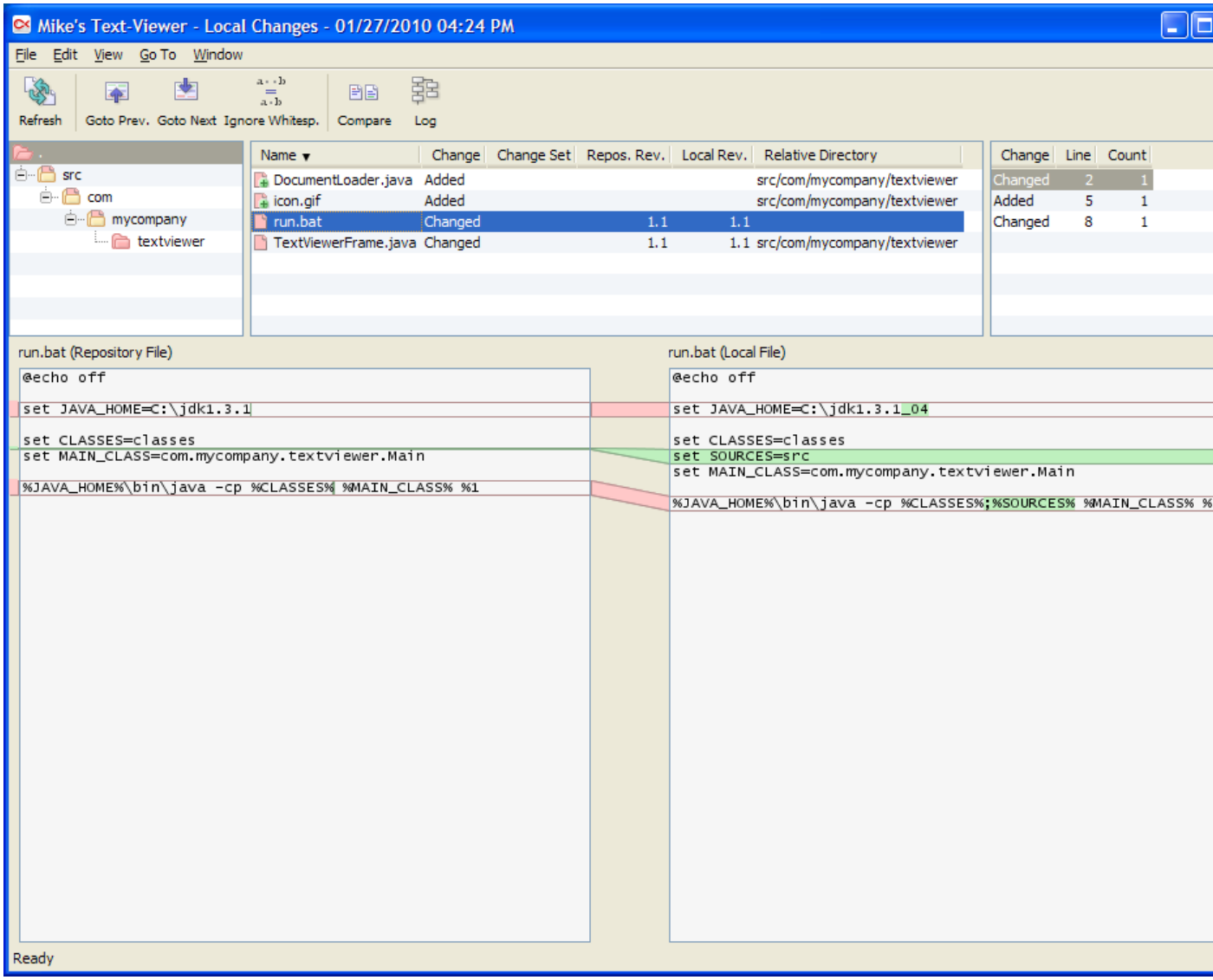

Figure 5.2: The Change Report window.

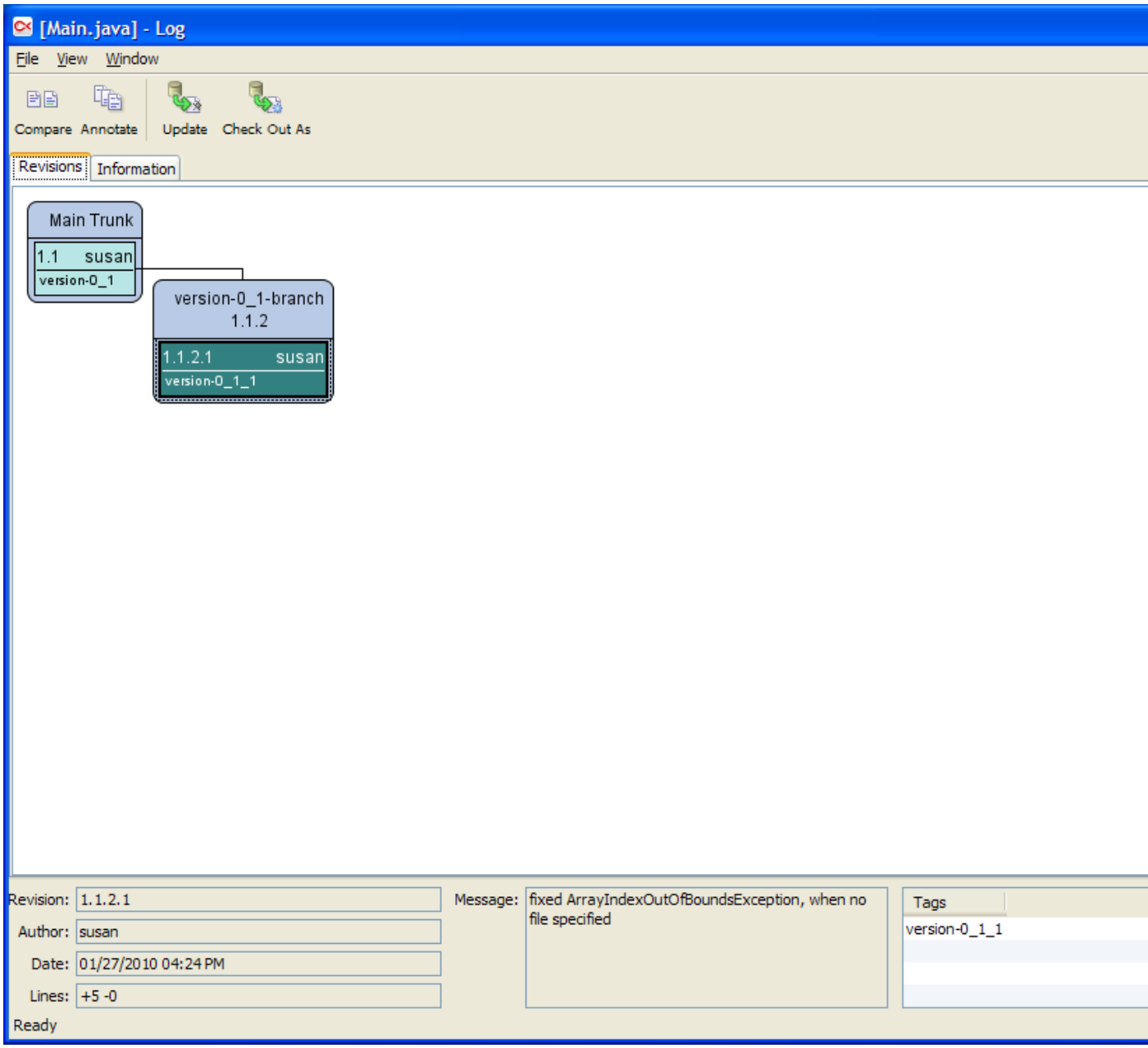

Figure 5.3: The Log window.

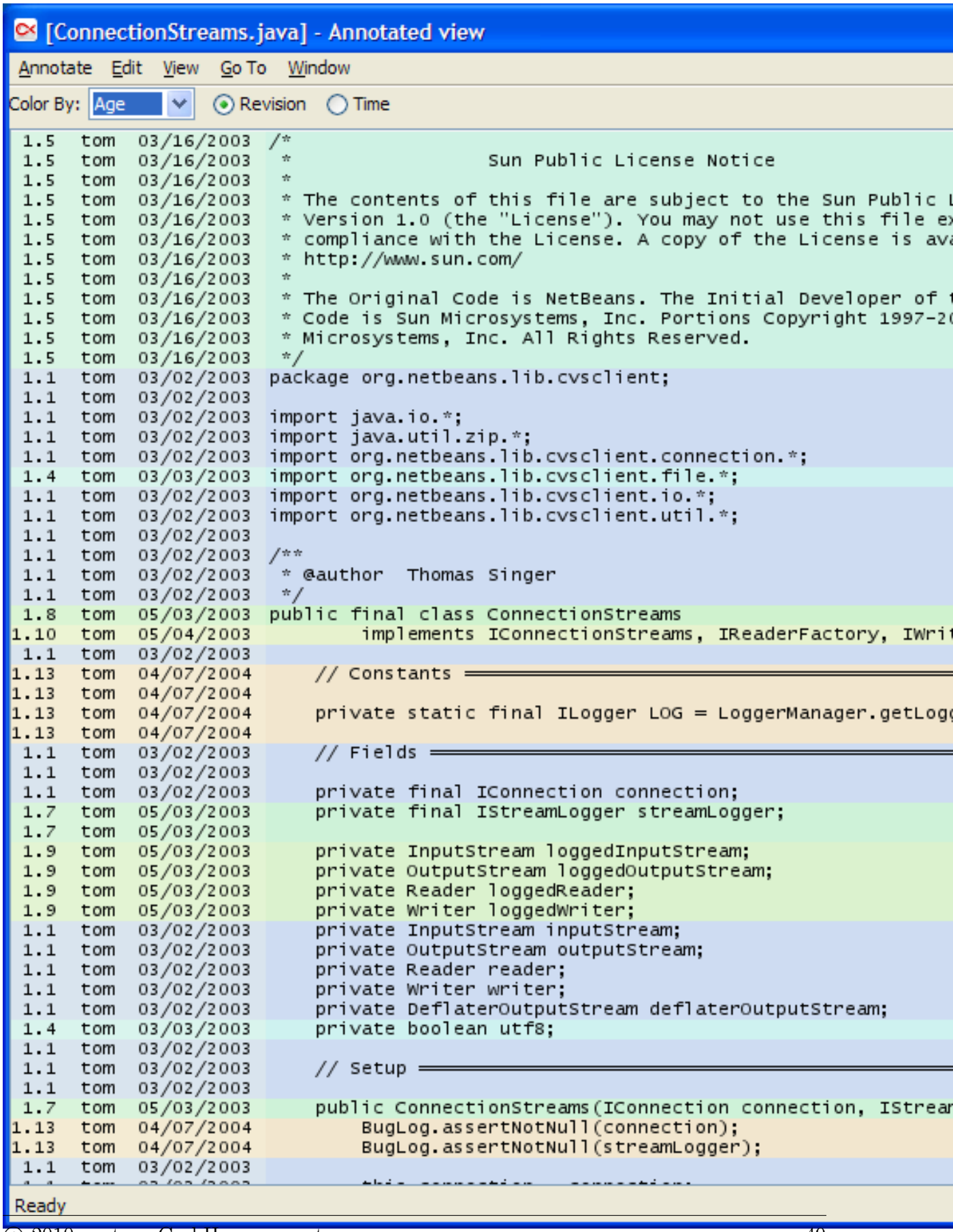

Figure 5.4: The Annotate window.

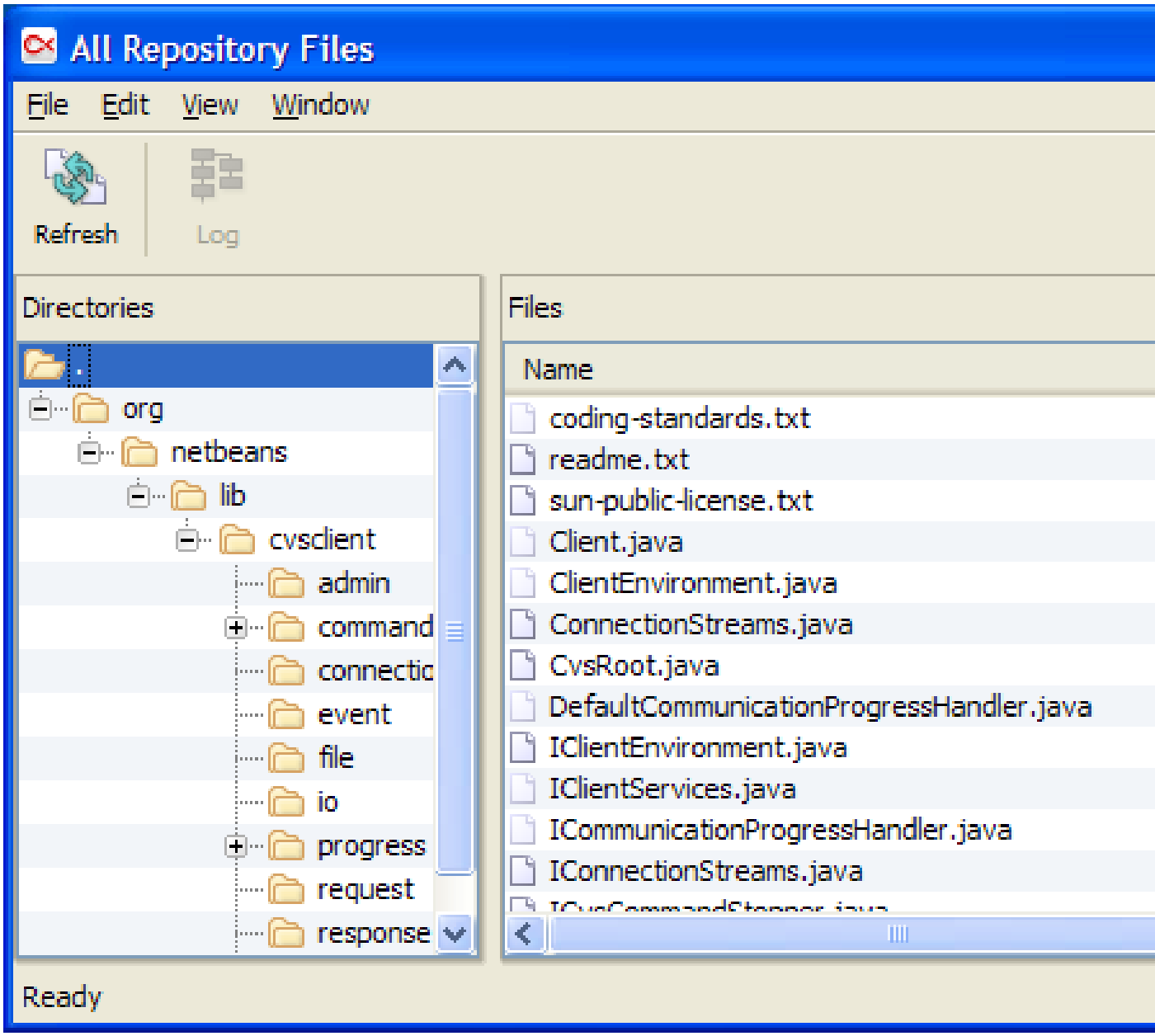

Figure 5.5: The List Repository Files window.

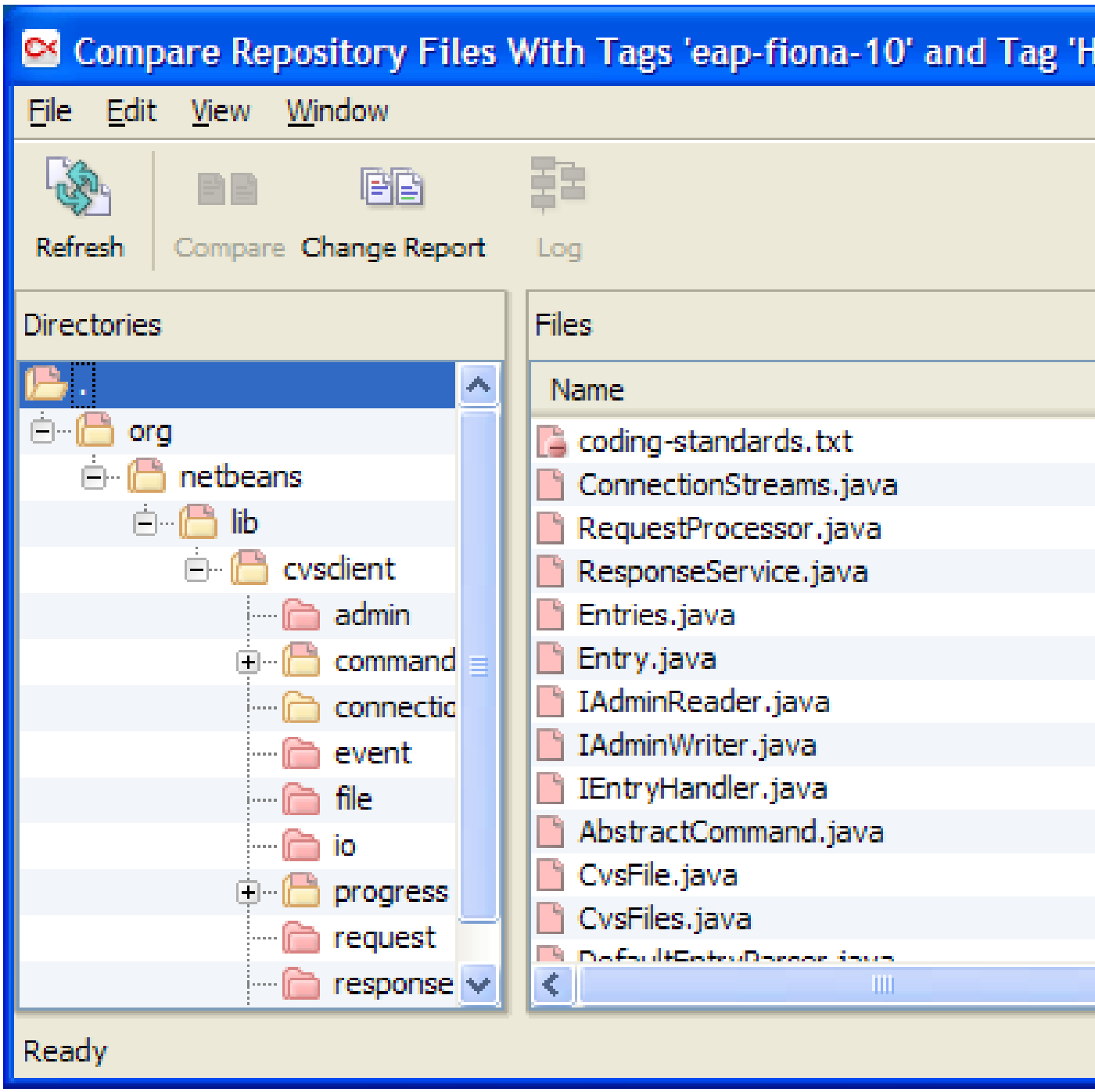

Figure 5.6: The Compare Repository Files window.

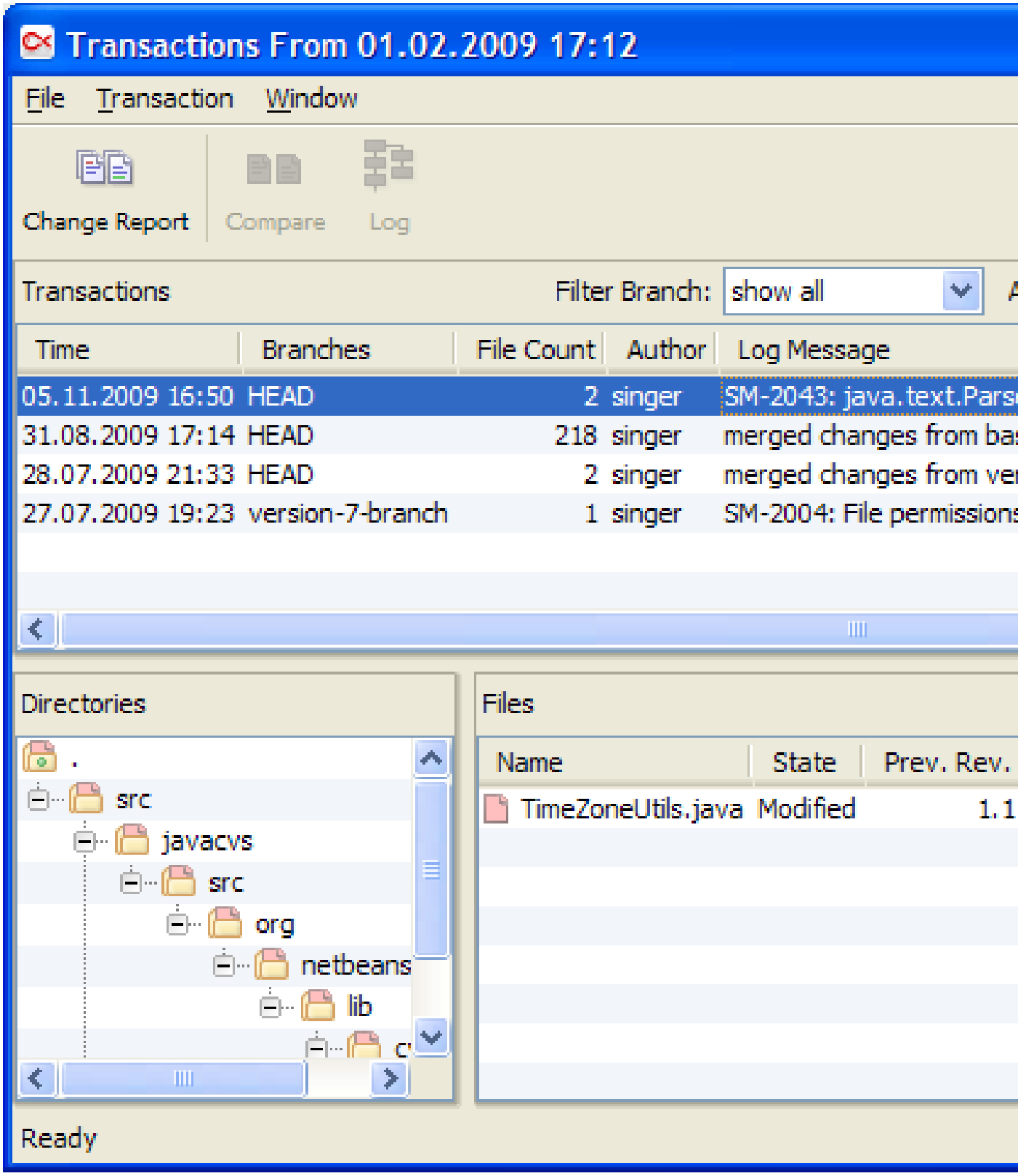

Figure 5.7: The Transactions window

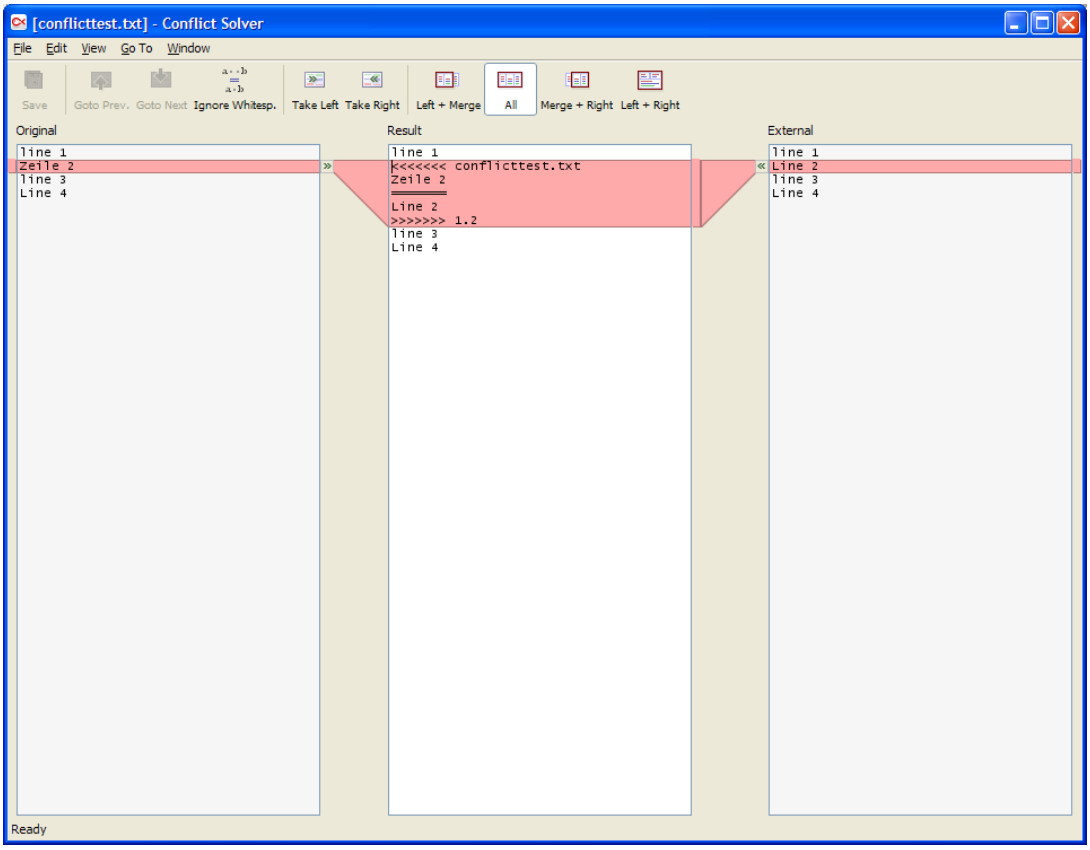

Figure 5.8: The Conflict Solver window

Chapter 6

Miscellaneous

6.1 Foundation and Professional version

SmartCVS is available in two versions, the free Foundation version with a basic feature set and the commercial Professional version with advanced and convenience features (see http://www.syntevo.com/smartcvs/features.html). Both program versions may be used for non-commercial and commercial purposes. The Professional version includes one year of free updates – no matter what version number.

To evaluate the Professional version, you can get a demo license key from the web-site http://www.syntevo.com/smartcvs/evaluate.html. To purchase SmartCVS Professional licenses, go to http://www.syntevo.com/smartcvs/purchase.html.

6.1.1 The license file

A physical difference between the Foundation and Professional version is, that the latter one is detected by a valid license file. Which kind of version you are currently using, is displayed in the title of the SmartCVS main window. To switch back to the Foundation version, delete the file `license` in the SmartCVS settings directory.

6.2 Special files

SmartCVS is designed to be run in a multi-user-environment. This means, that it only *reads* from the installation directory.

6.2.1 Where SmartCVS stores its settings?

If no system property `smartcvs.home` is set, SmartCVS will save its settings in its platform-dependence default settings directory. On Linux and other Unix-like operating systems, the default settings directory is `$HOME/.smartcvs/`, on Mac OS X it is `$HOME/Library/Preferences/SmartCVS/` and on Windows `%APPDATA%\syntevo\smartcvs\`. Below this default settings directory, version-dependent subdirectories are used to allow concurrent usage of two different SmartCVS versions. In the following `$SMARTCVS_HOME` is used for this path.

On Windows

In the file <installation-directory>\bin\smartcvs.vmoptions you can define Java-system properties as well as VM parameters (e.g. for memory management). The file should contain examples and the description. So, for example, you can change the smartcvs.home system property to ${smartcvs.installation}\.smartcvs to make SmartCVS portable:

```
-Dsmartcvs.home=${smartcvs.installation}\.smartcvs
```

As this example demonstrates, you can reference other system properties by referencing them as ${<other-system-property>}.

On Unix-like Systems

In the file <installation-directory>/bin/smartcvs.sh you can define Java system properties as well as VM parameters.

On Mac OS X

Right click the SmartCVS.app in the Finder and click **Show Package Contents**, double click the occurring directory Contents and there you will find the Info.plist file. Open this Info.plist file in a text editor to set system properties or change VM parameters.

General

Alternatively, you can set system properties (except smartcvs.home) in $SMARTCVS_HOME/smartcvs.properties.

6.2.2 $SMARTCVS_HOME/license

This is the license file which defines in what version (Foundation or Professional) SmartCVS will run. For more details about the license, see Section 6.1.

6.2.3 $SMARTCVS_HOME/passwords

This file contains the encrypted passwords and passphrases (see Section 3.5).

6.2.4 $SMARTCVS_HOME/log4j.properties

This file defines what SmartCVS should log into the $SMARTCVS_HOME/log.txt file. You usually don't need to change that file.

6.2.5 $SMARTCVS_HOME/smartcvs.properties

This file defines system properties for SmartCVS. Here you can, for example, define that SmartCVS should use the platform-dependend look and feel even on Windows (if you don't like the Windows look and feel) by adding following line:

```
smartcvs.lookAndFeel.usePlatformIndependent=true
```

6.2.6 $SMARTCVS_HOME/log.txt

This file contains the log information (see Section 6.2.4).

6.2.7 $SMARTCVS_HOME/accelerators.xml

This file contains the accelerator settings (see Section 4.4.2).

6.2.8 $SMARTCVS_HOME/projects.xml

This file contains the project settings.

6.2.9 $SMARTCVS_HOME/repositories.xml

This file contains the repositories profiles.

6.2.10 $SMARTCVS_HOME/settings.xml

This file contains the general settings of SmartCVS.

6.2.11 $SMARTCVS_HOME/uiSettings.xml

This file contains context menu settings.

6.2.12 $HOME/.cvsignore

This file may contain file patterns for files that should be globally ignored (affect each project). It can be extended by the Ignore command (see 4.4.4).

Note	SmartCVS expects one file pattern per line in the `.cvsignore` files. Hence it can handle also file patterns with spaces in the file name.

Chapter 7

What's New

7.1 New in Version 7.1

Following features are new or improved in version 7.1.

- :ssh:user@server:port/path is now an alternative supported notion for :ext:user@server:port/path to improve compatibility with other CVS clients
- Transactions: ability to specify time or tag range
- Transactions window: added ability to filter by branch and author
- (Smart) Commit dialog: ability to open a change report for the files to commit
- Preferences: View Defaults have been removed; instead all **View|Settings** dialogs now have a **Make default** option
- Preferences: ability to configure matching behaviour of file filters and file speed-search: case sensitive, ignoring case, smart upper case
- Preferences: the toolbar for the project window can be customized
- Preferences: the external tools can be configured to use the system open or edit association (Java 1.6 or newer)
- Preferences: the external conflict solver configuration allows to pass the name for the left and right files to the external tool
- Preferences: the file comparator configuration allows to configure file viewers to compare files by invoking two instances (e.g. useful to compare graphic files)
- Delete Tag command: by default the option **Remove tag also from not locally available files** is selected
- Windows: introduced a portable bundle which can be installed on a USB stick

- Windows: smartcvs[c].vmoptions now include user-specific configuration file %APPDATA%\syntevo\SmartCVS\vmoptions (if present) to avoid having to edit the usual read-only smartcvs[c].vmoptions file
- Project window: also remembers the output height
- Project window: the progress bar is now only occurs if it can display something
- built-in File Compare: option to ignore case changes
- built-in File Compare and Conflict Solver: added search and replace feature
- built-in file compare: ability to deactivate scrollbar synchronizing
- built-in file compare and change report: if binary file type is detected, the file length and SHA hash code is shown
- built-in file compare and conflict solver: added **Save** toolbar button
- built-in file compare and conflict solver: if showing whitespaces, an end-of-line symbol is drawn
- About dialog: shows (copyable) information like the settings directory
- time table columns: for times from the last two days shows *Today* and *Yesterday* instead of the date
- directory tree: when hovering with the mouse over partly hidden entries, they will be shown completely
- toolbar buttons now have a short descriptive text below the icon
- all text components now show a context menu for the usual edit items (cut, copy, paste, undo, ...)
- OS X: added **Edit**|**Reveal in Finder** to select the directory or file in the Finder
- OS X, file comparator configuration: by default uses the system graphics viewer to compare graphics
- OS X: the toolbar now grows out of the title bar ("unified toolbar"), although due to a Apple-Java bug the background painting is still incorrect when a dialog is shown
- OS X: message dialogs now have no title any more
- OS X: dialogs which previously had only a **Close** button now don't have them any more like native dialogs

7.2 New in Version 7

Following features are new or improved in version 7.

- the Compare window (see 5.1) allows to place the files above each other for better handling long lines
- the Conflict Solver window (see 5.8) allows to place the merge file below the left and right ones for better handling long lines
- on Java 6 (or newer) systems SmartCVS optionally can now live in the system tray; relaunching SmartCVS causes a new project window to be opened
- accelerators (see 4.4.2) can now be configured for all windows
- a lot of application-modal dialogs are now parent-modal; this allows to continue working in a window while in others dialogs are open, e.g. to review file changes in the Change Report window (see 5.2) while writing the commit message
- Change Report window (see 5.2): ability to create patch
- ability to apply patch (see 4.4.4)
- massive look and feel improvements, esp. on the Mac and on Windows
- on Windows by default the Windows look and feel is used; to use the platform-independent look and feel, set the environment variable `smartcvs.lookAndFeel.usePlatformInd` to `true`
- Commit command (see 4.4.4), Smart Commit command (see 4.4.4) and Move to Change Set command (see 4.4.5): file path and file name completion using <*Ctrl*>+<*Space*>
- Transactions window (see 5.7): exporting to XML allows to transform the output using XSL files
- ability to change or reset the master password (see 3.5)
- ability to select the font used in the Compare (see 5.1), Conflict Solver (see 5.8) or Annotate windows (see 5.4)
- the Repository Profiles Manager (see 4.4.1) allows to show known passwords and passphrases
- Log window (see 5.3): ability to hide empty branches
- the Compare command (see 4.4.7) now precompares the files byte-wise and, if identically, only shows the file compare when explicitly requested; this trivial compare also works for binary files
- file and directory input fields now support file system completion

- added drag-and-drop support for various tables
- the Switch (see 4.4.4) and Merge (see 4.4.4) command now also work for files in change sets
- its now possible to pass the used encoding on command line for external compare and merge tools
- the commit tag input field (see Section 4.4.4) now rejects known branches and was moved to the **Advanced** tab
- introduced own menu item for Create Project from Directory command (see 4.4.1)
- the Compare Repository Files command (see 4.4.7) now shows a warning when nothing was found for one tag, e.g. because it was entered incorrectly
- Change Sets (see 3.1) now have options for deleting when empty and for accepting uncommittable files
- it is now possible to store VM options in a user-specific `smartcvs.properties` (see 6.2.5) file
- in the Preferences (see 4.4.2) you now can configure the date and time format used by SmartCVS
- each text editor window (e.g. Compare window (see 5.1), Annotate window (see 5.4), ...) has its own configurable view settings (tab size, line number display, ...)
- table columns: different improvements, e.g. shift-double click between headers to make column wide enough for visible table content
- Tag Browser (see 4.4.6): readded filter text field
- ability to hide repository-only files (see Section 4.4.3)
- improved fingerprint dialogs
- ability to delete unversioned directory
- on Windows versions settings are now stored in the `AppData\Roaming\syntevo\SmartCVS\` directory
- all text fields provide now undo-support
- Check for New Project now also works behind a proxy
- improved the look and behaviour of the file filter text field
- Mac: modified windows, e.g. the Compare window (see 5.1) are now marked
- windows are now placed smarter

- the project state in the status bar (see 4.3) now shows a time stamp
- the Change Report (see 4.4.7) now also can be invoked for unchanged files
- input fields for external tools now show the required attribute variables
- external tools now can contain quotes in the commandline, e.g. for small scripts
- built-in file/directory choosers got replaced by file/directory text fields with the ability to show a file chooser upon request (on Mac a Mac-specific one is used)
- the `CVS/Root` and `CVS/Repository` files now have a trailing line separator for better compatibility with other tools

7.3 New in Version 6

Following features are new in version 6.

- *Change Sets*, see Section 3.1
- the local Change Report (see Section 5.2) now shows the Change Set of the files and allows to change it
- the Remote State (see Section 3.3) shows also repository-only files and directories
- support for file permissions (to disable it, set the Java environment variable `smartcvs.permissions.disable` to `true`)
- ability to customize accelerators (see Section 4.4.2) and popup menus (see Section 4.4.2) of the main window
- option to switch off automatic table column resizing (see Section 4.4.3)
- the Log Window (see Section 5.3) shows now details of the focused (instead of selected) revision and allows zooming
- some improvements for Mac OS X (settings will now be stored in `~/Library/Preferences/SmartCVS/`; Cmd+W closes windows)
- the Tag Browser (see 4.4.6) shows the revisions by default with newer tags/branches on top and only those from child directories
- some improvements for the built-in file compare (refreshing, more status information, Find shows results more centered)
- the SmartCVS version number is now sent to the CVS server, so for example a script on the corporate CVS server can show a warning when not using the latest version

- Select Local File (e.g. from the Compare Repository Files window) can operate now on multiple selected files
- the directory chooser now allows to paste the path to select from the clipboard
- the Welcome dialog now has the option to create a project from an already checked out working copy
- double clicking a modified file compares automatically with the same revision from repository without showing a dialog
- the Directory Tree (see Section 4.1) now shows sticky information (e.g. the branch-name) in braces behind the directory name
- separate command to reset unchanged, touched files (see Section 4.4.4)
- all tables now are striped

Chapter 8

How To

8.1 Read-Only Repository Access

To make a password authenticated (pserver) repository read-only for some users you have two options, including and excluding.

To grant read-write access to only a predefined group of users, create the `CVSROOT/writers` file in the repository and put each user name who should have write access on a new line (each line must be terminated by a line separator).

To set only a few users' access rights to read-only, create the `CVSROOT/readers` file in the repository and put each user name who should only have read-only access on a new line (each line must be terminated by a line separator).

If both files are present, the `CVSROOT/readers` file is ignored.