TortoiseCVS User's Guide

Version 1.8.0

Ben Campbell

Martin Crawford

Hartmut Honisch

Francis Irving

Torsten Martinsen

lan Dees

Copyright © 2001 - 2004 TortoiseCVS

Table of Contents

1. Getting Started

What is CVS?

What is TortoiseCVS?

Where to Begin?

2. Basic Usage of TortoiseCVS

Sandboxes

Checking out a Module

Windows Explorer and TortoiseCVS

Total Commander and TortoiseCVS

Updating your Sandbox

Committing your Changes to the Repository

Resolving Conflicts

Adding Files and Directories to the Repository

3. Advanced Usage of TortoiseCVS

Creating a New Repository or Module

Watch, Edit and Unedit

Tagging and Labeling

Reverting to an Older Version of a File

```
Branching And Merging
               Creating a Branch
               Selecting a Branch to Work On
               Merging from a Branch
               Going Back to the Head Branch
       Binary and Unicode Detection
       File Revision History
               History Dialog
               Revision Graph Dialog
       Web Loa
       Making a Patch File
4. Customizing TortoiseCVS
       Overlay Icons
               Selecting a Different Set of Overlay Icons
               Changing how the Overlay Icons Work
Command Reference for TortoiseCVS
       Installing TortoiseCVS
       Obtaining a Working Copy: CVS Checkout...
       Getting Other People's Changes: CVS Update
       CVS Update Special...
       Making Your Changes Available to Others: CVS Commit...
       Adding New Files: CVS Add and CVS Add Contents...
       Discarding Obsolete Files: CVS Remove
       Finding Out What Has Changed: CVS Diff...
       Making a Snapshot: CVS Tag...
       Lines of Development: CVS Branch...
       CVS Merge...
       CVS Make New Module
       Watching And Locking
       Finding Out Who to Blame: CVS Annotate
       Showing More Information: CVS Explorer Columns
       Keyboard Shortcuts
       How Web Log Autodetects the Server URL
6. Dialog Reference for TortoiseCVS
       Add Dialog
       Checkout Dialog
       Update Special Dialog
```

Commit Dialog

Branch Dialog

Make New Module Dialog

Progress Dialog

Tag Dialog

Preferences Dialog

Merge Dialog

History Dialog

Revision Graph Dialog

About Dialog

7. Articles, Tips and Tricks

Using TortoiseCVS with SourceForge

How CVS Differs from Microsoft Visual Source Safe

Creating new icons for TortoiseCVS

Creating a new locale (translation) for TortoiseCVS

Using PuTTY sessions

Using Pageant to avoid having to enter the SSH password every time

8. Resources

List of Figures

- 2.1. Checkout Dialog
- 2.2. TortoiseCVS and Windows Explorer
- 2.3. Updating Files and Folders
- 2.4. Commit Dialog
- 2.5. Resolve Conflicts Dialog
- 2.6. Add Dialog
- 2.7. Add Contents Dialog
- 3.1. Make New Module Dialog
- 3.2. Make New Repository Question
- 3.3. Tag Dialog
- 3.4. Branch Dialog
- 3.5. Update Special Dialog
- 3.6. Merge Dialog
- 3.7. History Dialog
- 3.8. Revision Graph Dialog

- 3.9. Web Log Dialog
- 5.1. Annotate Dialog
- 5.2. Choosing which columns to display
- 6.1. Add Dialog
- 6.2. Checkout Dialog Module Tab
- 6.3. Checkout Dialog Revision Tab
- 6.4. Checkout Dialog Options Tab
- 6.5. Update Special Dialog
- 6.6. Commit Dialog
- 6.7. Progress Dialog
- 6.8. Tag Dialog
- 6.9. Preferences Dialog Main Tab
- 6.10. Preferences Dialog Policy Tab
- 6.11. Preferences Dialog Tools Tab
- 6.12. Preferences Dialog Advanced Tab
- 6.13. Preferences Dialog Appearance Tab
- 6.14. Preferences Dialog Cache Tab
- 6.15. Preferences Dialog Ignored Files Tab
- 6.16. Preferences Dialog Iconset Chooser
- 6.17. History Dialog
- 6.18. Revision Graph Dialog
- 6.19. About Dialog

List of Tables

- 2.1. Icon Overlays
- 2.2. Right-Click Context

Chapter 1. Getting Started

Table of Contents

What is CVS? What is TortoiseCVS? Where to Begin?

What is CVS?

CVS, or the Concurrent Versioning System, is a version control system. Version control systems are generally used as part of the software development cycle to track and co-ordinate source code changes among a team of developers.

For example, bugs sometimes creep in when software is modified, and you might not detect the bug until a long time after you make the modification. With CVS, you can easily retrieve old versions to see exactly which change caused the bug. This can sometimes be a big help.

You could of course save every version of every file you have ever created. This would however waste an enormous amount of disk space. CVS stores all the versions of a file in a single file in a clever way that only stores the differences between versions.

CVS also helps you if you are part of a group of people working on the same project. It is all too easy to overwrite each others' changes unless you are extremely careful. Some editors, like GNU Emacs, try to make sure that the same file is never modified by two people at the same time. Unfortunately, if someone is using another editor, that safeguard will not work. CVS solves this problem by insulating the different developers from each other. Every developer works in his own sandbox, and CVS merges the work when each developer is done.

What is TortoiseCVS?

TortoiseCVS is a front-end client to make using CVS easier and more intuitive. It allows developers to work with files controlled by CVS directly from Windows Explorer TM .

One of the major drawbacks of CVS is the command-line interface that is provided. Many developers today are becoming more accustomed to the graphical integrated development environments (IDEs). TortoiseCVS aims to provide that "point-and-click" environment in a clever and intuitive way.

Note that TortoiseCVS is a CVS *client*, not a *server*. This document assumes that you either know how to set up your own server, or that you are using a server that somebody else set up. If you want to learn about setting up a CVS server, see Chapter 8, *Resources*.

Where to Begin?

- 1. The best way to learn how to use TortoiseCVS is to play with it. Start by installing TortoiseCVS.
- 2. If you are new to CVS start following along with Basic Usage of TortoiseCVS.
- 3. Once you've learned the ropes check out the advanced features in Advanced Usage of TortoiseCVS.
- 4. For pure reference, the chapters Command Reference for TortoiseCVS and Dialog Reference for TortoiseCVS will help you get the most out of TortoiseCVS.
- 5. And finally checkout Articles, Tips and Tricks for a complete and enjoyable version control experience.

Chapter 2. Basic Usage of TortoiseCVS

Table of Contents

Sandboxes
Checking out a Module
Windows Explorer and TortoiseCVS
Total Commander and TortoiseCVS
Updating your Sandbox
Committing your Changes to the Repository
Resolving Conflicts
Adding Files and Directories to the Repository

Sandboxes

CVS has a unique method of working from most other version control systems in that developers can edit the same files concurrently. First you *Checkout* a version of the source code from the repository into a local copy on your computer. This local copy is called a *sandbox*.

You then simply edit the files that you want to change. You can *Add* new files or remove files you no longer require. When you're done you *Commit* the changes to the repository.

If someone else has changed the same file while you were working on it, then the commit will fail. You must then *Update* all your source code files from the repository. This will automatically merge the other developers changes into your copy of the file.

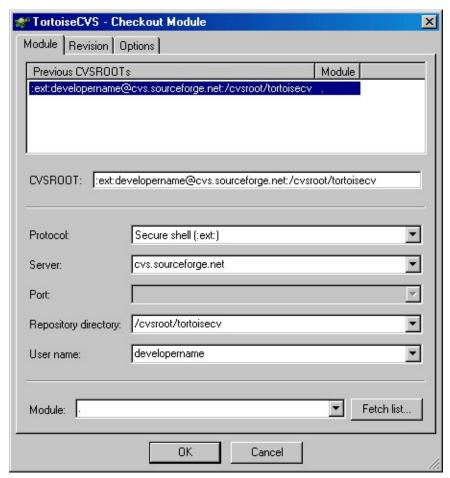
Sometimes CVS cannot do this automatically, for example if you both changed the same line of code. This is called a *Conflict*. Conflicts happen much less often than you might expect. CVS puts both versions of the conflicting code in the file, with markings separating them. Then you manually edit the file to resolve the conflict before you can commit the changes.

This method of working has lots of advantages. Each developer lives in a sandbox. Changes that another developer makes are isolated from you until you want to check in your changes. It stops bottlenecks where people cannot do things because someone else has the file checked out. Any developer can work on files without direct access to the server, they only need to connect to update or commit.

Checking out a Module

To obtain a module from the CVS server for the first time, is known as a checkout. Checking a module out from the repository creates a local sandbox of the module.

Figure 2.1. Checkout Dialog



To perform a checkout, Right-click on the folder where you would like the module placed, and pick **CVS Checkout...** from the pop-up menu. The **Checkout Dialog** will appear with the following fields:

- Protocol The protocol to use when communicating to the remote CVS repository.
- Server The name of the server hosting the remote CVS repository.
- Port The port for the remote CVS repository. Not usually required.

- Repository Directory The location of the CVS repository on the remote server.
- User name The username of your account for the CVS repository.
- **CVSROOT** The full connection string, comprised of the above fields. Often you will be given the connection string in this format, in which can just paste it in here.
- **Module** The name of the module you want to checkout. The module name is case sensitive.

You will need to know this information in advance. Most projects should have some documentation (often online) on how to connect and checkout their modules. Additionally, your software lead, or project manager should be able to provide this information to you.

Note: It is much easier to choose your project checkout settings now than it is to move your project to a different server or protocol later. For example, if your server offers two different CVS protocols, make sure you pick the one that best suits your needs

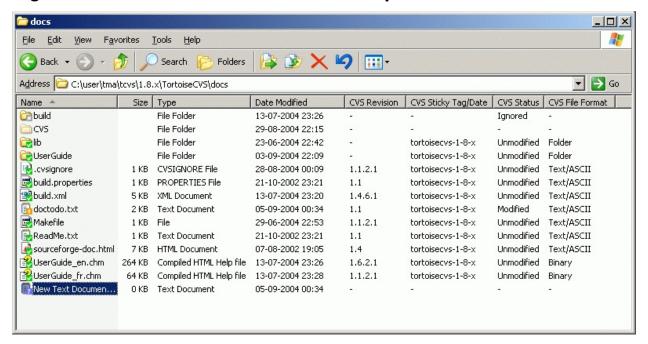
A folder named after the module will be created within the folder you checkout to, so you can keep all your checked out modules in the same folder, even if they are for different projects.

For more information see Obtaining a Working Copy: CVS Checkout...

Windows Explorer and TortoiseCVS

Having checked out a module, we can now *explore* how TortoiseCVS works with Windows[™]. You will notice that your files appear in Windows Explorer with different icon overlays. Additionally, if you are using either Windows 2000[™] or Windows XP[™] you can also see new CVS columns in the *Detail View* of Windows Explorer[™] (see Showing More Information: CVS Explorer Columns).

Figure 2.2. TortoiseCVS and Windows Explorer



The icon overlays indicate a file or folder's status in CVS. The following figure indicates the corresponding status for each icon^[1]:

Table 2.1. Icon Overlays

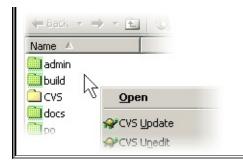
| u inmontalia | The file or folder is up-to-date with the CVS repository version. |
|---------------|---|
| IIVIOAITIEA I | The file or folder has been modified from the current CVS repository version. |
| | |

| | | The file or folder has been added to CVS, but not yet committed. |
|------------|---------------------------|--|
| | IC ontlict | The file or folder has a conflict with the current CVS repository version. |
| 1 ? | Not In CVS | The file or folder is not in the CVS repository. |
| | CVS Watch Icon Overlay | The file or folder is controlled under CVS Watch. |
| | Ignore Icon Overlay | The file or folder is being ignored by CVS. |

You interact with TortoiseCVS by right-clicking within Windows Explorer and choosing CVS operations from the context menu. Which files and folders the operation is performed on depends on what you have selected and where you have clicked:

Table 2.2. Right-Click Context

| development.shtml Open Faq. CVS Update CVS Unedit CVS Commit | One or more selected files, the operations is performed on those files. |
|--|---|
| docs po pen src we CVS Update CVS Unedit CVS Commit | One or more selected folders, the operations is performed on these folders and the files or folders contained within. |
| | |



Anywhere else within the Explorer pane, the operation is performed the operation on all files and folders within the view.

[1] The above table shows the default icon overlays; TortoiseCVS may be configured to use alternative icon sets — see Overlay Icons.

Total Commander and TortoiseCVS

The popular Total Commander™ (previously known as Windows Commander) file manager also works with TortoiseCVS. We recommend that you go into the options menu and configure the right mouse button for Windows standard operation.

Updating your Sandbox

Occasionally you may want changes done by others to get incorporated in your local working copy. The process of getting changes from the server to your local copy is known as Updating. Updating may be done on single files, a set of selected files, or recursively on entire folder hierarchies. To update, highlight the files and/or folders you want, right-click and select **CVS Update**. The Progress Dialog will pop up displaying the progress of the update as it runs.

Figure 2.3. Updating Files and Folders



Changes done by others will be merged into your files, keeping any changes you may have done to the same files. The repository is *not* affected by performing an update.

If you receive reports of conflicts during the update, please read Resolving Conflicts.

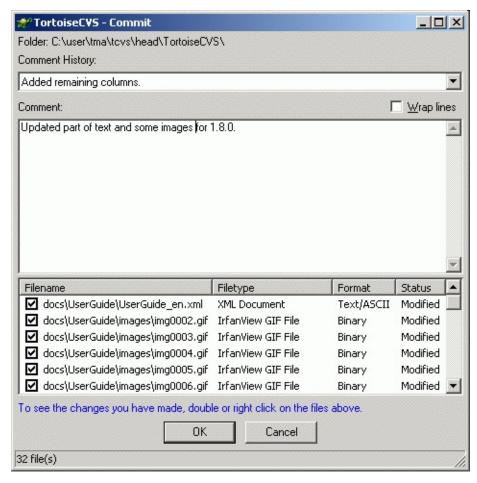
For more information see Getting Other People's Changes: CVS Update.

Committing your Changes to the Repository

Making local modifications available in the repository is known as *committing* the changes. Before committing, you should do an update to make sure there are no conflicts (see Updating your Sandbox).

To commit your changes start by selecting the file(s) or folder(s) that you want to commit. Right-click on the selection, and choose the **CVS Commit...** menu item.

Figure 2.4. Commit Dialog



You will then be presented with the Commit Dialog where you can enter a brief summary of what was changed. You can also exclude changed files from the commit by unchecking their checkboxes. Once you are satisfied with what you are committing, click the **OK** on the dialog to go ahead with the commit operation.

Please note that committing changes will not automatically add new files that you have created to the repository. See Adding Files and Directories to the Repository on how to add files.

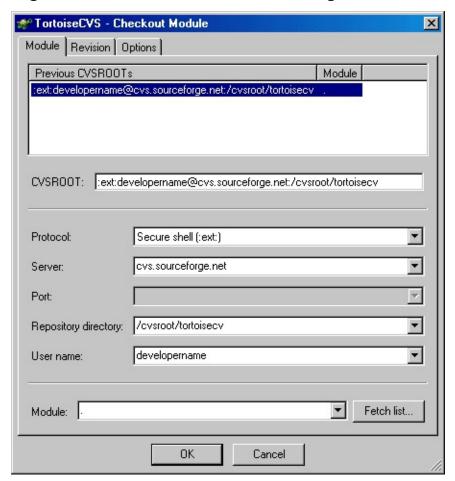
For more information see Making Your Changes Available to Others: CVS Commit....

Resolving Conflicts

Sometimes, the CVS server will report a conflict when you update your files from the repository. A conflict occurs when two or more developers have changed the same lines of a file. As CVS knows nothing of your project, it leaves resolving the conflicts to the developers.

Each conflicting file is marked with a "C" in the progress dialog (see Progress Dialog). When the update is complete, the Resolve Conflicts dialog is shown:

Figure 2.5. Resolve Conflicts Dialog



Right-click on the conflicting file(s) and select **Merge Conflicts...**.

TortoiseCVS will now invoke the merge application that you have selected in Preferences/Tools/Merge application. Resolve each conflict, save the resulting file, and exit the merge application. TortoiseCVS will

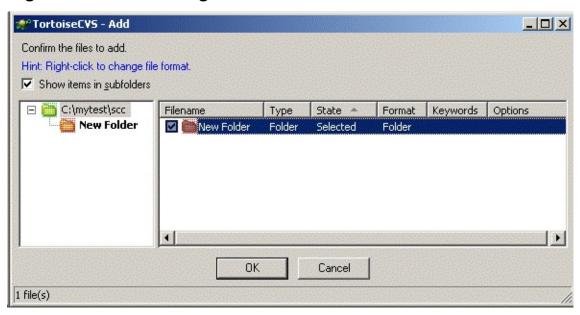
ask you to confirm that you want to save the result of the merge.

You can also bring up this dialog after closing the progress dialog by right-clicking in the sandbox and selecting the $CVS \rightarrow Resolve$ Conflicts... menu item.

Adding Files and Directories to the Repository

If you have made new files or directories you will notice that they appear with the *Not In CVS* status icon overlay (see Windows Explorer and TortoiseCVS). To put new files or folders under CVS control select the item(s) that you want to add, right-click and choose **CVS Add** from context menu to schedule the addition. The Add Dialog is displayed so you can verify the file you are adding.

Figure 2.6. Add Dialog

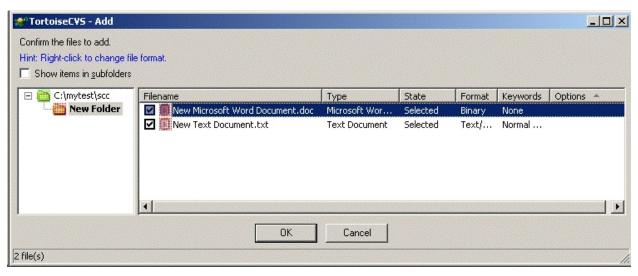


You do not need to worry about whether a file is ASCII/Text, Unicode/Text, or Binary as TortoiseCVS automatically detects this. For more on how TortoiseCVS handles this see Binary and Unicode Detection.

After an add operation has been performed, the file or files icons appear as "changed". This is because additions are treated as local changes and are not applied to the repository until you *Commit* them.

Additionally, you can add a number of files and folders to CVS using the CVS Add Contents... command. This operation recursively descends down the folder structure, and displays all unadded files in the Add Dialog. Here you can check and uncheck the files and folders you wish to add to CVS.

Figure 2.7. Add Contents Dialog



For more information see Adding New Files: CVS Add and CVS Add Contents....

Chapter 3. Advanced Usage of TortoiseCVS

Table of Contents

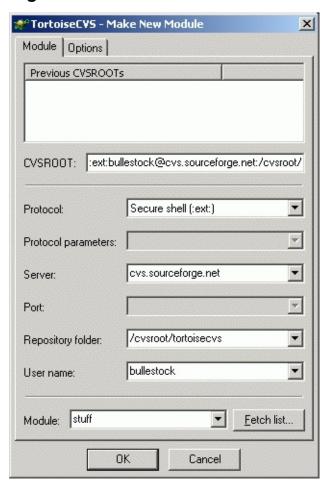
Creating a New Repository or Module
Watch, Edit and Unedit
Tagging and Labeling
Reverting to an Older Version of a File
Branching And Merging
Creating a Branch
Selecting a Branch to Work On
Merging from a Branch
Going Back to the Head Branch
Binary and Unicode Detection
File Revision History
History Dialog
Revision Graph Dialog
Web Log
Making a Patch File

Creating a New Repository or Module

Typically a System Administrator will setup the CVS repository on a remote server. TortoiseCVS, however, can configure repositories and modules either locally or remotely (if you have to correct permissions).

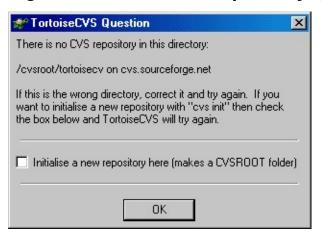
To create a new repository and/or module prepare the folders and files locally as a new sandbox. Right-click on the top level folder for your new module or repository and choose the CVS → Make New Module... menu item. The Make New Module Dialog will be displayed. Enter the details for the new or existing CVS repository as you would for a *Checkout*. Click OK to create the new module.

Figure 3.1. Make New Module Dialog



If the remote CVS server could be reached but the CVS repository does not exists, you will be presented with the following question:

Figure 3.2. Make New Repository Question



To create the new repository enable the checkbox **Initialise a new repository here** and click **OK**. If you have the right permissions on the remote server TortoiseCVS will setup the new repository.

Once TortoiseCVS has either created a new or connected an existing CVS repository the new module will be created. By default TortoiseCVS uses the name of the selected folder for the new module's name, but you can change this by modifying the **Module** textbox.

Watch, Edit and Unedit

CVS can also follow the semantics of *Lock/Unlock* that most version control systems use, although CVS calls it *Edit/Unedit*. This feature, however, is not enabled in CVS by default so before you can begin you need to enable *Watch*.

You can enable Watch using TortoiseCVS when you *Make a New Module* by enabling the checkbox **Check files read-only** (on the **Options** tab). If you wish to enable watch on an existing module, you will need run the following command (using the Command Prompt) on the top level of the module:

cvs watch on

Once Watch is enabled, when performing *Checkout* or *Update* the working files will be created as read-only. To change one or more files, select the directory, file or files that you want to modify, right-click on the selection, and choose the **CVS Edit** menu item. This command works slightly different depending on whether the file is text or binary. For binary files (such as .doc), TortoiseCVS performs an *Update* on the file and then checks if another user is already editing the file. If this is the case, you will not be allowed to edit it. For text files (such as .txt or .cpp), you may work on the file even if another user is already working on it; in this case, TortoiseCVS will notify you of that fact.

When you are done working on the file or files, *Commit* them as you would normally. If, however, you decide that you do not want to make changes select the file or files: Right-click on this selection and choose the **CVS** → **Unedit** menu item. TortoiseCVS will ask you if you want to revert the file to the original version (thus losing your changes).

If you would like to see which files are currently being edited and by whom, right-click anywhere in Windows Explorer and choose the CVS → Show edited files menu item.

Tagging and Labeling

At a given stage of development, giving one or more files a common label that refers to their revisions, is known as tagging those files. Tagging is typically used on entire modules, so that the current state of the module can be reconstructed in the future. This kind of tagging should always be done on project deliverables, and before starting major changes.

To tag one or more files or directories with a label, select the directory, file or files that you want to tag. Right-click the mouse button on the selection, and choose the **CVS** → **Tag...** menu item.

Figure 3.3. Tag Dialog



You will then be presented with the Tag Dialog. Here you can enter a label in the Tag field. CVS is quite restrictive when it comes to what characters a tag may contain. A tag must start with a letter, and may contain letters, digits, "-" (dash) and "_" (underscore) only. In particular, this means no dots, and no spaces. If you want to include version numbers in a tag, replace the dots with dashes. Two tag names are reserved, as they have special meaning in CVS: "HEAD" refers to the most recent version available in the repository, while "BASE" is the revision you last checked out into the local directory.

Once the label has been entered, press the **OK** button. Unlike most CVS operations, the tag is immediately applied to the repository, and no commit is required.

For more information see Making a Snapshot: CVS Tag....

Reverting to an Older Version of a File

Developers occasionally need to undo changes that have already been checked in. Going back to a specific file revision in TortoiseCVS is easy:

- 1. Right-click on the file and select CVS → History to bring up the History Dialog.
- 2. Right-click on the revision you need and choose the "Save this revision as..." option.
- 3. When the Save As dialog appears, do not click the "Save" button; instead, double-click on the file's name in the directory listing.
- 4. Answer "Yes" when TortoiseCVS prompts you to overwrite the file.

Branching And Merging

One of the features of version control systems, is the ability to isolate changes onto a separate line of development. This line is known as a *Branch*.

Branching is useful to control changes during the lifecycle of a software project. For example, suppose you have made the first release of your software project version 1.0. You now begin adding new features in preparation for your next major release 2.0. Between your initial release and the new version it is discovered that there are bugs in the software. The current revision of the sources code is in a state of flux and are not expected to be stable for at least another month and therefore there is no way to make a bugfix release based on the newest sources.

Instead of attempting to make the fix to the current version, or the *HEAD* branch, you would create a branch on the revision trees for all the files that made up release 1.0. You can then make modifications to the branch without disturbing the head branch. When the modifications are finished you can either incorporate them on the head branch or leave them on the release 1.0 branch.

Creating a Branch

To create a branch, select the directory or files that you want to branch. Right-click the on the selection, and choose $CVS \rightarrow Create Branch...$ from the context-menu.

Figure 3.4. Branch Dialog



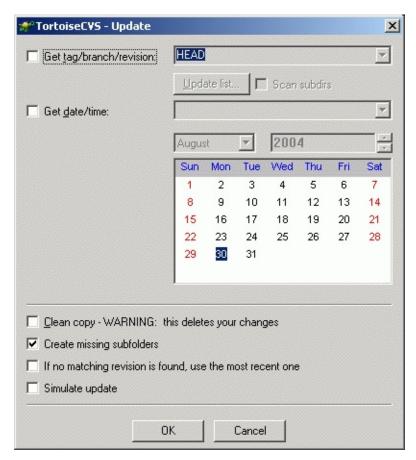
You will then be presented with the Branch Dialog. Here you can enter a label in the **Branch** field. The same name restrictions that apply to *Tags*, apply to branches as well. Once you have entered the desired branch name click the **OK**. Unlike most CVS operations, the branch is immediately applied to the repository, and no commit is required. The branch, however, is only applied on the repository. To start working on the newly created branch select the branch to work on.

Selecting a Branch to Work On

To start working on a branch instead of the default development line, you have to bind your local copy to the branch. This is needed to make sure that actions such as updates, commits etc. work on the branch rather than on the main line of development.

To move your local copy to another branch, select the top level folder of the project. You can also select the exact folders and files that are part of the desired branch if you know this information. Right-click the on the selection, and choose CVS → Update Special....

Figure 3.5. Update Special Dialog



You will then be presented with the Update Special Dialog. Here you can enter the branch name in the Get Tag/Branch/Revision field you wish to select. Click the OK button, and TortoiseCVS will now do the necessary updates to move your working copy to the desired branch. The updating may also include adding or removing files depending on the state of the branch.

TortoiseCVS puts what is known as sticky tags on the files that are affected by the branch. To remove the sticky tags you must go back to the head branch.

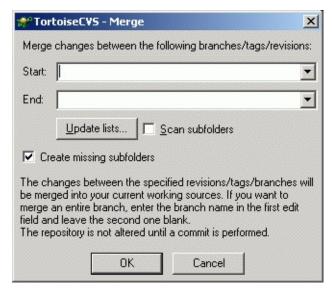
Merging from a Branch

When you are satisfied with the changes you have done on a branch, you may want those changes to be available on the head branch of development. Incorporating changes from one branch to another, is known as merging.

To merge from a branch, move your local copy to the branch you want to

merge the changes into. See Selecting a Branch to Work On or Going Back to the Head Branch. Select the top level folder of the project. You can also select the exact folders and files that are part of the desired branch if you know this information. Right-click the on the selection, and choose CVS \rightarrow Merge....

Figure 3.6. Merge Dialog



You will then be presented with the Merge Dialog. Here you can enter the branch name in the **Branch to merge from** field you wish to merge. Click the **OK** button, and TortoiseCVS will now merged the branch specified into your local copy. Your changes will not be made on the server repository, until you commit your changes.

The merge given above will try to merge changes from the start of the branch. If you do the operation a second time (to merge changes done to the branch after the last merge), merging from the start of the branch is not what you want, and it will most likely get you into trouble. To get around this problem, you should give the branch a new tag after every merge, and use the new tag when naming the branch for subsequent merges.

Note: The above paragraph only applies for standard Unix CVS (aka cvshome.org CVS). If your server runs CVSNT, you can take advantage of a special feature known as *merge points*. This means that CVS keeps track of your last merge, so that you can effortlessly merge from the same branch repeatedly. For more information on this feature, see the

CVSNT Wiki page (note that the CVSNT Manual is not yet up to date on this point).

Going Back to the Head Branch

If you want to stop working on a branch and move your local copy back to the main line of development, you have to make TortoiseCVS remove all sticky tags.

To remove the sticky tags, and thus update your local copy to the main development line, select the top level folder of the project. You can also select the exact folders and files that are part of the desired branch if you know this information. Right-click the on the selection, and choose CVS → Update Special....

You will then be presented with the Update Special Dialog. Select the Return to main (HEAD) branch checkbox and click the OK button. TortoiseCVS will now do the necessary updates to move your working copy back to the head branch.

Binary and Unicode Detection

TortoiseCVS tries to automatically detect the type of file you are adding to CVS. It can detect whether the file is <code>Text/ASCII</code>, <code>Text/Unicode</code>, or <code>Binary</code>. TortoiseCVS first checks the file extension to determine whether the file is Binary or Text. For example <code>.doc</code> and <code>.exe</code>, are always assumed to be Binary while other extensions, such as <code>.cpp</code> and <code>.txt</code>, are always assumed to be text. TortoiseCVS does this using a built-in list of Binary and Text extensions which includes most the common file extensions. To customize this list, you can copy the file <code>TortoiseCVS.FileTypes</code> from the TortoiseCVS installation directory to your HOME directory and edit it for your needs.

By default TortoiseCVS will also examine the first 4000 bytes of a file to determine the type. This is used in cases where the file extension is not known. These first few bytes are also used to determine whether the file is Text/ASCII or Text/Unicode.

TortoiseCVS also provides a sophisticated plugin mechanisim that you can implement a DLL to perform the file type detection. The details of this plugin interface is also available in the FileTypes.config.

In all cases, the Add Dialog presents the files you are about to add their file types, so that you have a chance to correct the choices made.

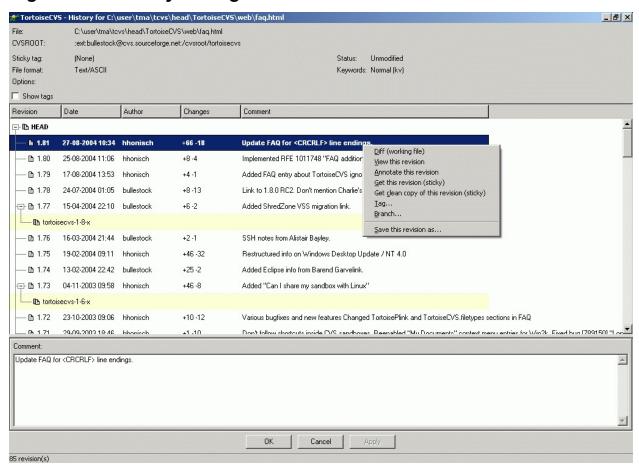
File Revision History

TortoiseCVS provides two methods of reviewing and working with the history of a file. The first is via the History Dialog.

History Dialog

You can view a file's history by selecting it and right-clicking $\textbf{CVS} \rightarrow \textbf{History}$.

Figure 3.7. History Dialog



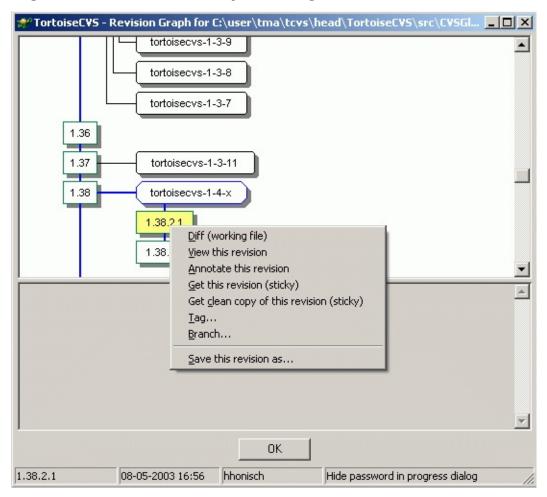
The History Dialog will show a log of the revisions, tags, and branches for the selected file. It will also offer you the opportunity, by right-clicking on a specific revision, tag, or branch, to *diff*, *save* the revision locally, or *update* with sticky tags (an advanced CVS concept; usually, the *save* option is preferable to a sticky *update*). You can also edit any current or previous commit message by clicking on a revision and typing into the

text area at the bottom of the window (remember to click Apply before moving to a new revision).

Revision Graph Dialog

TortoiseCVS also provides a Revision Graph Dialog to graphically show you the history of a file. You can view a file's revision graph by selecting it and right-clicking CVS → Revision Graph.

Figure 3.8. Revision Graph Dialog



Like the History Dialog, the Revision Graph Dialog will allow you to see the details for and perform CVS operations on any revision, tag, or branch.

If your CVS server runs CVSNT, you will also be able to see which merge operations have been performed on the file: A purple arrow is drawn from the source revision to the destination revision (see Mergepoints).

Web Log

Web Log offers much of the same functionality as the History and Revision Graph dialogs. One important difference is that Web Log is not part of TortoiseCVS itself, but depends on special software that may be installed on the CVS server you use. Two popular packages for providing Web Log functionality are ViewCVS and CVSweb — ask your CVS administrator if one of these is available.

To view the Web Log, you open a special URL using a web browser. TortoiseCVS knows the most common URLs in use for this and can in most cases detect the correct one automatically^[2]. If your configuration is nonstandard, you may have to enter the URL manually the first time - next time TortoiseCVS will remember it.

You can view Web Log for a file or folder by selecting it and right-clicking $CVS \rightarrow Web Log$. If TortoiseCVS fails to detect the URL, the dialog below will appear.

Figure 3.9. Web Log Dialog



Enter the correct URL and click OK. If the URL of the Web Log server has changed, you may have to tell TortoiseCVS to perform the automatic scan again.

[2] For details about the automatic URL detection, see How Web Log Autodetects the Server URL.

Making a Patch File

Often you might have read access to a CVS repository or module, but not write access. Open source projects typically work this way to maintain quality control for all contributions to a project. The easiest and most reliable way to contribute changes to these projects is by creating a *Patch File* to submit to them.

In TortoiseCVS this is done by selecting the folders or files with changes that you wish to include in the patch file. Right-click on your selections and choose CVS → Make Patch... from the context menu. You will then be prompted with a Save dialog to indicate where to save your patch file. Finally, TortoiseCVS will open the new patch file in Notepad, where you can validate your changes.

Sometimes your contribution involves adding new files, which are not yet in CVS. TortoiseCVS cannot guess that you would like these to be included in the patch, but in the Save dialog you can click the **Browse** button to bring up the Add dialog. Here you can select files to be included in the patch.

Chapter 4. Customizing TortoiseCVS

Table of Contents

Overlay Icons
Selecting a Different Set of Overlay Icons
Changing how the Overlay Icons Work

Overlay Icons

Selecting a Different Set of Overlay Icons

TortoiseCVS comes with several sets of overlay icons. You might prefer one of the alternative icon sets to the default one; open the Preferences Dialog - Iconset Chooser Iconset Chooser dialog to select another icon set. Regrettably, Explorer caches the overlay icons, which means that you have to restart Windows to see the effect of the change. The dialog shows a preview of the icons to help you choose the correct one.

Changing how the Overlay Icons Work

By default, the overlay icon for a folder will indicate "modified" if any of the files in the folder are modified. It is possible to configure TortoiseCVS to indicate "modified" if any of the files in the folder *or in any subfolders* are modified.

As updating this status can easily be resource-intensive, TortoiseCVS offers the possibility of either updating the status each time you open the folder, or only on demand. In the latter case, the icons will show the cached status, and you will have to invoke $CVS \rightarrow Refresh Folder$ Status to update the status.

Go to Preferences Dialog - Advanced Tab and change the **Folder icon overlays** setting as you desire.

Chapter 5. Command Reference for TortoiseCVS

Table of Contents

Installing TortoiseCVS

Obtaining a Working Copy: CVS Checkout...

Getting Other People's Changes: CVS Update

CVS Update Special...

Making Your Changes Available to Others: CVS Commit...

Adding New Files: CVS Add and CVS Add Contents...

Discarding Obsolete Files: CVS Remove

Finding Out What Has Changed: CVS Diff...

Making a Snapshot: CVS Tag...

Lines of Development: CVS Branch...

CVS Merge...

CVS Make New Module

Watching And Locking

Finding Out Who to Blame: CVS Annotate

Showing More Information: CVS Explorer Columns

Keyboard Shortcuts

How Web Log Autodetects the Server URL

This chapter is written to illustrate most main features and functionality of TortoiseCVS. It assumes that a) you already have an understanding of CVS and b) you have access to an already installed a CVS server with created a CVS repository. If you are new to CVS then please refer to the section "TortoiseCVS for Beginners."

Installing TortoiseCVS

- Download the latest version of TortoiseCVS from the web site (http://tortoisecvs.org/) and save to a temporary location on your hard drive.
- 2. Run the installer. If you are running Windows NT or 2000 you must have local admin privileges to install TortoiseCVS correctly.
- 3. Choose the location for TortoiseCVS to be installed. The default is C:\Program Files\TortoiseCVS.
- 4. Choose the installation type, either **Full**, **Compact**, or **Custom installation**. If you desire internationalisation support, you will need either the **Full** or **Custom installation** options.
- 5. Once the installer has completed you must reboot the computer for TortoiseCVS to finish its installation^[3].

[3] Strictly speaking, it is only necessary to restart Windows Explorer. If you are feeling adventurous, you can try killing explorer.exe from Task Manager. In some cases WindowsTM will automatically start a new Explorer instance, in other cases you will have to do it yourself.

Obtaining a Working Copy: CVS Checkout...

Abstract

Creates a local sandbox from a CVS repository.

- 1. Create a folder for the sandbox in Windows Explorer.
- 2. Right-click on the folder.
- 3. Select **CVS Checkout...** from the context-menu.
- The Checkout Dialog will appear. From here you can either enter the repository's CVSROOT into the CVSROOT field, or construct a CVSROOT using the Protocol, Server, Port, Repository name, and User name fields.
- 5. If you know the module name you can enter it in the **Module** field. Conversely, you can checkout an entire repository by entering a period (.) into the **Module** field. If you do not know the module name you can click on **Fetch list** to populate the drop-down with available modules. Note: Some CVS servers do not support this feature; if you cannot populate this list you should contact the CVS server administrator.
- 6. Click **OK** to being checking out the sandbox.
- 7. Depending on the protocol and if this is the first time checking out from this CVS repository, you will be prompted to enter a password. Enter the password in the dialog and click **OK**.
- 8. The Progress Dialog will appear, which provides a detail status of what files are being checked out by TortoiseCVS. By default this dialog will close after the operation is complete, unless it encounters a problem.

Getting Other People's Changes: CVS Update

Abstract

Synchronizes a local sandbox with a CVS repository. See also Update Special.

- 1. To update a folder either right-click on it or within it on the Explorer view (but not on any specific files) or to update one or more files select the files to update and right-click on them.
- 2. Select **CVS Update** from the context-menu.
- 3. The Progress Dialog will appear to provide a detail status of the files that are being updated by TortoiseCVS. By default this dialog will close after the operation is complete, unless it encounters a problem.

CVS Update Special...

Abstract

Synchronizes a local sandbox with a CVS repository. See also Update.

- 1. To update a folder either right-click on it or within it on the Explorer view (but not on any specific files) or to update one or more files select the files to update and right-click on them.
- 2. Select CVS → CVS Update Special... from the context-menu.
- 3. The Update Special Dialog will appear. Enter into the textbox labeled **Get tag/revision/branch:** either the Revision number (i.e. 1.3), Tag (i.e. release_1_1_0), or Branch (i.e. vendorx_branch) to change your sandbox copy to.
- 4. The Progress Dialog will appear to provide a detail status of the files that are being updated by TortoiseCVS. By default this dialog will close after the operation is complete, unless it encounters a problem.

Making Your Changes Available to Others: CVS Commit...

Abstract

Update the CVS repository with your local changes.

- To commit changes either right-click on the desired folder and/or files.
- 2. Select **CVS Commit...** from the context-menu.
- 3. The Commit Dialog will appear, which will allow you to enter a comment for the commit action. Additionally, you can choose which changes to commit by checking the individual files on the dialog list.
- 4. Click **OK** (or press **Ctrl-Enter**) once you have entered the appropriate comments and selected the changes to commit.
- The Progress Dialog will appear to provide a detail status of the committing process performed by TortoiseCVS. By default this dialog will close after the operation is complete, unless it encounters a problem.

Adding New Files: CVS Add and CVS Add Contents...

Abstract

Add local files and/or folders to the CVS repository.

- To add files or folders to the CVS select and right-click the files or folders you wish to add. These files must have the CVS Status of Not In CVS.
- 2. If you have selected a folder you will be provided the option to **Add Recursively** in the context-menu. This option will add all files in the selected folder and its sub-folders. Otherwise, to only add those files and folders selected select **Add** from the context-menu.
- 3. The Progress Dialog will appear to provide a detail status of the adding process performed by TortoiseCVS. By default this dialog will close after the operation is complete, unless it encounters a problem.

Discarding Obsolete Files: CVS Remove

Abstract

Remove a file from the current branch. The file will still be available when retrieving an older revision or tag.

- 1. To remove one or more files from the CVS repository select the desired files and/or folders to remove and right-click on them.
- 2. Select **CVS** → **Remove** from the context-menu. TortoiseCVS performs the removal and shows a Tortoise Tip that reminds you that you must commit the removal to finish the process.
- 3. Finally, invoke the **CVS Commit** command on the folder (or a parent folder).
- 4. When another user now performs an Update, CVS will remove the file(s) from the sandbox.

Finding Out What Has Changed: CVS Diff...

Abstract

Compare a local file to a version in the CVS repository.

- 1. To compare a file to a different version from the CVS repository, select the desired file and right-click on it.
- 2. Select **CVS** → **Diff** from the context-menu.
- 3. If this is the first time you have used TortoiseCVS's Diff feature, you will be asked to select an external diff application. WinMerge and ExamDiff are both available free of charge and are well-suited for text-based source code. If you are developing in the LabVIEW programming language, you can use LVDiff to compare your block diagrams visually. CSDiff is designed to compare different versions of Word documents. To use a combination of diff tools based on the file type you are comparing, try meta-diff.
- 4. TortoiseCVS will launch the diff tool you have chosen and compare your local copy of the selected file with the version from the repository. For more sophisticated diff options (such as comparing to an older repository copy or comparing two older versions to each other), see File Revision History.

Making a Snapshot: CVS Tag...

Abstract

Make a snapshot of a project at a given point in time for later retrieval.

- 1. To create a tag, select the desired files and/or folders to remove and right-click on them. Note: A tag is normally applied to a folder, usually the top folder of the sandbox.
- 2. Select **CVS** → **Tag** from the context-menu.
- 3. Enter the desired tag name. Note that CVS places some constraints on the tag name; these constraints are shown at the bottom of the dialog. To see which tags are already used in this folder, click the **Update list...** button.
- 4. By default, the **Check that the files are unmodified** checkbox is enabled. This prevents you from applying the tag if you have uncommitted changes. Under special circumstances, you might want to disable this.
- 5. If the tag already exists, CVS will not move it unless you select **Move** existing tag.
- 6. To remove a tag which you applied by mistake, or no longer want, select **Delete existing tag**.
- 7. Finally, click **OK**.

Lines of Development: CVS Branch...

Abstract

Create a branch (aka fork) for a separate line of development.

- To create a branch, select the desired files and/or folders to remove and right-click on them. Note: A branch is normally created on an entire folder, usually the top folder of the sandbox.
- 2. Select **CVS** → **Branch** from the context-menu.
- 3. Enter the desired branch name. Note that CVS places some constraints on the branch name; these constraints are shown at the bottom of the dialog. To see which branches are already used in this folder, click the **Update list...** button.
- 4. By default, the **Check that the files are unmodified** checkbox is enabled. This prevents you from creating the branch if you have uncommitted changes. Under special circumstances, you might want to disable this.
- 5. If the branch already exists, CVS will not move it unless you select **Move existing branch**.
- 6. To remove a branch which you created by mistake, or no longer want, select **Delete existing branch**. You should be very careful when doing this.
- 7. Finally, click **OK**.
- 8. Note that your sandbox is not automatically changed so that you are working on the newly created branch. To do this, see Selecting a Branch to Work On.

CVS Merge...

Abstract

Update one branch with changes from another branch.

Note

TODO: Write...

CVS Make New Module

Abstract

Create a new module (project) in the CVS repository.

Note

TODO: Write...

Watching And Locking

Abstract

Track who is editing which files.

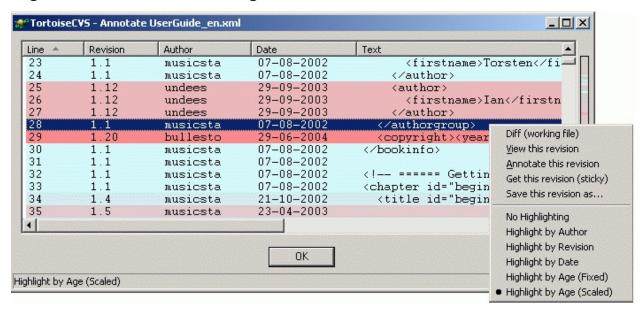
Note

TODO: Write...

Finding Out Who to Blame: CVS Annotate

When something suddenly stops working, it can be valuable to find out exactly who is to blame for the change. This is what **CVS Annotate** is for.

Figure 5.1. Annotate Dialog



For each line of the file, the dialog shows

- 1. From which revision that line originates (e.g. lines marked as 1.1 have not been changed since the first revision of the file).
- 2. The author of the line, i.e. the person who made the last change.
- 3. The time and date when the line was last changed.
- 4. The current content of the line.

If you right-click a line in the dialog, you have the option of launching various commands, including viewing an annotation for a particular revision.

Also, you can change the appearance of the dialog by choosing between various highlighting options.

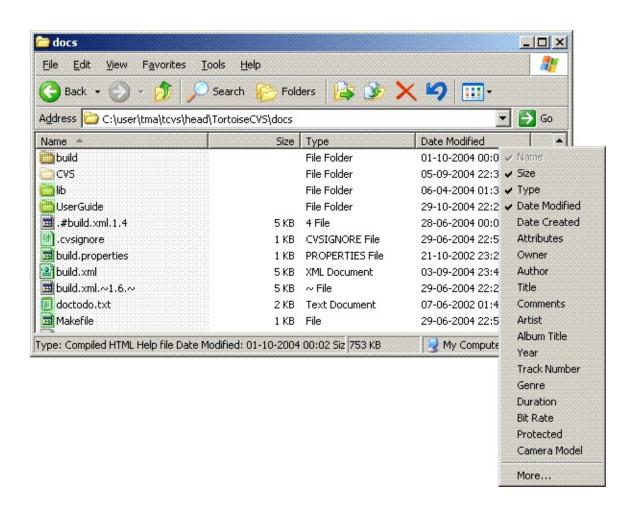
Showing More Information: CVS Explorer Columns

In Details view, TortoiseCVS can show CVS-specific columns in the Explorer view (only on Windows 2000 and later). The available columns are

- CVS Revision
- CVS Sticky Tag
- CVS Status
- CVS File Format

To select which columns to display, right-click on the header as shown in the figure below. Select **More...**, and you will see a dialog listing all the available columns on your system. Scroll down to find the "CVS" columns and select the ones you want.

Figure 5.2. Choosing which columns to display



Keyboard Shortcuts

TODO: This section will explain how to use TortoiseCVS without the mouse.

How Web Log Autodetects the Server URL

As mentioned in Web Log, TortoiseCVS can in many cases automatically detect the URL used by your ViewCVS or CVSweb server. It does this by using the following algorithm:

- 1. Start with the name of the CVS server, with a http://prefix.
- 2. Append one item from the list
 - ∘ /cgi-bin
 - o /viewcvs
 - o /cvsweb
 - ∘ /bin
 - (the empty string)

and an item from the list

- ∘ /viewcvs.cgi
- ∘ /viewcvs
- ∘ /cvsweb.cgi
- o /cvsweb.pl
- o /cvsweb
- 3. Append the relative path for the file or folder (i.e. the value of 'Repository' shown in the CVS Properties dialog).
- 4. Check if the computed URL works. If not, add ?cvsroot= followed by the name of the module and try again.

The above steps are done for each possible permutation of the different prefixes and suffixes. If a URL is found which returns HTTP status 200 (OK), the search stops. If no such URL is found, and an URL is found which returns HTTP status 401 (Denied) or 403 (Forbidden), that URL is used (based on the assumption that the user will be allowed access after authenticating to the web server).

Chapter 6. Dialog Reference for TortoiseCVS

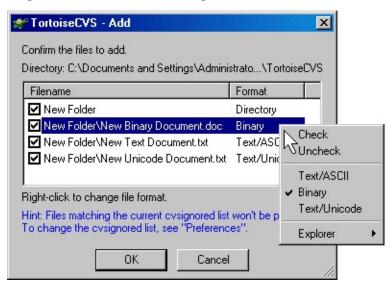
Table of Contents

Add Dialog
Checkout Dialog
Update Special Dialog
Commit Dialog
Branch Dialog
Branch Dialog
Make New Module Dialog
Progress Dialog
Tag Dialog
Preferences Dialog
Merge Dialog
History Dialog
Revision Graph Dialog
About Dialog

This chapter details every dialog within TortoiseCVS.

Add Dialog

Figure 6.1. Add Dialog



Note

Checkout Dialog

Figure 6.2. Checkout Dialog - Module Tab

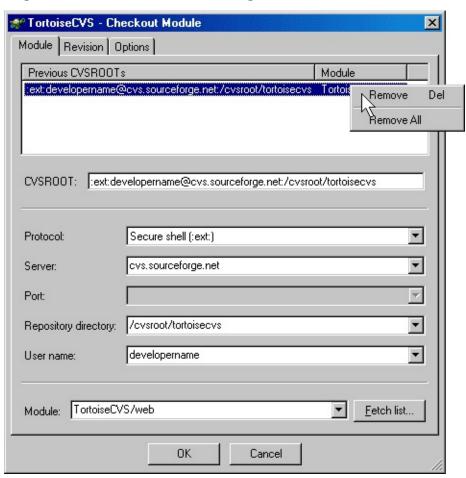


Figure 6.3. Checkout Dialog - Revision Tab

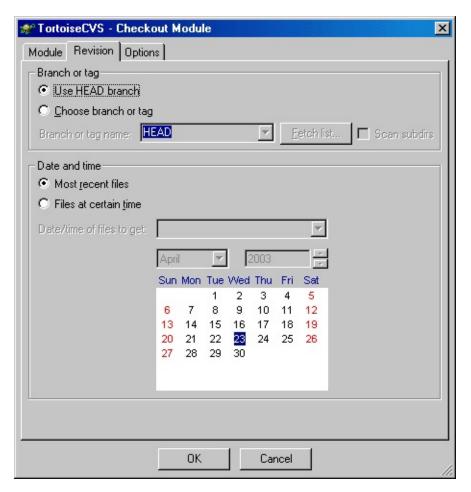
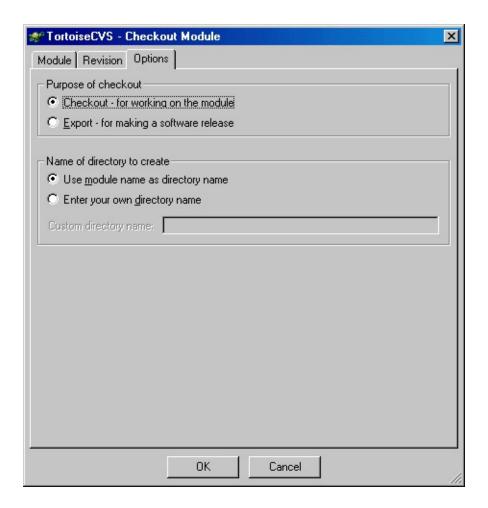


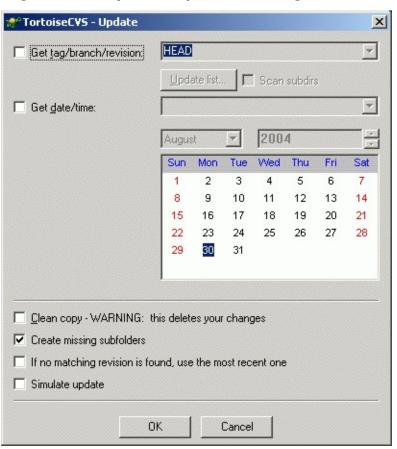
Figure 6.4. Checkout Dialog - Options Tab



Note

Update Special Dialog

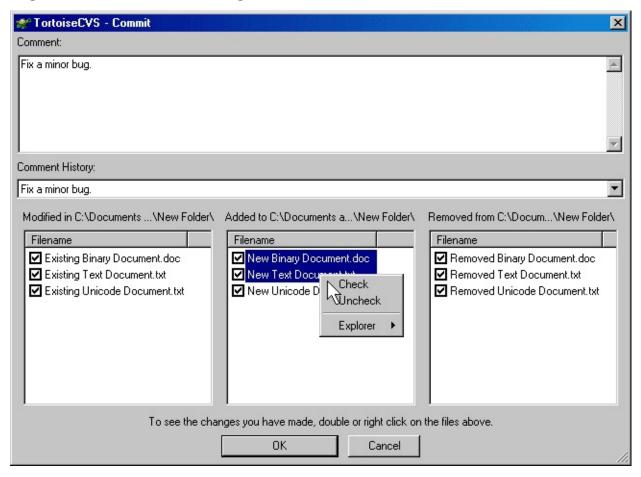
Figure 6.5. Update Special Dialog



Note

Commit Dialog

Figure 6.6. Commit Dialog



Note

Branch Dialog

Note

TODO: Write... Screenshot

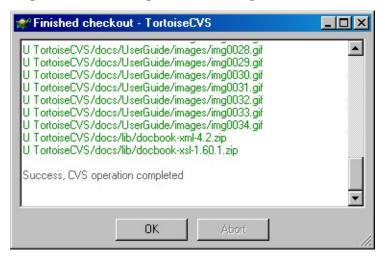
Make New Module Dialog

Note

TODO: Write... Screenshot

Progress Dialog

Figure 6.7. Progress Dialog



This dialog shows each CVS command that TortoiseCVS executes, as well as any information, errors, or warnings generated by CVS. The different types of messages are highlighted in different colours (you can customize these colours using the Appearance tab in the Preferences dialog).

During an Update, the Progress Dialog indicates whether or not each file has changed in CVS or on the local PC, using the following single-letter codes:

C

The file has changed both in CVS and locally. TortoiseCVS attempted to merge both sets of changes into the local copy, but could not (probably because both changes were made to the same part of the file). For more information, see Resolving Conflicts.

M

The file has only changed on the local PC. TortoiseCVS will not change the local copy.

P

The file has changed in CVS, and TortoiseCVS has updated the

local copy to match

U

The file exists in CVS, but not on the local machine. TortoiseCVS will create a copy on the local PC. You will also see this for files which have changed in CVS (as for *U*.

Α

The file has been added to CVS, but not yet committed.

R

The file has been removed from CVS, but not yet committed.

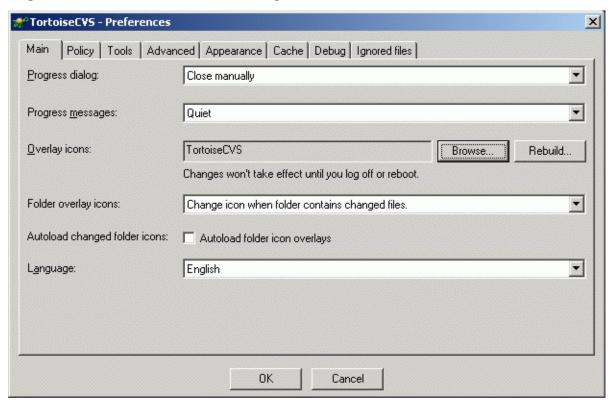
Tag Dialog

Figure 6.8. Tag Dialog



Preferences Dialog

Figure 6.9. Preferences Dialog - Main Tab



TODO: Describe notebook pages

Figure 6.10. Preferences Dialog - Policy Tab

| * | ^e TortoiseCVS - Preferences | X |
|---|--|---|
| | Main Policy Tools Advan | ced Appearance Cache Debug Ignored files |
| | Empty folders: | IV Prune empty folders |
| | Update creates missing folders: | ✓ Create new <u>f</u> olders on update |
| | Force head revision: | If no matching revision is found, use the most recent one |
| | Simulate update: | ☐ Just simulate an update, but don't modify any fil If you've specified a tag / branch / date |
| | Automatic unedit: | it matches no revision for a file, CVS will or revision instead. |
| | Automatic commit: | Automatically commit file after add (NOT recommended) |
| | Allow Network Drives: | Permit sandboxes and repositories on network (remote) drives (NOT recommended) |
| | Allow Removable Drives: | Permit sandboxes on removable drives (NOT recommended) |
| | Parse web logs: | ✓ Parse output of CVS web applications (ViewCVS, CVSweb,) |
| | Explorer Columns: | ✓ Show columns only for CVS-controlled folders |
| | | |
| | | OK Cancel |

Figure 6.11. Preferences Dialog - Tools Tab

| 🎢 TortoiseCVS - Preference | 25 | x |
|-----------------------------------|---|--------|
| Main Policy Tools Ad- | vanced Appearance Cache Debug Ignored files | |
| <u>D</u> iff application: | WinMerge.exe | Browse |
| <u>I</u> wo-way diff parameters: | "%1" "%2" | |
| Merge application: | WinMerge.exe | Browse |
| <u>I</u> wo-way merge parameters: | "%mine" "%yours" | |
| SSH <u>a</u> pplication: | TortoisePlink.exe | Browse |
| SSH parameters: | -l''%u'''%h'' | |
| | | |
| | | |
| | | |
| | OK Cancel | |

Figure 6.12. Preferences Dialog - Advanced Tab

| fain Policy Tools Advance | d Appearance Cache Debug Ignored files | |
|-----------------------------------|--|--------|
| Home folder: | Always recalculate home folder | |
| Custom home <u>f</u> older: | C:\Documents and Settings\tma.TMANOTE2 | Browse |
| etwork compression: | 0 - None | - |
| xcluded folders: | | |
| ncluded folders: | | |
| Menu icons: | Enable menu įcons | |
| Maximum number of tags for graph: | -1 | |
| andbox DOS/UNIX: | Autodetect (default to DOS) | |

Figure 6.13. Preferences Dialog - Appearance Tab

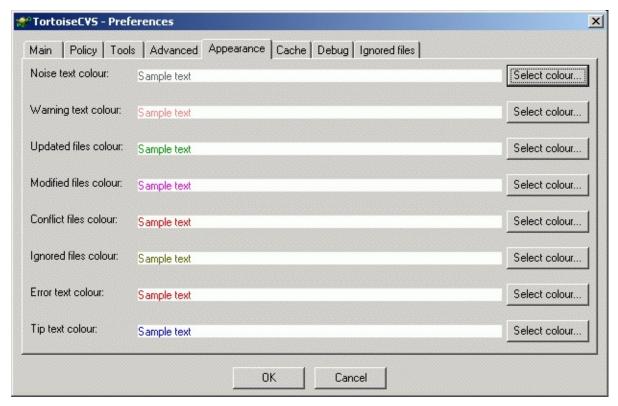


Figure 6.14. Preferences Dialog - Cache Tab

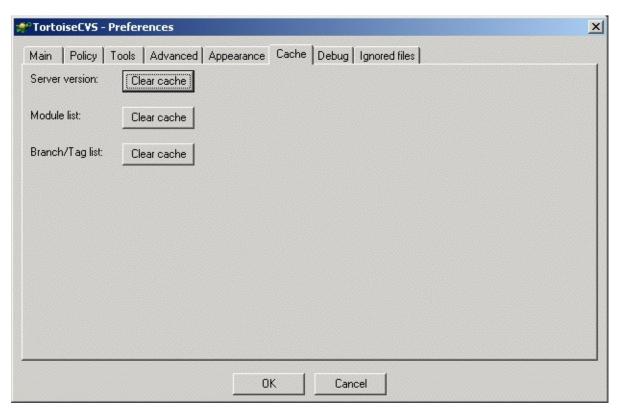
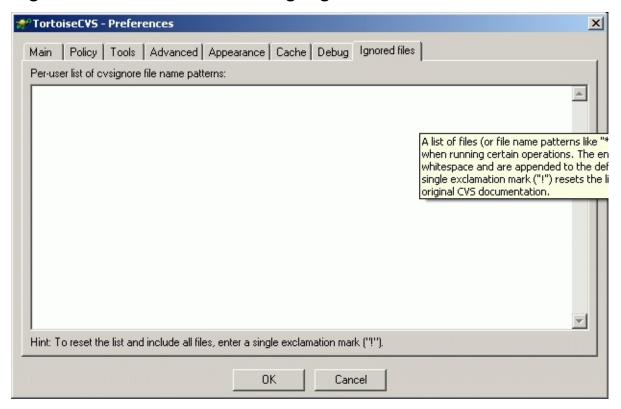


Figure 6.15. Preferences Dialog - Ignored Files Tab



TODO: Figure is cropped wrong

Figure 6.16. Preferences Dialog - Iconset Chooser



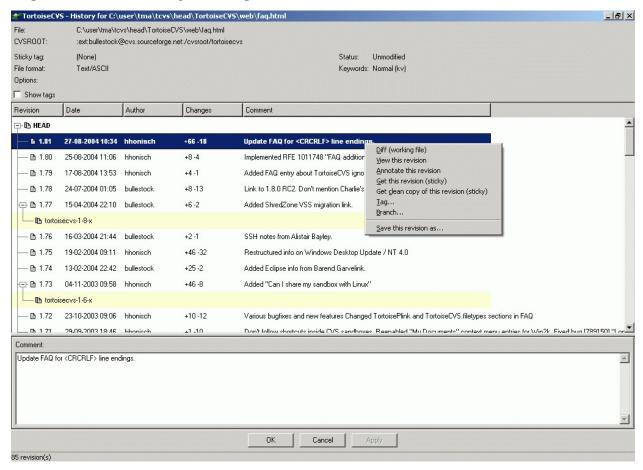
Merge Dialog

Note

TODO: Write... Screenshot

History Dialog

Figure 6.17. History Dialog

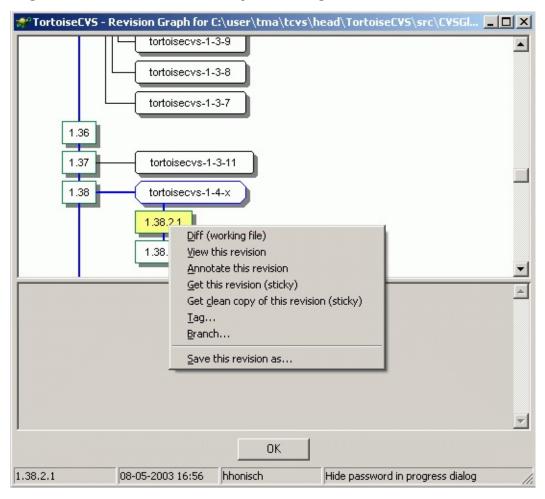


Note

TODO: Write

Revision Graph Dialog

Figure 6.18. Revision Graph Dialog



Note

TODO: Write

About Dialog

Figure 6.19. About Dialog



The About dialog shows the version of TortoiseCVS itself as well as the CVS client it uses, and the SSH application you have configured.

TODO: Describe and show dialog with and without CVS server version

Chapter 7. Articles, Tips and Tricks

Table of Contents

Using TortoiseCVS with SourceForge
How CVS Differs from Microsoft Visual Source Safe
Creating new icons for TortoiseCVS
Creating a new locale (translation) for TortoiseCVS
Using PuTTY sessions
Using Pageant to avoid having to enter the SSH password every time

Using TortoiseCVS with SourceForge

TortoiseCVS Client Installation Instructions

This document provides instructions for installing and using the TortoiseCVS CVS client software. These instructions are designed for SourceForge.net project developers who are using 32-bit Microsoft Windows platform. It is for developers who would like to work with the source code for a project which is hosted on SourceForge.net. For background information you may like to read the other available documents about CVS.

TortoiseCVS has advantages over WinCVS for work on SourceForge. It has a clearer interface which is integrated into Windows Explorer. WinCVS has a more complex and mature interface. For more obscure commands it may be easier or necessary to use WinCVS.

A particular benefit is that TortoiseCVS has an SSH client built into it, without special setup. Anyone can easily commit code to SourceForge.net projects using TortoiseCVS.

Installing TortoiseCVS

- 1. Go to http://tortoisecvs.org/ and download the Download page.
- 2. To install, run the executable and follow the instructions.
- 3. Since TortoiseCVS is a shell extension, you need to reboot your machine.

Checking out an existing project

If you are working on an existing project first you now need to check out the code.

- 1. Make sure you have made an account on SourceForge.net, and that you have been added as a developer to the project you are going to work on. A project administrator needs to do this.
- 2. In Windows Explorer go to the folder that you would like to check the

code out into.

- 3. Right click and from the context menu choose the command **CVS Checkout...**; it's next to the Tortoise icon.
- 4. You will need your user name on SourceForge.net, and the project UNIX name. Fill in the dialog as follows:

Protocol: Secure Shell (:ext)

Server: cvs.sourceforge.net

Directory: /cvsroot/projectname

• Username: username

- 5. Click **Fetch List** to get a choice of modules. You will need to enter your password in the dialog at this point. Then pick the module that you would like.
- 6. Choose **OK**.TortoiseCVS will now check out the code. You will need to enter your password.
- 7. You can now alter files and then select **CVS Commit** on the top level folder to make your changes in the repository. For more information read the TortoiseCVS User's Guide.

Get an error?

If you get an error saying something like could not chdir to home folder /home/users/r/rd/rdonkin: No such file or folder then you need to log into the server with SSH to force creation of your home folder. Do this by connecting to username@cvs.sf.net either with Putty or a command line SSH.

Working on a new project

If you are working on a new project then you first need to create a new module in the repository.

- 1. Create a folder with the name you would like the module to have.
- 2. Put the files you would like to add in the folder.

- 3. Right click on the folder, and choose **Make New Module** from the CVS submenu.
- 4. Fill in the dialog as described in the previous section. Note that the module name is filled in for you.
- 5. Choose **OK**. You will need to enter your password twice, and the module will be made for you.
- 6. Proceed to add files and folders by right clicking on them and choosing CVS Add or CVS Add All Recursively from the context menu.
- 7. For more information read the TortoiseCVS Daily User's Guide

Tired of entering your password all the time?

You need to make a public/private key pair. The public key goes on the server, and the private key sits on your hard drive. Now, when you connect the server can verify who you are because only you have your private key.

To further protect your private key it is encrypted on your hard disk with a passphrase. You can however choose an empty passphrase, in which case you never have to enter a password but anyone with access to your computer can access your SSH CVS account. Or you can use an SSH Agent (such as Pageant) which will remember your passphrase for the duration of one session, so you only have to enter it once.

TortoiseCVS uses a version of Plink as its SSH client. This means you can use PuTTYgen to make a public/private key pair, and you can use Pageant to point to where the private key is. First of all download them both from the PuTTY web site.

For information on how to use them read the PuTTY documentation on PuTTYgen and Pageant. In particular, make sure you read the sections "8.2: Using PuTTYgen, the PuTTY key generator" and "9.3.1 Making Pageant automatically load keys on startup".

After making your public key with PuTTYgen you need to upload it using the SourceForge.net web interface. Log into the SourceForge web site, and go to your Account Preferences page. Down at the bottom is a section "Shell Account Information" with an entry "CVS/SSH Shared Authorized Keys". You need to add your public key there.

How CVS Differs from Microsoft Visual Source Safe™

TortoiseCVS differs from Visual Source Safe $^{\text{TM}}$ (VSS) in many ways. The most apparent difference may be that TortoiseCVS does not require users to lock the files they are working on, as VSS does by default. In fact, the CVS documentation even encourages users not to use file locking. In the rare occasion where several people have changed the same file at the same time, CVS will normally be able to merge their changes. If two or more developers have changed the same few lines, TortoiseCVS will report a conflict, insert directives in the file, and leave it to the developers to decide what to do. Such conflicts are very rare, as they normally occur as a result of lacking communication between the developers (eg. two people trying to fix the same problem).

Another important difference is that VSS gives you a server view, while TortoiseCVS shows a client view. In practice this means that, unlike VSS, TortoiseCVS will not tell you about changes in the repository until you do an update, or explicitly query the status of selected files. Changes reported by TortoiseCVS reflect modifications done by you after the last checkout, update or commit.

Creating new icons for TortoiseCVS

TortoiseCVS icons are Windows icon files that are overlaid on top of source code file icons in Explorer to show their status in CVS. The reason this technique works is that most of the area in a TortoiseCVS icon is transparent, allowing the source icon to show through. If you are designing your own TortoiseCVS icon set, you'll need to strike a careful balance between visibility of the original file and easy viewing of CVS states. Some icon sets achieve this balance by adding a distinctive colour or shape to one corner of each icon; others achieve the illusion of shading the entire icon by alternating transparent pixels with coloured ones. You can even combine these two ideas of unique shapes and semitransparent shading: see the "NG" icon set that ships with TortoiseCVS for an example.

- 1. Create your icon set using a graphics editor. For the best viewing results, you should create 16x16, 32x32, and 48x48 versions of each icon. The icon editors that ship with the Borland and Microsoft compilers can do this task easily, as can most shareware icon editors. If you prefer to stay with free software, you can use The GIMP to create a PNG file for each icon size, and then combine them together with png2ico.
- Make sure you have created one correctly-named icon for each of the seven possible CVS states: TortoiseNotIncvs.ico, TortoiseIncvs.ico, TortoiseIncvsReadOnly.ico, TortoiseChanged.ico, TortoiseConflict.ico, TortoiseAdded.ico, and TortoiseIgnored.ico.
- 3. Create a folder inside the directory where you installed TortoiseCVS, name it something that matches the theme of your icon set, and copy your seven new icon files into the new folder.
- 4. Open the Registry Editor (Click **Start** → **Run** and type regedit). Navigate to hkey_local_machine\software\tortoiseCvs\customicons. Inside this key, create a new string value. For the name of this new value, type in the name that you would like to call your new icon set. Double-click on the new value to set the actual data, which is the

name of the folder you created in step 3.

Creating a new locale (translation) for TortoiseCVS

TODO

Using PuTTY sessions

If you need different SSH parameters for different hosts, you can do so by using a feature of the PuTTY suite called *sessions*.

- 1. Download the PuTTY suite from the PuTTY site.
- 2. Run putty.exe and enter the required parameters, then enter an appropriate session name below the **Saved Sessions** label, and click **Save**.
- 3. In the TortoiseCVS Checkout Dialog, enter the session name instead of the host name.

Using Pageant to avoid having to enter the SSH password every time

When using SSH, you normally have to enter your password every time you do a CVS operation. To avoid this, while maintaining a reasonable level of security, you can use an *SSH Agent*. The SSH Agent in the PuTTY suite is known as Pageant.

- 1. Download the PuTTY suite from the PuTTY site.
- 3. To avoid having to do the above every time you boot up your computer, create a shortcut to Pageant in your **Programs** → **Startup** folder, specifying the key file as argument.
- 4. TODO

Chapter 8. Resources

The book *Open Source Development with CVS* is available in electronic form from the book's web site.

The reference manual for CVSNT™ is available here.