

Users Guide

visualCVS

Configuration Management Workbench



Table of Contents

Table of Contents	2
Introduction	4
System Overview	5
References	6
Finding Information about CVS	6
visualCVS Technical Support	6
Workspace Browser	7
Item Status	7
Item Status	8
Adding Items to the repository	8
Configuring add/import file types	9
Commit Items to the Repository	9
Update	10
Removing Items form the Repository	10
Status Function	10
Item Log Report	10
Diff Items	10
Show Baseline Tags	11
Adding Baseline Tag	11
Creating Baseline Views	11
Creating Branches	11
Creating Annotation Reports	12
Expert Options	13
Filtering	13
Hiding items	14
Maintaining Filter Lists	15
Navigating within the Workbench Browser	16
Collapse/Expand	16
Search	16
Sort	16
Selecting System Drives (Windows only)	16
Resizing Columns	16
Resizing a View	16
Keyboard bindings	17
Default Bindings List View	17
User Specific Bindings (default setting)	17
Default Bindings Tree View	18
Hidden Files	18
CVS Command Result, Show Last CVS Message	18
Archive Browser	19
Selecting the Repository	20
Browser Mode	20
Updating mode	20
Non-updating mode	20
Import	20
Import	21
Checkout	21
Creating Baseline Views	22
Log Report	22
Filtering within Archive Browser	22
Keyboard Support	22
Default Bindings Repository List View	22
Default Bindings Tree View	22
Baseline Browser	23
Creating a Baseline View	24
Drawing Baselines	24
Navigate Along Baselines	24
View Item Pop Up	24

View Item Log and Revision Log	25
View Baseline Tags	25
Viewing Differences of Revisions	25
Zooming	26
Keyboard Bindings.....	27
VisualCVS Finder	28
Searching through files	28
Finding files.....	29
Regular Expressions.....	30
visualDiff Tool.....	31
Integrated Text Editor	31
Configuration Options.....	32
Configuration Files	32
File Format	32
visualCVS Client	32
visualCVS Server	32
Editor Integration	33
Adjusting Keyboard Bindings	33
Configuration Parameter	34
visualCVS Server.....	34
Global Application Definitions	34
visualCVS Client.....	35
Global Application Definitions	35

Introduction

visualCVS is a configuration management environment based on the open source tool CVS (Concurrent Version System).

The visualCVS Components:

- Workbench Browser for daily use or as integration environment (IDE)
- Archive Browser for repository management
- Baseline Browser for graphical representation of the repository
- Workflow support
- Powerful search tool
- Extensive search and filter capabilities
- Multiple platform support (Windows, Linux, Solaris, ...)

CVS has become one of the most spread systems for configuration management. Originally started as a collection of unix shell scripts, CVS grew to a reliable system. Due to the practical approach during the development, CVS offers all you need to maintain data throughout its life cycle.

Compared to other systems CVS keeps a rather loose control over its data. Following an example session of a traditional system:

- A single item is checked out and locked
- Changes are applied to the item
- The item is committed back into the repository as a new revision
- The system is built including the change

And the same session with CVS:

- A working copy of a system is checked out
- Changes are applied to some of the items
- The system is built and tested locally
- The changed items are committed back to the repository

Due to the distributed approach of CVS, several advantages arise:

- Any development tools can easily be integrated without restrictions
 - Integration within traditional CM environments are complex, costly and often reduce functionality
 - Traditional systems can add a lot of CM overhead work, the team has to cope with
- Concurrent development
 - Each user has his own consistent set of items
 - The CVS approach works within large development teams and distributed environments
- Parallel development is painless
 - The automatic merging of CVS works reliably
- The repository can run even on small sized servers
 - no bottle neck, excellent scalability
- Successful introduction and usage of CM depends on the acceptance of the solution
 - The content and quality of a repository is an interesting measure whether the team lives the CM process or not

System Overview

VisualCVS can be installed locally or within a networked environment. Due to the cross platform support of visualCVS and CVS, client workstations with different operating systems can share one or several central CVS repositories.

The access to the repository can be through a local file system (local access), through LAN access (rsh or pserver) or through external programs as e.g. ssh (secure shell).

Note: Using a repository on a networked drive is not recommended. Multiple access can cause data loss due to the timing behaviour of the network drive (CVS file locking mechanism).

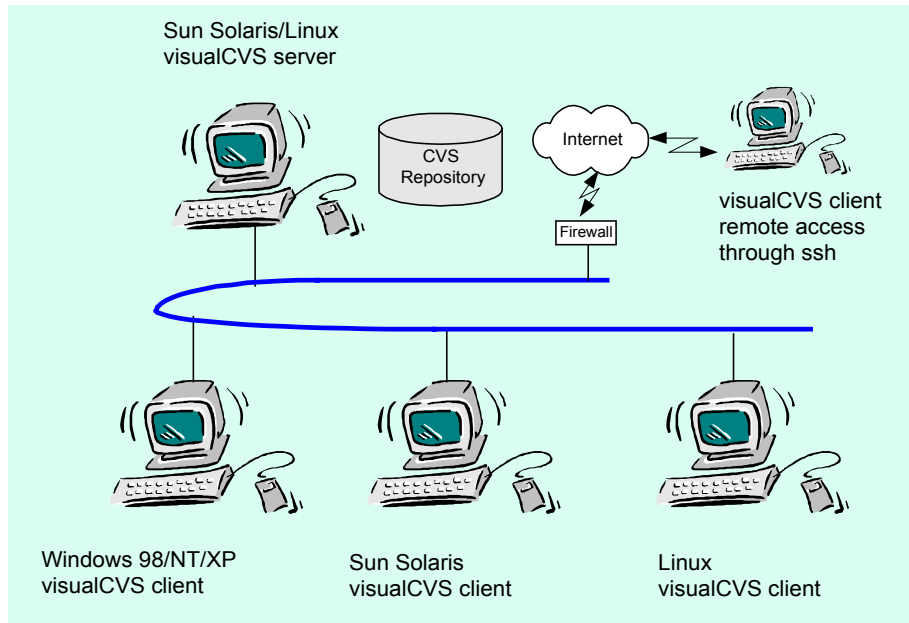


Figure: Example of a distributed visualCVS installation

References

- [1] CVS Reference Manual
Version Management with CVS by Per Cederqvist available at:
www.scentech.ch/products/visualcvs/documentation
- [2] The home of CVS
www.cvshome.org
- [3] The home of CVS for NT
www.cvsnt.org

Finding Information about CVS

The CVS home WEB Site provides documentation and links related to CVS to help you find the information that you are seeking. See the CVS home page at:

- www.cvshome.org
- www.cvsnt.org
- www.cvshome.org/cvslinks.html

The CVS related news groups offer a highly dynamic discussion forum to other CVS users at:

- comp.cvs.help
- comp.software.config-mgmt
- comp.cvs.bug

There are a number of related books covering Configuration Management with CVS at:

- www.cvshome.org/docs/books.html

visualCVS Technical Support

Scentech offers several levels of Technical Support. For information for visualCVS Technical Support, please email to:

- visualcvs@scentech.ch

Workspace Browser

The Workspace Browser offers functions for daily CM use as:

- Status view of a collection of CVS items
- Apply CVS commands (commit, update, add, remove, status, log, ...)
- Basic file functions (create, delete, rename, touch)
- Filtering for filename, item state, date and file type
- Integrated editor
- Integrated visualDiff-tool

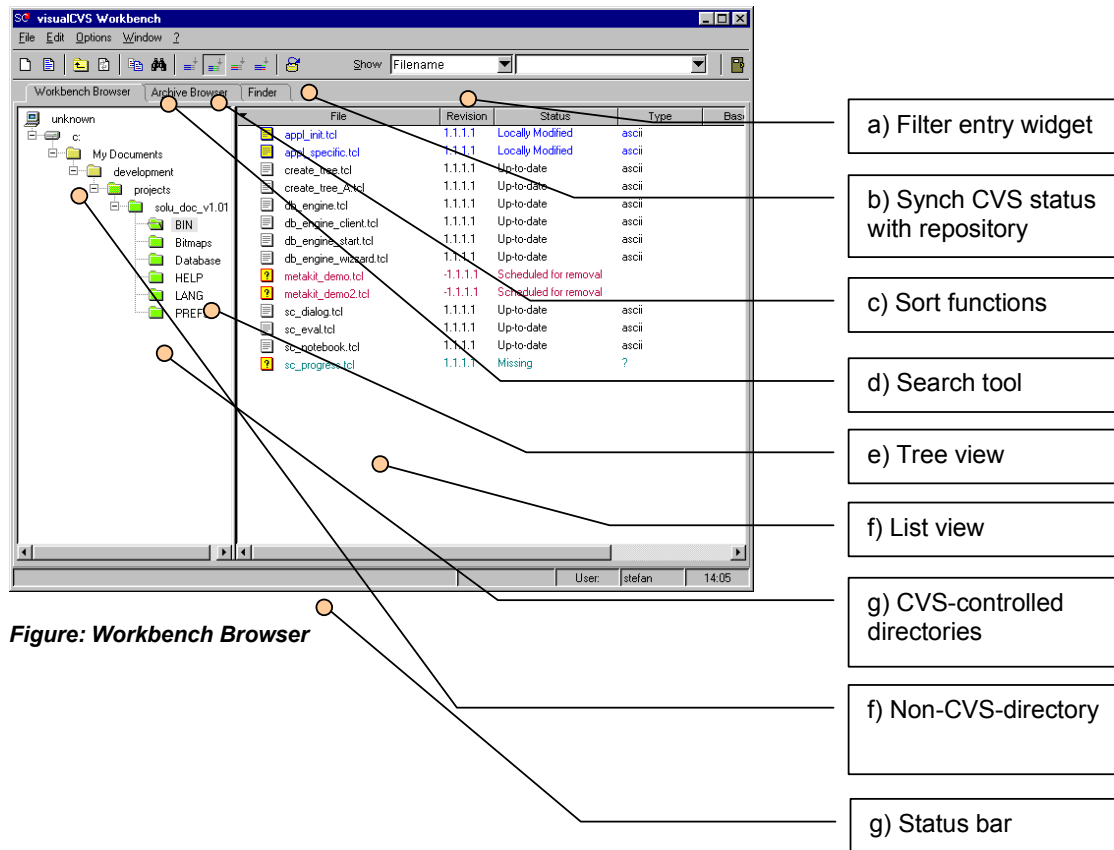












Figure: Workbench Browser

Item Status

The Workbench Browser recognises CVS and non CVS items. Depending on the item state, functions of the context menu are activated or disabled. visualCVS represents different states with the following icons:

	Directory unknown to visualCVS
	Item unknown to visualCVS
	Directory controlled by visualCVS
	Directory unknown to visualCVS (needs add)
	Item unknown to visualCVS (needs add)
	Item controlled by visualCVS (Up-to-date)
	Locally modified item (needs commit)
	Locally added item (needs commit)
	Stale time tag of item (needs update or touch)
	Repository item (Archive Browser)

Adding Items to the repository

Select the item and invoke the *add* function through the context menu (**right mouse button**). In case the file extension is unknown to visualCVS, the add/import dialogue window appears. Within the dialogue, the file type must be specified:

- add item as text
- add item as binary (default setting)
- ignore, do not add/import

In order to change the type, select the entry and press the right mouse button to open the context menu. The type definition will be stored in the session in case the **Save in session** flag is active. The types can be pre-configured see Configuring add/import file types.

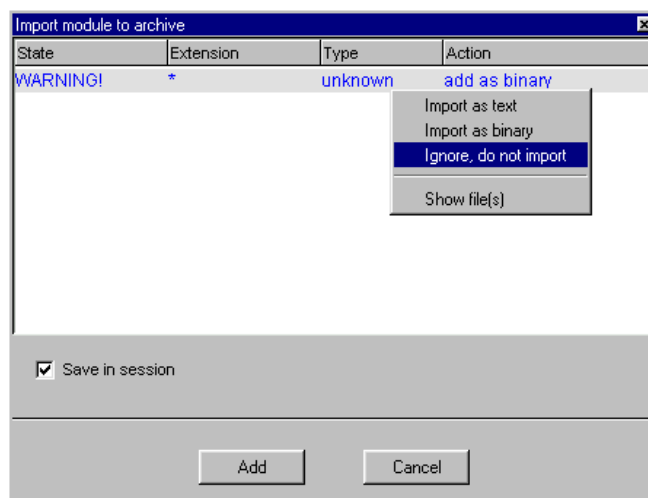


Figure: Add/import dialogue

If the item has been successfully added, the status changes from *Unknown* to *Locally Added*. Use commit to add the item to the repository.

Configuring add/import file types

Invoke the function **Options->Configuration->Edit CVS add/import file types** from within the main menu bar to open the edit wizard. Then double-click on existing entries to change the type or add new entries. The types can also be pre-configured see Add/Import Format.

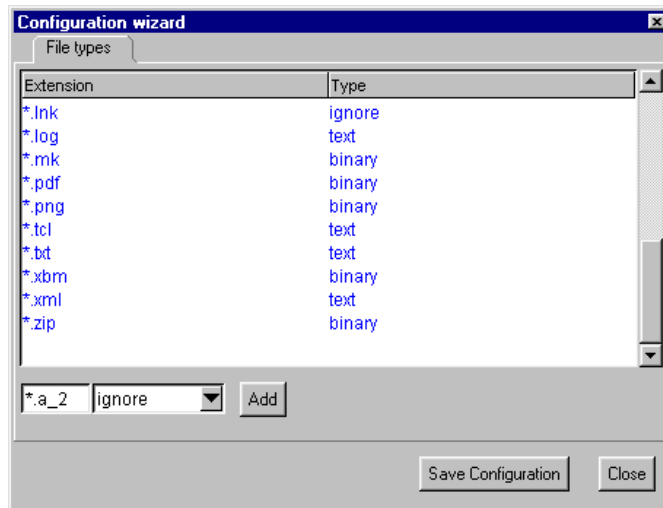
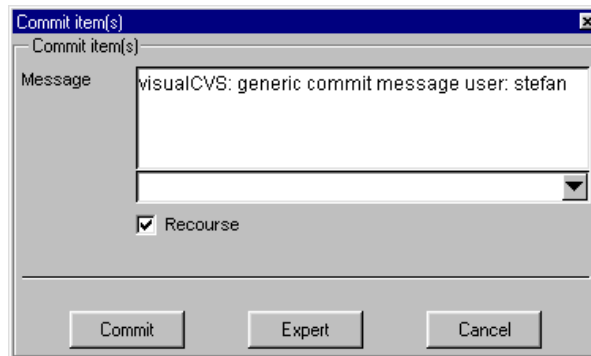


Figure: Edit CVS add/import file types dialogue

Commit Items to the Repository

Select the item(s) to commit and invoke the **commit** function through the context menu (**right mouse button**). Enter a message text describing the changes to the item(s) or select a message text from a previous commit. Then press the commit button. For a description of the Expert options see



Expert Options.

Figure: Commit dialogue

Update

Select the item(s) to update and invoke the *update* function through the context menu (***right mouse button***). Enter a baseline tag or a revision id and then press the update button. For a description of the Expert options see

Expert Options.

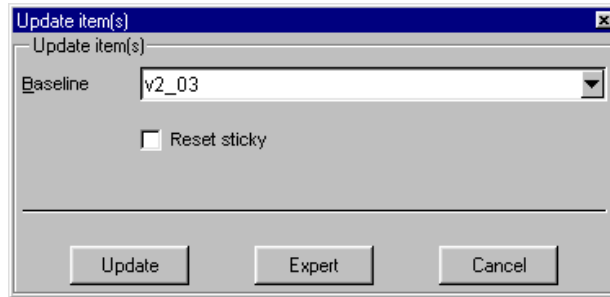


Figure: Update dialogue

Removing Items form the Repository

Select the item(s) to be removed and invoke the *remove* function through the context menu (**right mouse button**). The item state changes to *scheduled for removal*. Use commit to remove the scheduled item permanently.

Status Function

Select the item(s) to retrieve a status report and invoke the *status* function through the context menu (**right mouse button**).

Item Log Report

Select the item(s) to retrieve a CVS log report and invoke the *log report* function through the context menu (**right mouse button**).

Diff Items

The integrated diff function compares the local item with a repository version of the item and highlights the differences. visualCVS diff offers several options:

- Diff the local item against the archive item
- Diff the local item against a specific revision of the archive item
- Diff item with the most recent revision (head revision within repository)
- Diff item with a specific revision

In case the *diff repository items with specific revision* function is called, the dialogue bellow offers selection lists by revision id or by baseline tag. The content of the selection boxes is queried from the repository when the function is called.

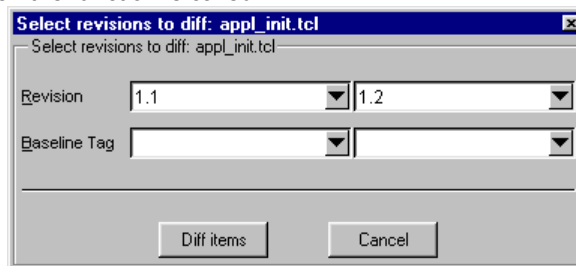


Figure: Diff repository items with specific revision

Show Baseline Tags

The function retrieves a list of all baseline tags of the selected item.

Adding Baseline Tag

Apart of the common CVS tags, visualCVS offers an extended baseline tagging function. The function allows:

- Adding user specific fields to a baseline tag e.g. comment field, ...
- Setting a creation time tag
- Adding the creator's name
- Defining min/max length of the user entry

Depending on the tag definition, the dialogue box changes the number of entry fields. For information on how to customise see Enhanced Baseline Tags. The default baseline tag window:

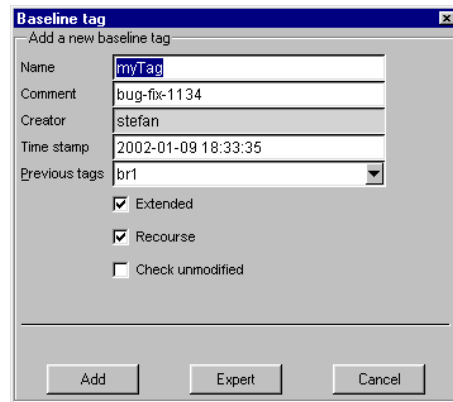


Figure: Baseline tag add dialogue

Creating Baseline Views

For a detailed description on baseline views please refer to the section Baseline Browser of this manual.

Creating Branches

Select the item(s) to be branched and invoke the *create branch* function through the context menu (**right mouse button**). For more information on branching see [1]. Note that the branch tag function supports extended tags.

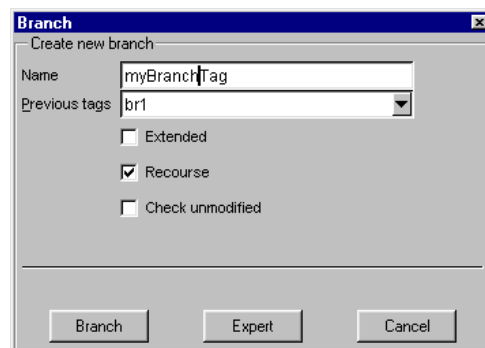


Figure: Branch tag dialogue

Creating Annotation Reports

The *annotate* command offers a historical view to one or several items including information on what has changed by whom and when.

The annotate report can be generated over one or several items or even a whole project. The output can be filtered with a date or a baseline tag. In the example bellow all items are returned with the status on January 30.

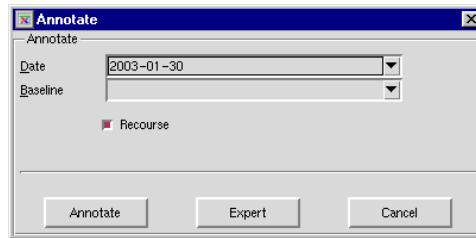


Figure: Annotate dialogue

The result of the *annotate command* is presented in a report window. The report window offers functions to:

- show all changes of a specific user
- show all changes which were done on a specific date
- show all changes which were done on a specific revision index
- search function

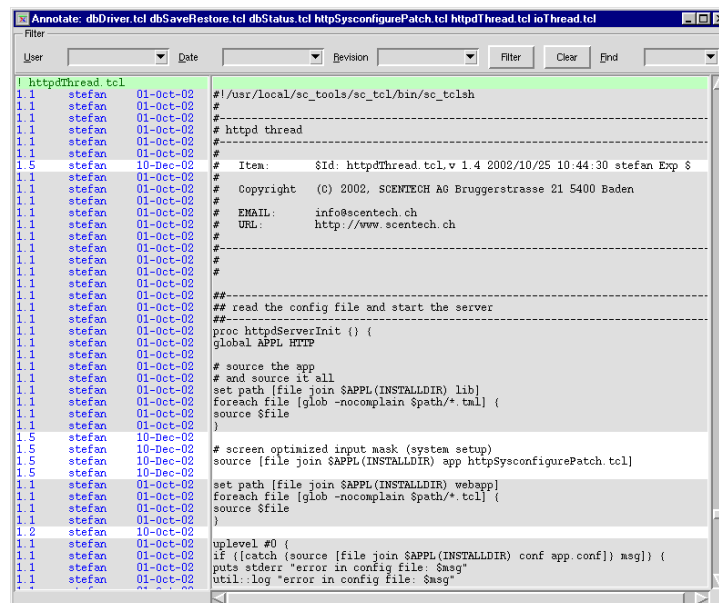


Figure: Annotate report

Expert Options

In some of the dialogue windows an additional entry line will be shown by pressing the **Expert** button. Within this entry line, additional CVS command options can be entered. A CVS command is compiled as:

```
CVS [APPL(CVSFLAGS,GLOBAL)] command [APPL(CVSFLAGS,command)]
    [Expert command options] item(s)
```

For details on CVS command options please refer to [1] and CVS Command Line Flags.

Filtering

Items within the list view can be filtered. Filters are available for:

- Filename
- State
- Type
- Date

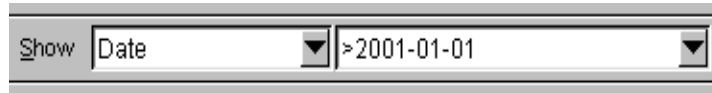


Figure: Show filter

The filter entries can be combined with + as well as with a globe style pattern matching.

- ? match to any single character
- * match to any sequence of characters
- + combine filters
- > bigger than
- < lower than

The following syntax applies:

Filter Type	Filter Entry Example	Description	Supported Operators
Filename	a* + *.h	Show all items starting with a of with extension .h	<>+?
State	Up-To-Date+Unknown	Show all items with status up-to-date or unknown	<>+?
Type	?	Show all items with unknown type	<>+?
Date	2001-01-01 12:39:55	Filter item time tag	<>+ no wildcard support time can be omitted seconds can be omitted

Table: Filter Syntax

Hiding items

The Workbench Browser has an integrated filter to hide items. Place the mouse pointer over the show filter label text (see Figure: Show filter) and click on the **right mouse button** to toggle to the hide filter widget. Following a filter definition which hides all files with the file extensions `.c`, `.swp` and `.bkp`. For a description of the syntax see Table: Filter Syntax.



Figure: Hide filter

Maintaining Filter Lists

Place the mouse pointer within a filter list then press the **right mouse button** to open the context menu. The maintenance context menu offers:

- Changing the order of the filter entries
- Sorting the entries ascending/descending
- Removing filter entries

The function is integrated in all editable filter list within the visualCVS application. The list length of a filter list is truncated when visualCVS is started. To adjust the maximum length see APPL(SESSION,MAXLENGTH) Global Application Definitions.

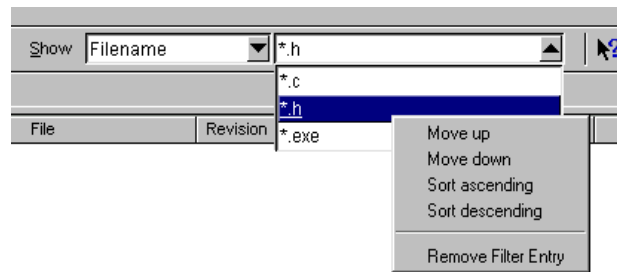


Figure: List maintenance context menu

Navigating within the Workbench Browser

Collapse/Expand

Within the tree view double-click on icons or labels to collapse or expand the tree items. The function can also be invoked with the left/right arrow keys.

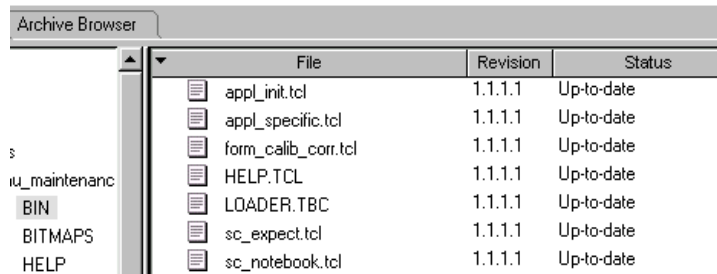
Search

Place the mouse pointer within a tree- or list-view. Then press the first character of the entry to be found. The selection moves to the next matching entry starting from its current location. If multiple entries match, the next entry can be found by pressing the character again.

Sort

All Columns within the list view can be sorted ascending and descending. Click on the column label to sort the entries:

- First click to sort ascending
- Second Click to sort descending
- Third Click to deactivate sort and show the entries as read from the disk



File	Revision	Status
appl_init.tcl	1.1.1.1	Up-to-date
appl_specific.tcl	1.1.1.1	Up-to-date
form_calib_corr.tcl	1.1.1.1	Up-to-date
HELP.TCL	1.1.1.1	Up-to-date
LOADER.TBC	1.1.1.1	Up-to-date
sc_expect.tcl	1.1.1.1	Up-to-date
sc_notebook.tcl	1.1.1.1	Up-to-date

Figure: List-view, file-column sorted ascending

Selecting System Drives (Windows only)

To retrieve a list of system drives, double-click on the system name. Then select the desired drive.

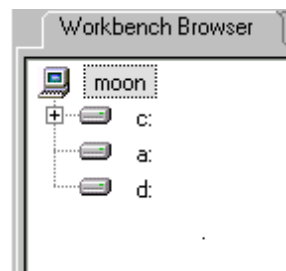


Figure : System drives

Resizing Columns

Place the mouse pointer to the title line of a column and move it to the column border until the cursor changes to a resize cursor. Click the **right mouse button** and drag the column to its new size.

Resizing a View

Place the mouse pointer over the separator line between a tree view and a list view until the cursor changes to a resize cursor. Click the **right mouse button** and drag the view to its new size.

Keyboard bindings

Default Bindings List View

Place the mouse pointer into the list view to set the keyboard focus (see f: Figure: Workbench Browser). Within the list view the following keys are supported:

Key	Function	Comment
Left Mouse Button	Select item	Select the item beyond the mouse pointer, reset previous selection
Ctrl-Shift-Left Mouse Button	Set selection range	Select all items from the last selected until the item beyond the current mouse pointer position
Ctrl-Left Mouse Button	Include item in selection	Select the item beyond the mouse pointer, keep the selection
Left Mouse Button	Edit selected items	
Return	Edit selected items	If keyboard focus is outside of the selected items only the item with the focus will be opened
Up	Move focus up	
Down	Move focus down	
Space	Select focus	Select single item, toggle
Ctrl-Space	Select focus	Include focused item in the selection, toggle
PageUp	Move focus by 1 page	
PageDown	Move focus by 1 page	
Home	Set focus to topmost item	
End	Set focus to last item	
Ctrl-a	Select all	
Ctrl-q	Toggle selection	Toggle selection for all items within the list view

User Specific Bindings (default setting)

Apart of the default bindings, the following actions can be mapped to a user specific setting (see also).

Key	Function	Comment
Return	Edit selected items	
F2	Rename/move file	
F3	Create new file	
F4	Edit new file	
F5	Refresh view	
F6	Rename item(s)	
F7	Create directory	
F8, Del	Delete item(s)	
F9	Touch time tag	
F10	Go up	

Default Bindings Tree View

Place the mouse pointer into the tree view to set the keyboard focus (see e: Figure: Workbench Browser). Within the tree view the following keys are supported:

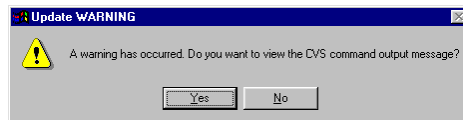
Key	Function	Comment
Left Mouse Button	Select item	Select the item beyond the mouse pointer, reset previous selection
Up	Move focus up	Change to new directory and refresh list view
Down	Move focus down	Change to new directory and refresh list view
Left	Collapse tree	
Right	Expand tree	
PageUp	Move selection by 1 page	
PageDown	Move selection by 1 page	
Home	Select topmost item	
End	Set focus to last item	

Hidden Files

Activate the menu select button **Options->Configuration->Show hidden files** from within the main menu bar to include hidden files in the Workbench Browser file list.

CVS Command Result, Show Last CVS Message

VisualCVS makes use of the standard CVS program. After a command execution, the CVS return message is analysed for warnings and errors. Depending on the command result and the selected warning level, a dialog box appears. Select yes if you want to review the command output. Invoke the function "Show last CVS message" **Options->Configuration->Show last CVS message** from within the main menu bar to review the result of the last command call.



Adjust the warning level by selecting one of the options (low, medium, high) **Options->Configuration->Warning level** from within the main menu bar.

Warning levels are:

- Low report all abnormalities
- Medium report warnings/errors
- High report only severe warnings/errors

Archive Browser

The Archive Browser offers a view to the repository content:

- Browse the repository content
- Import modules
- Checkout modules
- Log report
- Creating baseline views of repository items
- Filtering for filename, item state, date and file type
- Connection to multiple repositories

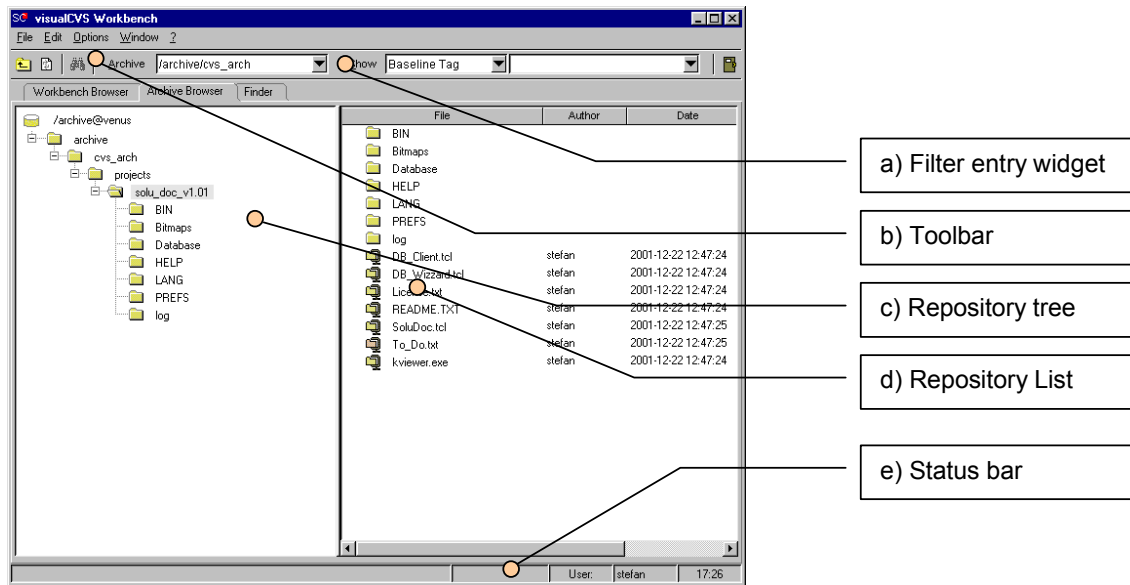


Figure: Archive browser

Selecting the Repository

Select the repository through the archive widget. After selecting the repository, visualCVS tries to connect to the selected visualCVS server.

In case no connection can be achieved or the connection drops, the *connection to archive lost* icon is drawn. Double-click on the *connection to archive lost* to re-establish the connection.

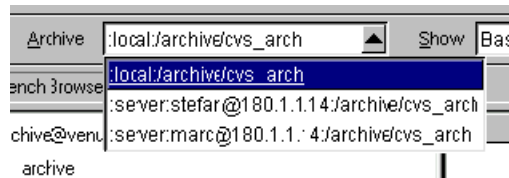


Figure: Repository selection

Browser Mode

The repository browser can be used in a updating and non-updating mode (see Configuration Parameter Archive Browser).

Updating mode

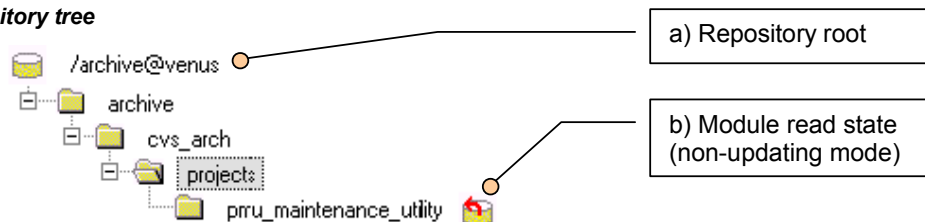
In updating mode the browser requests an update from the visualCVS server after each change of a directory.

Non-updating mode

The non-updating mode is designed for the use through slow access lines (e.g. modem connection to the repository). The visualCVS server sends as less information as possible in the non-updating mode. If a module did not receive it's child items, the module read state icon is drawn (b). To receive the child items from the server, select the module with the mouse pointer or the keyboard. To refresh the repository tree double-click on the repository root (a) or use the toolbar button.

For more information on navigation within the repository please refer to the section Navigating within the Workbench Browser of this manual.

Figure: Repository tree



Import

Select the project- module-root directory within the repository tree. Invoke the *import* function through the context menu (**right mouse button**). Select the source directory within the import dialogue and enter the fields for vendor-, branch-tag and import comment message.

After pressing the *import* button, the source tree is scanned for unknown file types. In case unknown types are found, the import type dialogue appears. Adjust the types (import as ascii, import as binary, ignore) and press the *import* button of the dialogue. For more information on the import file types see the section Figure: Add/import dialogue of this manual.

After successful import, the archive browser needs to be refreshed through the function *Refresh view*.

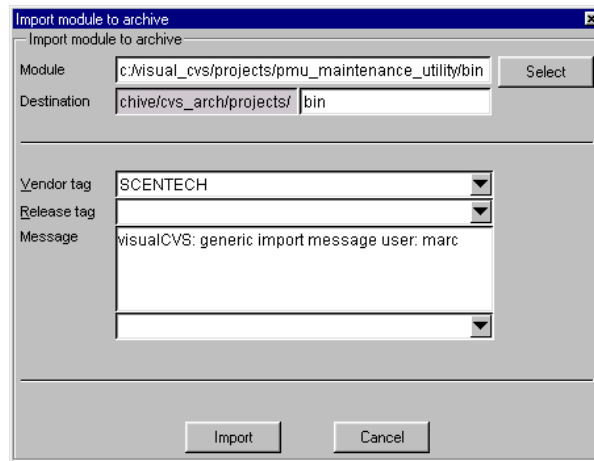


Figure: Import module dialogue

Checkout

Select the module or item to check out and invoke the *checkout module* function through the context menu (**right mouse button**). Select the destination directory within the import dialogue and enter a baseline tag or a specific item revision. If the baseline entry field is omitted, the most recent revision is checked out.

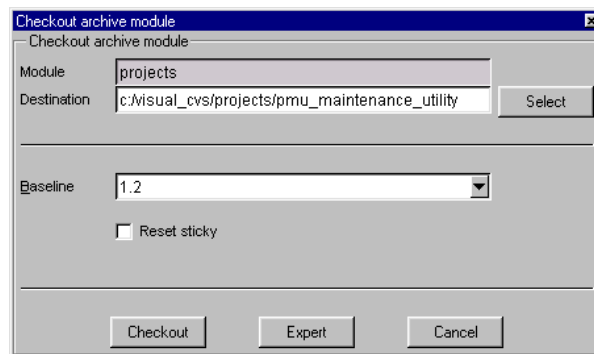


Figure: Checkout module/item dialogue

Creating Baseline Views

Baseline views can be created without checking out the repository items. For a detailed description on baseline views please refer to the section Baseline Browser of this manual.

Log Report

Log reports can be created without checking out the repository items. Select the desired items or directories with the mouse pointer and invoke the function *Log report* from within the context menu.

Filtering within Archive Browser

For more information on usage of the filters please refer to the section Filtering of this manual.

Keyboard Support

Default Bindings Repository List View

Place the mouse pointer into the repository list view to set the keyboard focus (see d: Archive Browser). Within the list view the following keys are supported:

Key	Function	Comment
Left Mouse Button	Select item	Select the item beyond the mouse pointer, reset previous selection
Ctrl-Shift-Left Mouse Button	Set selection range	Select all items from the last selected until the item beyond the current mouse pointer position
Ctrl-Left Mouse Button	Include item in selection	Select the item beyond the mouse pointer, keep the selection
Up	Move focus up	
Down	Move focus down	
Space	Select focus	Include focused item in the selection, toggle
PageUp	Move focus by 1 page	
PageDown	Move focus by 1 page	
Home	Set focus to topmost item	

Default Bindings Tree View

Place the mouse pointer into the repository tree view to set the keyboard focus (see c: Archive Browser). Within the repository tree view the following keys are supported:

Key	Function	Comment
Left Mouse Button	Select item	Select the item beyond the mouse pointer, reset previous selection
Up	Move focus up	Change to new directory and refresh list view
Down	Move focus down	Change to new directory and refresh list view
Left	Collapse tree	
Right	Expand tree	
PageUp	Move selection by 1 page	
PageDown	Move selection by 1 page	
Home	Select topmost item	
End	Set focus to last item	

Baseline Browser

Baseline views can be created within the Workbench Browser as well as within the Archive Browser. The Baseline Browser offers:

- Graphical view of the revision tree of items
- Draw baselines between items
- View log information on each item/revision
- View baseline tags on each item/revision
- Filter function for
 - Filename
 - Revisions
 - Date
 - Status
- Navigation along baselines
- Zooming

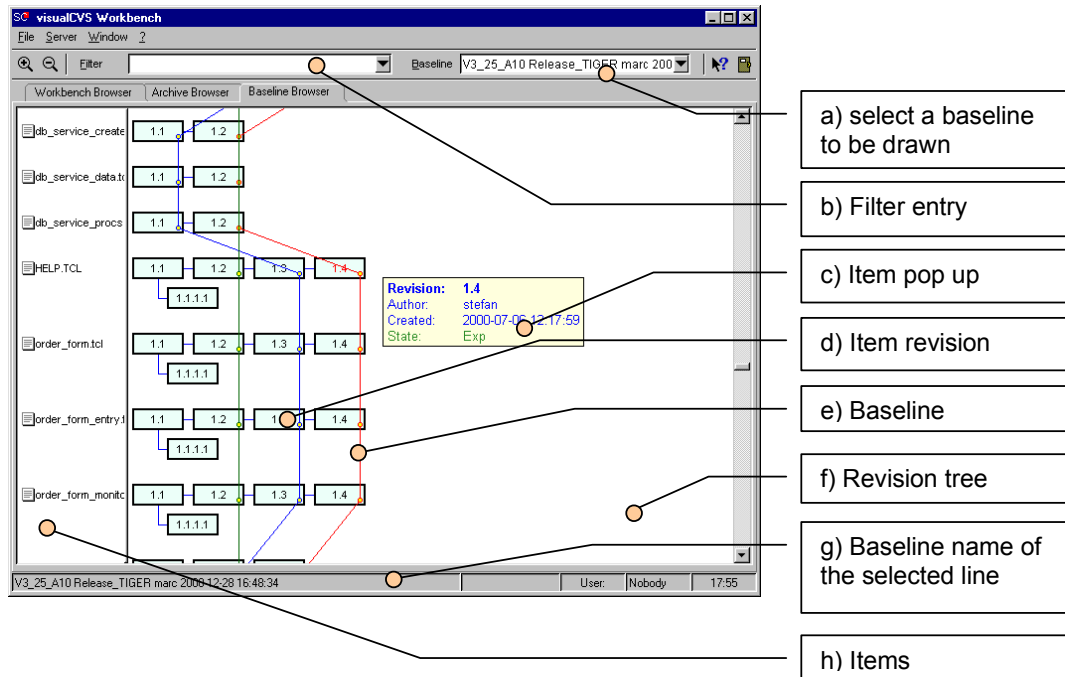


Figure: Baseline view

Creating a Baseline View

Workbench Browser:

- Select the desired items or directories with the mouse pointer
- Invoke the function **Create Baseline View** from within the context menu
 - click on the right mouse button
 - select the sub-menu **Baselines**
 - select the menu entry **Create baseline view**

Archive Browser:

- Select the desired items or directories with the mouse pointer
- Invoke the function **Create Baseline View** from within the context menu
 - click on the right mouse button
 - select the sub-menu **Baselines**
 - select the menu entry **Create baseline view**

Note: When invoking the mouse menu within the archive tree, the function **Create baseline view within folder** considers all items within the selected archive folder whereas the function **Create baseline view including subfolders** considers all items beyond the selected archive folder (recursive).

Drawing Baselines

Baselines are drawn by selecting an entry of the combo box widget (see a Figure: Baseline view). The function interconnects all items with a specific baseline tag with a line.

The line is highlighted in case the baseline was already drawn before.

Navigate Along Baselines

Place the mouse pointer over a baseline (see e Figure: Baseline view) and click on the **right mouse button** to open the context menu. The menu offers the following functions:

Function	Description	Keyboard bindings
Previous item	jump to the previous item	p, b
Next item	jump to the next item	n, f
Goto begin	jump to the first item	
Goto end	jump to the last item	
Hide baseline	hide the line	

View Item Pop Up

The item pop up window appears, when the mouse pointer is placed over a item revision (see c Figure: Baseline view). The pop up window contains information about the item revision, the author, the date of the revision when the item was checked in into the repository and its state.

Revision:	1.1
Author:	stefan
Created:	2000-12-28 15:44:39
State:	Exp

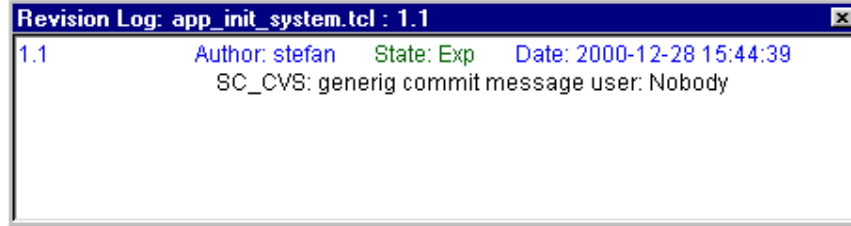
The pop up window disappears if the mouse pointer is moved.

View Item Log and Revision Log

The log window can be invoked by the mouse context menu. Place the mouse pointer over a item or a item revision (see *h, d* Figure: Baseline view) and click on the **right mouse button**, then select the function **show full revision log** or **show revision log**. The function can also be invoked by the l-key.

The window can be updated by selecting another item revision and again calling the function as described above.

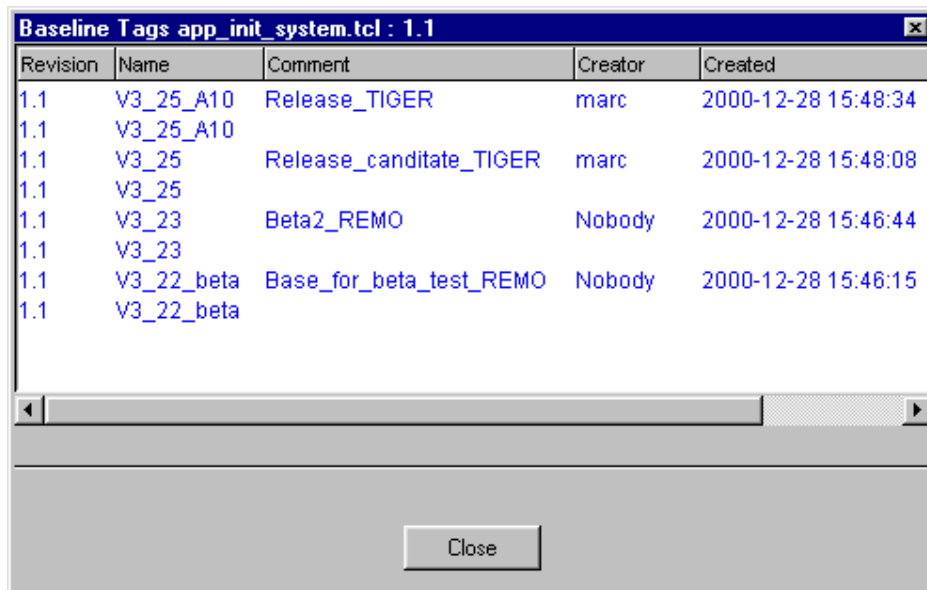
The log window can be closed by the **right mouse button** or by any key press.



View Baseline Tags

The log window can be invoked by the mouse context menu. Place the mouse pointer over a item or a item revision (see *h, d* Figure: Baseline view). and click on the **right mouse button**, then select the function **show all baseline tags** or **show baseline tags**. The function can also be invoked by the b-key.

The log window can be closed by the **close button**, the **right mouse button** or by any key press.



Viewing Differences of Revisions

Select two item revisions (see *d* Figure: Baseline view). by pointing and clicking on the **left mouse button**. The colour of the item revision changes to indicate the selection. Then open the context menu by clicking on the **right mouse button** and select the function **Diff revisions** to start the visualDiff tool. For more information on visualDiff see the section visualDiff Tool of this manual.

Filtering

The baseline view can be filtered by entering a filter sequence into the filter entry widget (see *b* Figure: Baseline view). Several filter criteria can be combined with + (AND) or | (OR). A possible filter sequence could be:

```
date>2001-02-15 + author=stefan | filename=a*
```

Filter	Description	Operators
Filename=demo.c	Filter the file name	<>=
date=2001-01-01 12:39:55	Filter item time tag	<>= no wildcard support time can be omitted seconds can be omitted
revision=1.3	Filter revision index	<>=
author=stefan	Filter for author	<>=
state=Exp	Filter for the state field	<>=
Comment	Filter for a revision log string	<>=

Zooming

The baseline view contains a zoom function. The function can be activated with the **left mouse button** (click and drag) or through the toolbar by clicking on the toolbar icons:



visualCVS controls a zoom stack. By pressing the **Escape key**, the view returns to the previous zoom level.

Keyboard Bindings

Within the baseline viewer the following keys are supported:

Key	Function	Comment
l	Show revision log, show full item log	If an item has the input focus the log window is invoked
b	Show baseline tags	If an item has the input focus the baseline tag window is invoked
Mouse Button 1	Select item	Select item to apply action to
Mouse Button 3	Open context menu	
Mouse Button 1+Drag	Zoom in	Keep the button pressed and move the mouse pointer. A red square defines the aerea to zoom into.
Mouse Wheel	Scroll up/down	
Home	Move to top	
End	Move to bottom	
PageUp	Page up	
PageDown	Page down	
Arrow-Up	Scroll up	
Arrow-Down	Scroll down	
Arrow-Left	Scroll left	
Arrow-Right	Scroll right	
+	Zoom in	
-	Zoom out	
Escape	Previous zoom level	

Within a selected baseline the following keys are supported:

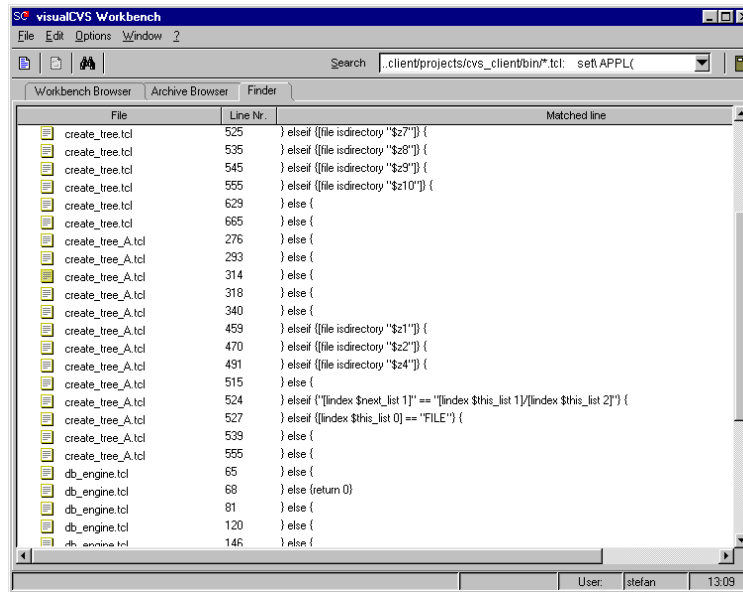
Key	Function	Comment
Mouse Button 1	Select baseline	
Mouse Button 3	Open context menu	
F	Jump to next item	
N	Jump to next item	
B	Jump to previous item	
P	Jump to previous item	

Within the baseline view log window the following keys are supported:


Key	Function	Comment
Mouse Button 3	Close window	
Mouse Wheel	Scroll up/down	
Home	Move to top	
End	Move to bottom	
PageUp	Page up	
PageDown	Page down	
Arrow-Up	Scroll up	
Arrow-Down	Scroll down	
Arrow-Left	Scroll left	
Arrow-Right	Scroll right	
Any other key	Close window	Remove the log window by pressing any key

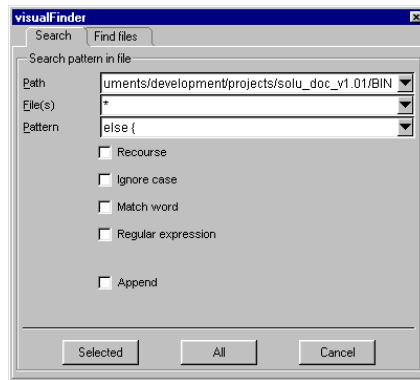
VisualCVS Finder

The visualCVS finder is a powerful search utility to find files by name as well as to search a pattern within files. The search result is presented in a link list. With double-click the file can be opened at the line where the match occurred (depends on the level of integration of your editor).



Searching through files

The search dialog can be invoked through the toolbar search button . The pattern is searched within the file content. The search result is presented in the finder window.



Examples:

Search by glob pattern matching style:

Pattern: else
Pattern: el.e

search lines which contain the patterns else
quote single character with .


Search by regular expression pattern:

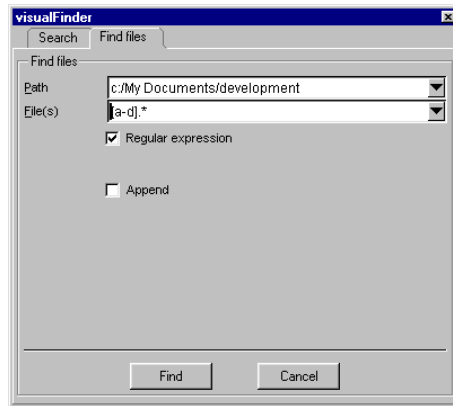
Pattern: ^set
Pattern: demo\$
Pattern: \$DEM([A-F0-9])

search for lines starting with "set"
search lines ending with "demo" (tailing \$ sign)
range A-F or 0-9

For more information on regular expression see Regular Expressions.

Finding files

The find files dialog can be invoked through the toolbar search button . It recursively searches the directory structure for file names to which the pattern applies. The search result is presented in the finder window.



Examples:

Search by glob pattern matching style:

File name pattern: *.c

* matches any character

File name pattern: a??*.c

? matches a single character

Search by regular expression pattern:

File name pattern: a.*.c

. matches a single character, .* matches any character

File name pattern: [a-d].*.c

matches all .c files starting with character a b c or d

For more information on regular expression see Regular Expressions.

Regular Expressions

Regular expressions are made up of normal characters and *metacharacters*. Normal characters include upper and lower case letters and digits. The metacharacters have special meanings and are described in detail below.

In the simplest case, a regular expression looks like a standard search string. For example, the regular expression "testing" contains no metacharacters. It will match "testing" and "123testing" but it will not match "Testing".

The table below lists metacharacters and a short explanation of their meaning.

Metacharacter	Description
.	Matches any single character. For example the regular expression <code>r.t</code> would match the strings <code>rat</code> , <code>rut</code> , <code>r t</code> , but not <code>root</code> .
\$	Matches the end of a line. For example, the regular expression <code>weasel\$</code> would match the end of the string "He's a weasel" but not the string "They are a bunch of weasels."
^	Matches the beginning of a line. For example, the regular expression <code>^When in</code> would match the beginning of the string "When in the course of human events" but would not match "What and When in the" .
*	Matches zero or more occurrences of the character immediately preceding. For example, the regular expression <code>.*</code> means match any number of any characters.
\	This is the quoting character, use it to treat the following character as an ordinary character. For example, <code>\\$</code> is used to match the dollar sign character (\$) rather than the end of a line. Similarly, the expression <code>\.</code> is used to match the period character rather than any single character.
[] [c1-c2] [^c1-c2]	Matches any one of the characters between the brackets. For example, the regular expression <code>r[ao]t</code> matches <code>rat</code> , <code>rot</code> , and <code>rut</code> , but not <code>ret</code> . Ranges of characters can be specified by using a hyphen. For example, the regular expression <code>[0-9]</code> means match any digit. Multiple ranges can be specified as well. The regular expression <code>[A-Za-z]</code> means match any upper or lower case letter. To match any character except those in the range, the complement range, use the caret as the first character after the opening bracket. For example, the expression <code>[^269A-Z]</code> will match any characters except 2, 6, 9, and upper case letters.
\{ \} \{i, \}	Match a specific number of instances or instances within a range of the preceding character. For example, the expression <code>A[0-9]{3}</code> will match "A" followed by exactly 3 digits. That is, it will match <code>A123</code> but not <code>A1234</code> . The expression <code>[0-9]\{4,6\}</code> any sequence of 4, 5, or 6 digits.

visualDiff Tool

The visualDiff tool is a graphical editor browse and merge differences of source files (Note: visualDiff is based on the open source tool tkDiff 3.05).

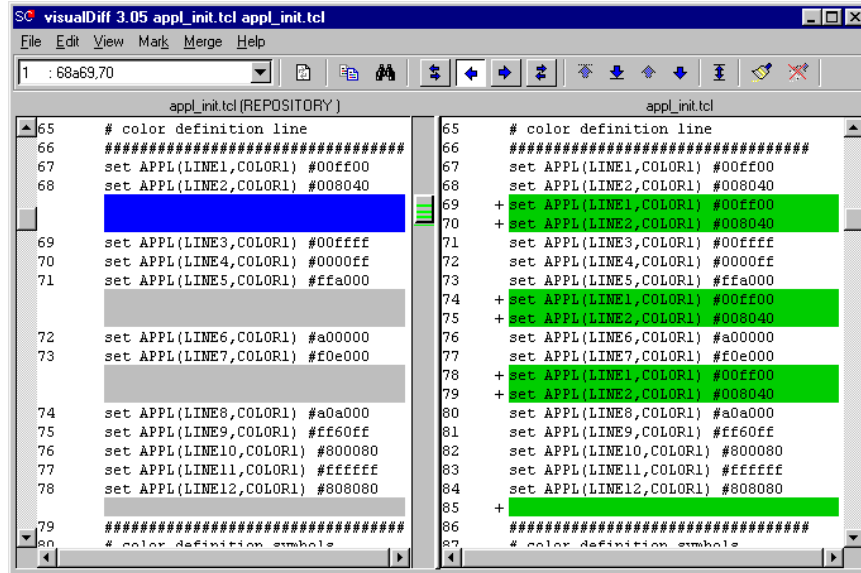


Figure: visualDiff Tool

Integrated Text Editor

Together with the standard distribution visualCVS contains a easy to use text editor called visualWordpad. If you would like to integrate your own programmer editor please refer to the section Editor Integration of this manual.

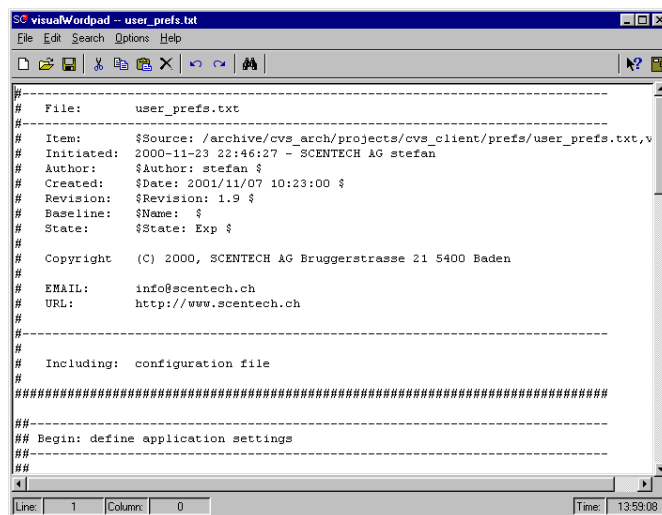


Figure: Integrated Text Editor

Configuration Options

Configuration Files

File Format

All configuration files are plain ASCII files which can be manipulated with any text editor. If the syntax of one of the configuration files is erroneous, an error is raised during the start of the application.

visualCVS Client

Application Preferences

Global application definitions are located within the configuration file. The configuration options are explained in the table below.

```
$installation_directory/prefs/user_prefs.txt
```

Session Files

visualCVS maintains a session file for each user. In case a session file is corrupted, it can be removed without damaging the installation.

The file is located:

Windows

```
$installation_directory/sessions/_visual_CVS_$USER
```

Unix

```
$HOME/.visual_CVS_$USER
```

visualCVS Server

Application Preferences

Global application definitions are located within the configuration file. The configuration options are explained in the table below.

```
$installation_directory/prefs/user_prefs.txt
```

Editor Integration

VisualCVS offers an API to integrate your own favourite editor. The following steps are required:

1. Create/modify an editor integration script
2. Adjust `user_prefs.txt` to point to the editor integration script

Create/modify an editor integration script:

The integration scripts are located within the directory `$installation_directory/prefs`. There is a range of integration scripts available for vim, emacs and some other editors. You can integrate the editor by creating your own script or by adjusting an existing script. On existing scripts you may need to adjust the path variable pointing to the editor.

Adjust `user_prefs.txt` to point to the editor integration script:

To connect to your editor script change the parameter `APPL(INTEGRATED_EDITOR)` by invoking the configuration editor of visualCVS (menu **Options/Configuration/Edit Configuration**). The variable must point to the integration script you plan to use.

To activate the changes, visualCVS needs to be restarted.

Note: on windows use / (slash) as path separator instead of \ (backslash). This applies to all platforms. Also use the windows shortname for path definitions if the path contains whitespaces.

Adjusting Keyboard Bindings

The keyboard binding can be modified in the file `$installation_directory/prefs/keyboard_binding.tcl`. The file contains a list of functions and the keys which trigger the function. To adjust the key bindings edit the text file with an editor.

The syntax of the key list: `{[key sym 1] [key sym n] [accelerator text]}` the accelerator text is optional (menu accelerator label within the visualCVS context menu)

Example update function:

In the following example, the update function is called on the keys +, F12 and Ctrl-u. The accelerator text is Ctrl-u.

```
set FUNCTION(KEY,UPDATE)      {<+><F12><Control-u> <Ctrl-u>}
```

Configuration Parameter

visualCVS Server

Global Application Definitions

<i>Variable</i>	<i>Function</i>	<i>Range</i>	<i>Description</i>
APPL(CVSROOT)	Path to the repository		Must point to a CVS repository. Use / (slash) instead of \ (backslash) on all platforms
APPL(CVSHOST)	Logical name		Logical name of the repository server
APPL(DB,PORT)	Port to listen to	2540	Must match with port definition of the visualCVS client
APPL(THREAD,MAX_REQUEST_IDLE_TIME)	Max idle time	120000 [ms]	Time in ms until a request is interrupted
APPL(THREAD,TIMEOUT)	Idle waiting time when connecting to thread pool	1000 [ms]	May not be modified
APPL(THREAD,NUM_OF_THREADS)	Number of threads in thread pool	8	Number of available working threads in thread pool. This number may be increased if a large number of users are accessing the repository

visualCVS Client

Global Application Definitions

Variable	Function	Range	Description
APPL(ADMIN,CVSROOT)	List of CVS archives		List entry 1: \$CVSROOT List entry 2: port nr. of DB server
APPL(EXEC,CVS)	Path to CVS program	Valid directory path name pointing to a CVS program	Defines which version of CVS to be used for visualCVS operations.
APPL(LANGUAGE)	language	[english]	Defines the GUI-language
APPL(TEMP)	Temp directory	Valid directory path name	Directory to store temporary files
APPL(SESSION,SAVE)	Save the session on exit	[0],[1]	Save the current session when the application is shut down
APPL(SESSION,MAXLENGTH)	Truncate lists to this length	[30]	Truncate lists when saving sessions for housekeeping
APPL(TIME,GMT)		[0],[1]	Compare time stamp relative to GMT. This flag shall always be set to 1 due to the way CVS resolves time stamps.
APPL(TIME,FORMAT)	Format of time tags	["%Y-%m-%d %T"]	Defines the output format of time tags. The keywords are expanded, other characters are interpreted as string. The default time format is expanded as "2001-01-01 12:00:00"

The following keywords are supported for APPL(TIME,FORMAT):

- %a Abbreviated weekday name (Mon, Tue, etc.).
- %A Full weekday name (Monday, Tuesday, etc.).
- %b Abbreviated month name (Jan, Feb, etc.).
- %B Full month name.
- %d Day of month (01 - 31).
- %H Hour in 24-hour format (00 - 23).
- %I Hour in 12-hour format (00 - 12).
- %j Day of year (001 - 366).
- %m Month number (01 - 12).
- %M Minute (00 - 59).
- %p AM/PM indicator.
- %S Seconds (00 - 59).
- %T Time as %H:%M:%S.
- %U Week of year (00 - 52), Sunday is the first day of the week.
- %w Weekday number (Sunday = 0).
- %W Week of year (00 - 52), Monday is the first day of the week.
- %y Year without century (00 - 99).
- %Y Year with century (e.g. 1990)

Logging

Variable	Function	Range	Description
APPL(LOG)	enable/disable logging function	[1], [0]	With the function enabled, the application writes status and debugging information into a log file
APPL(LOG,DIRECTORY)	directory to store the log files in	Valid directory path name	Define/create the location where to store the log files
APPL(LOG,LEVEL)	defines the log level	[-info], [-exec] [-debug]	Filter for the log messages: -info : some status messages -exec : CVS calls and answers -debug : debugging information

Workbench Browser

Variable	Function	Range	Description
APPL(CVS,WARNING,LEVEL)	Warning level when analysing cvs command result	[low], [medium], [high]	Low=show warnings and errors Medium=show errors High=show only severe errors
APPL(MERGE,CONFLICT,SHOW)	Show merge conflict left over files	[1],[0]	Show/hide files created by cvs on merge conflicts
APPL(FILENAME,CASE,CHECK)	Activate case check	[1],[0]	Activate a upper/lowercase check between archive filename and local filename (only on windows)
APPL(FILENAME,CASE,WARNING)	upper/lowercase warning	[1],[0]	Warning in state field of the workbench browser if there is a difference between upper/lowercase (only on windows)
APPL(FILE,AUTO_TEMPLATE)	Activate template mechanism	[1],[0]	Activates the template mechanism when creating new files
APPL(TEMPLATE_DIR)	Location of the templates	Valid directory path name	Directory which contains the template files

Archive Browser

Variable	Function	Range	Description
APPL(UPDATE,ARCHIVE)	Updating/non updating browser mode	[0],[1]	Use non-updating on slow server connection lines

Add/Import Format

Variable	Function	Range	Description
array set APPL { CVSEXTENSION,TYPE,.c 0 CVSEXTENSION,TYPE,.dll 0 CVSEXTENSION,TYPE,.bak 3 }	Defines how to add/import depending on file extensions	-ascii=[0] -binary=[1] -ignore=[2]	The list of known extensions is modified if the flag save in session of the import/add dialogue window is set. The list is stored in the session files.

Font Definitions

Variable	Function	Range	Description
APPL(FONT,MAINMENU)	font		Font for the menu bar
APPL(FONT,MENU)	font		Font for the pop-up menu
APPL(FONT,VIEW)	font		Font for the tree/list view entries
APPL(FONT,LABEL)	font		Font for labels
APPL(FONT,ENTRY)	font		Font for entry widgets
APPL(FONT,BALLOONHELP)	font		Font for the dynamic help windows
APPL(FONT,BASELINE,LABEL)	font		Font for the baseline window

Auxiliary Settings

Variable	Function	Range	Description
-----------------	-----------------	--------------	--------------------

CVS Command Line Flags

The CVS flag settings define the default CVS command options. Details about the configuration options can be found in the CVS manuals (see [1]).

Variable	Function	Range	Description
APPL(CVSFLAGS,GLOBAL)	Cvs flag	[""]	Global CVS flag
APPL(CVSFLAGS,COMMIT)	Cvs flag	[""]	Flags for CVS commit
APPL(CVSFLAGS,CHECKOUT)	Cvs flag	["P"]	Flags vor CVS checkout
APPL(CVSFLAGS,UPDATE)	Cvs flag	[""]	Flags vor CVS update
APPL(CVSFLAGS,ADD)	Cvs flag	[""]	Flags vor CVS add
APPL(CVSFLAGS,REMOVE)	Cvs flag	["-f"]	Flags vor CVS remove
APPL(CVSFLAGS,LOG)	Cvs flag	[""]	Flags vor CVS log
APPL(CVSFLAGS,STATUS)	Cvs flag	[""]	Flags vor CVS status
APPL(CVSFLAGS,TAG)	Cvs flag	[""]	Flags vor CVS tag

Enhanced Baseline Tags

The variables bellow control the extended baseline tag function. In order to add a baseline tag field the following variables needs to be created/modified:

```

TEXT(BASELINE,TAG, ORDER)
TEXT(BASELINE,TAG, name)           where name is a logical name for the baseline tag field
TEXT(BASELINE,TAG, name_B)
TEXT(BASELINE,TAG, name_MAXLENGTH)
TEXT(BASELINE,TAG, name_MINLENGTH)
TEXT(BASELINE,TAG, name_VIEWLENGTH)

```

Variable	Function	Range	Description
TEXT(BASELINE,TAG,EXTENDED)	Activate/deactivate extended baseline tag function	[1],[0]	
TEXT(BASELINE,TAG,SEPARATOR)	Separator character	[%]	Field separator character for the baseline tag string
TEXT(BASELINE,TAG, FIELDLENGTH)			
TEXT(BASELINE,TAG, ORDER)	Order of the fields	{name comment %USER% %TIME%}	Defines the order of the entry fields
TEXT(BASELINE,TAG, EXPANDCOLUMN)	Column to be expanded		Defines which column shall be expanded within the baseline tag window
TEXT(BASELINE,TAG, VIEWWIDTH)	With	[80]	With of the baseline tag window (in characters)
TEXT(BASELINE,TAG, VIEWHEIGHT)	Height	[10]	Height of the baseline tag window (in characters)
TEXT(BASELINE,TAG, <i>name</i>)	New field " <i>name</i> "		Field definition, new field called <i>name</i>
TEXT(BASELINE,TAG, <i>name_B</i>)			Dynamic balloonhelp of the entry field
TEXT(BASELINE,TAG, <i>name_MAXLENGTH</i>)	max length of <i>name</i>		Defines the max length of the user entry
TEXT(BASELINE,TAG, <i>name_MINLENGTH</i>)	Min length of <i>name</i>		Defines the min length of the user entry
TEXT(BASELINE,TAG, <i>name_VIEWLENGTH</i>)	View length of <i>name</i>		Defines display length of the entry field