AccuRev

Defining Change in SCM

The TimeSafe® Configuration Management System

AccuRev User's Guide

(Graphical User Interface)

Version 3.8

August, 2005

AccuRev User’s Guide (GUI)
August, 2005

Copyright © AccuRev, Inc. 1995–2005
ALL RIGHTS RESERVED

TimeSafe, **AccuRev**, and **StreamBrowser** are registered trademarks of AccuRev, Inc.
Java is a trademark of Sun Microsystems, Inc.
Rational and **Rational Rose** are trademarks of IBM Corporation.

Table of Contents

Overview of the AccuRev GUI

This document provides an orientation to the AccuRev Version 3.x graphical user interface (GUI). Before continuing, make sure that:

- You have installed the AccuRev client software on your computer.
- The AccuRev server software has been installed on a computer to which you have a network connection. Running the client and server software on the same machine is fine, too; this is typical for a pre-sales evaluation of the product.
- You have read the *AccuRev Concepts Manual*. This is important, because AccuRev works differently than CM systems with a branches-and-labels architecture.

Getting Started

You can start the AccuRev GUI from the desktop (Windows only) or from a command shell:

- On the Windows desktop, double-click the **AccuRev** icon. There is also an **AccuRev** entry in the **Start > Programs** menu.
- In a command shell, use the **acgui** command:

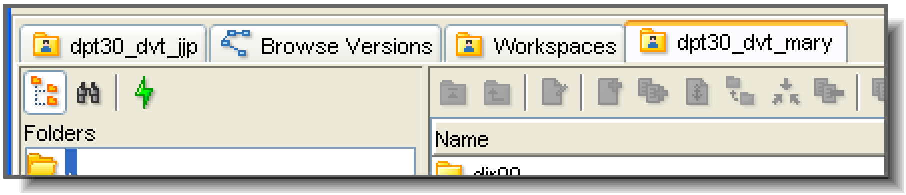
```
> acgui                 (Windows)
> acgui &               (Unix)
```

The way the GUI window initially appears varies. It might be blank, or it might open one or more tabs, displaying your work from the preceding GUI session. Such tabs can contain the File Browser, which displays the contents of an individual workspace, or the Stream Browser, which provides an overall display of your development environment. (Don't worry if you don't know what a "workspace" is. The explanation is just below.)

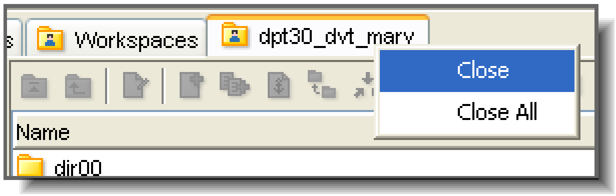
Multiple Tabs

The AccuRev GUI window uses multiple tabs (or "windows within a window") to enable you to switch quickly among several activities. For example, you might wish to switch among:

- Viewing the contents of your main workspace, using the **File Browser**.



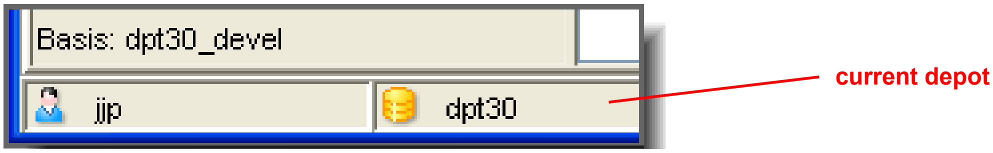
- Browsing through all the versions of a particular file.
- Viewing a list of all workspaces (or just the workspaces that belong to you).
- Viewing the contents of one or your colleague's workspaces.



At any time, you can “clean up”, by closing one or more of the tabs. Right-click on a tab area to display the menu shown here.

Typically, the AccuRev repository is organized into multiple depots. Each depot stores the complete history of one particular directory tree. For example, there might be separate depots for the development, testing, and documentation groups.

Most GUI tabs display the data from one particular depot. When you’re using a particular tab, the associated depot is termed the current depot. Its name is displayed at the bottom of the overall GUI window.

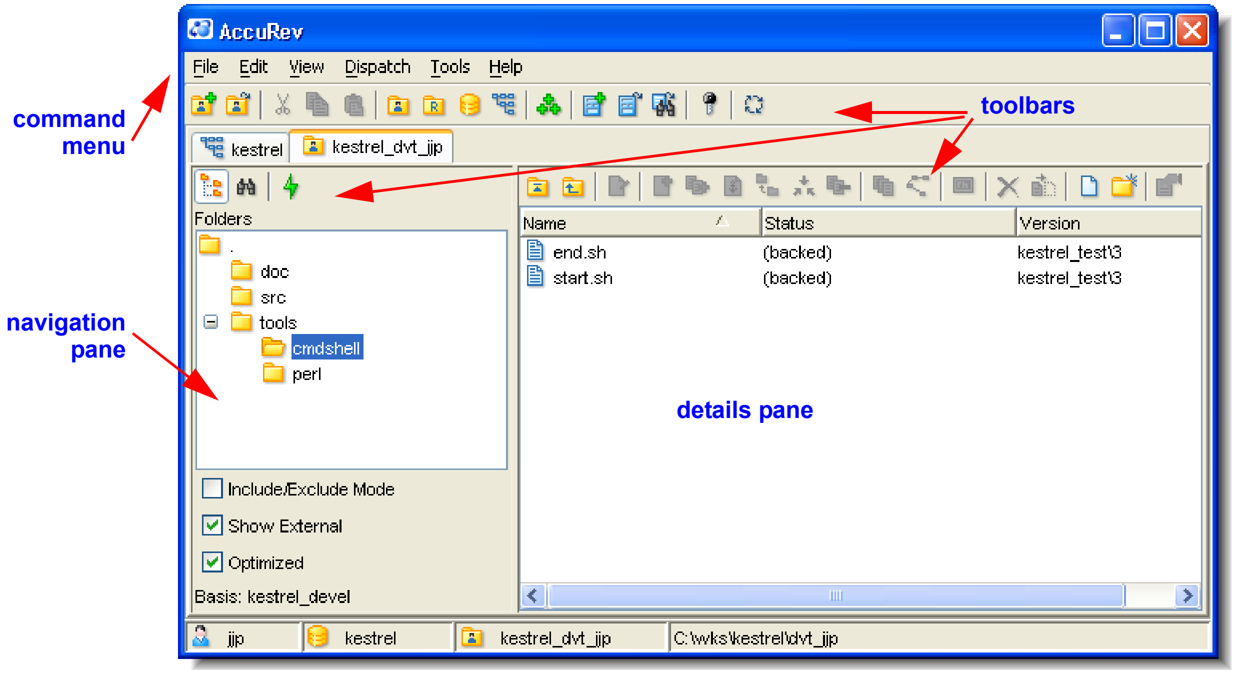


Workspaces

The AccuRev GUI makes it easy to view, monitor the status, and change the contents of files located in workspaces. An AccuRev workspace is a directory hierarchy on your hard disk, containing files that are being managed by AccuRev. You can create any number of workspaces — different ones for different development projects.

> Note: usually, you can think of a workspace as simply containing a collection of files. Sometimes, though, it helps to adopt AccuRev’s viewpoint: each file is a version-controlled element; your workspace contains a copy of a certain version of that element.

The format of the File Browser tab resembles that of Windows Explorer. It contains a navigation pane and a details pane. You can execute AccuRev commands by selecting from the pull-down command menu or by clicking toolbar buttons.



command menu

toolbars

navigation pane

details pane

Name	Status	Version
end.sh	(backed)	kestrel_test\3
start.sh	(backed)	kestrel_test\3

☐ Include/Exclude Mode

☑ Show External

☑ Optimized

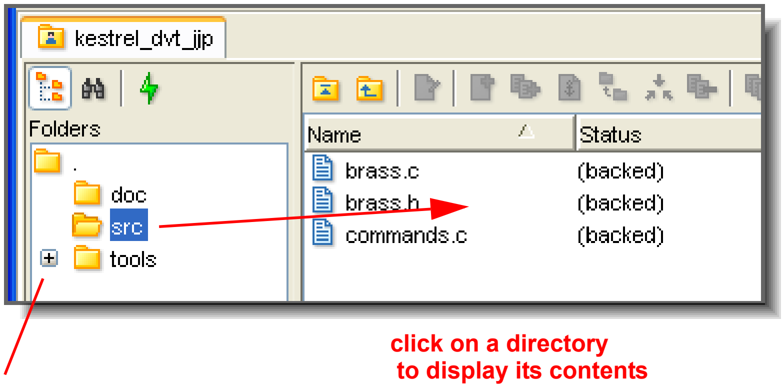
Basis: kestrel_devel

The navigation pane shows the directory hierarchy of one workspace. The details pane (usually) shows the contents of one of the workspace's directories. Each of these panes has its own toolbar, separate from the GUI window's overall toolbar.

The following sections describe how to get around and invoke commands in the File Browser. If you're familiar with Windows Explorer (or any number of similar programs), there won't be any surprises. And if you're in a hurry, you might want to skip down to section *Details Pane: File Status and Workspace Views* on page 5.

Workspace Navigation

Use the "standard explorer" gestures to navigate the hierarchy:



Name	Status
brass.c	(backed)
brass.h	(backed)
commands.c	(backed)

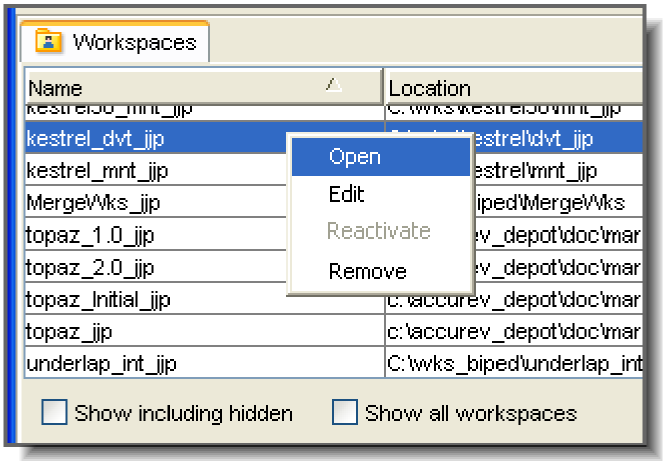
click on a "–" control to hide subdirectories
click on a "+" control to show subdirectories

click on a directory
to display its contents
in the details pane

To view the contents of another workspace:

- Select **View > Workspaces** from the command menu.
- In the **Workspaces** tab, right-click the desired workspace, and select **Open** from the context menu.

By default, the **View Workspaces** tab shows only workspaces that belong to you. To see workspaces that belong to other users, use the **Show All Workspaces** checkbox at the bottom of the tab.



Name	Location
kestrel30_mnt_jjp	C:\wks\kestrel30\mnt_jjp
kestrel_dvt_jjp	estrel\dvt_jjp
kestrel_mnt_jjp	estrel\mnt_jjp
MergeWks_jjp	iped\MergeWks
topaz_1.0_jjp	v_depot\doc\mar
topaz_2.0_jjp	v_depot\doc\mar
topaz_Initial_jjp	c:\accurev_depot\doc\mar
topaz_jjp	c:\accurev_depot\doc\mar
underlap_int_jjp	C:\wks_biped\underlap_int

Executing AccuRev Commands

AccuRev commands are invoked in the standard “noun-verb” manner: you select one or more files in the details pane; then, you select the command to be executed on those files.

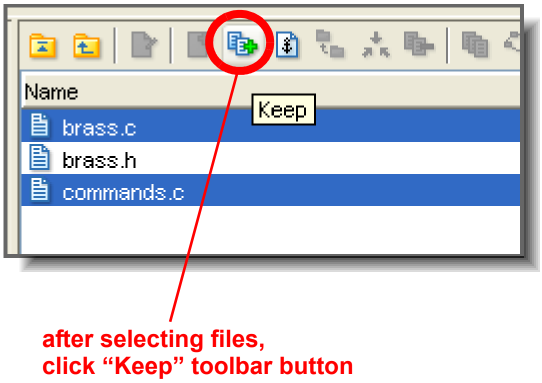
Selecting Files

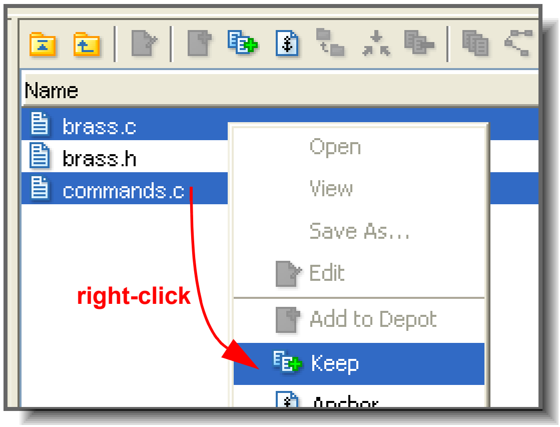
The file-selection gestures are the same ones used by Windows Explorer:

- To select a file, click it with the left mouse button.
- To select a contiguous range of files, click-and-drag with the left mouse button.
- To add or subtract a file from an existing selection, hold down the **Ctrl** key and click that file with the left mouse button.
- To extend an existing selection to a particular file, hold down the **Shift** key and click that file with the left mouse button.
- To open a file with a text editor, double-click on it (left mouse button).

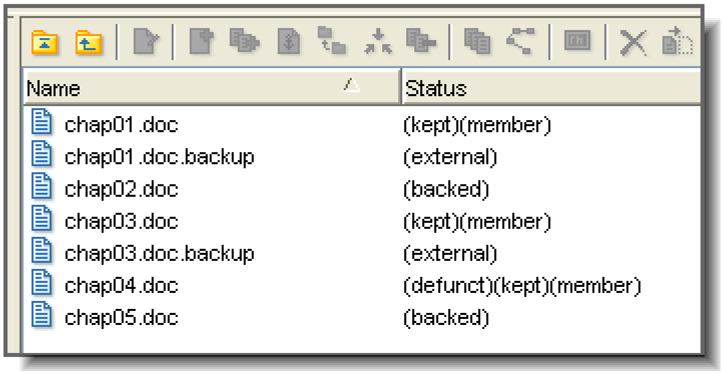
Selecting a Command

After you’ve selected one or more files, you select a command by clicking a toolbar button or by right-clicking to bring up a “context menu” of commands:





Details Pane: File Status and Workspace Views



Name	Status
chap01.doc	(kept)(member)
chap01.doc.backup	(external)
chap02.doc	(backed)
chap03.doc	(kept)(member)
chap03.doc.backup	(external)
chap04.doc	(defunct)(kept)(member)
chap05.doc	(backed)

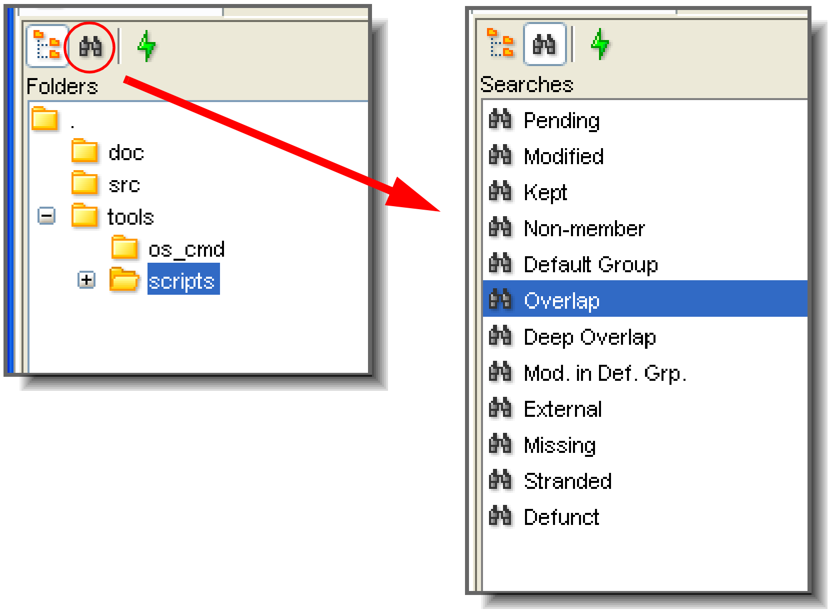
By default, the details pane displays the files in one directory — the one currently selected in the navigation pane. This is called the directory view of the workspace. For each file, the **Status** column shows one or more status flags.

The **(backed)** flag means that you haven’t made any changes to the file. The **(kept)** flag means that you’ve made some changes, and have stored them in a “private” version, visible only in this workspace. The **(modified)** flag means that there are changes to the file that have not been “kept”. (For more on status flags, see *Element Status* on page 23 of the *AccuRev User’s Guide (CLI)*.)

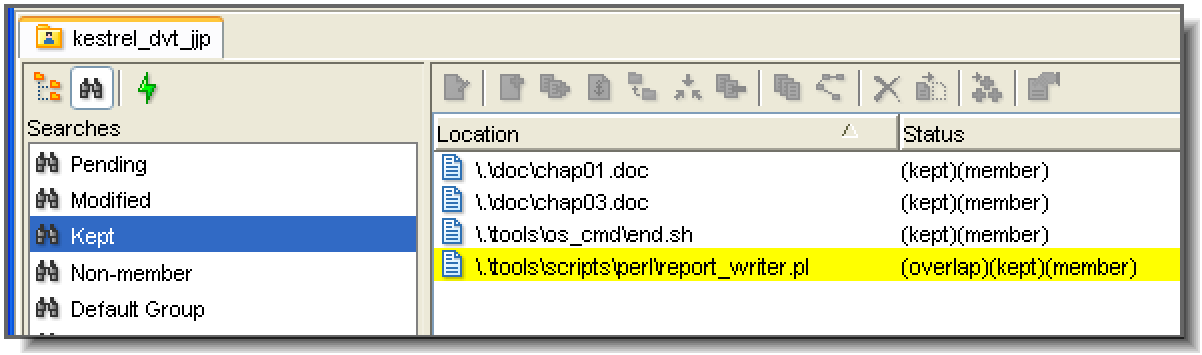
The AccuRev GUI search the workspace, as well, enabling you to see “just the files you care about”, instead of “all the files”. Which files do you care about? In general, a workspace contains a copy of the entire source base, which might include hundreds, or perhaps thousands, of files. But for a given development project, you’ll probably modify only a handful of the files. You need the others for general reference, and to enable you to perform software builds and tests in the workspace.

For example, you may have changed files in several different directories, and for each one kept a private version. So you may “care about” all the files in the workspace whose status flag is **(kept)**. Use the kept search to see exactly the information you want: To use this search:

- Click the **Searches View** button in the navigation pane. The tree control disappears, replaced by a list of search criteria.



- Select the **Kept** search:



Location	Status
\.\doc\chap01.doc	(kept)(member)
\.\doc\chap03.doc	(kept)(member)
\.\tools\os_cmd\end.sh	(kept)(member)
\.\tools\scripts\perl\report_writer.pl	(overlap)(kept)(member)

Note that the Kept search displays complete pathnames within the workspace hierarchy. That's because the search view shows files located throughout the entire workspace, not just in a particular subdirectory. (The fact that the directory hierarchy disappears from the navigation pane emphasizes this point.)

Similarly, the Modified search makes it simple to locate all files in the workspace that have a **(modified)** status flag. In fact, the GUI provides quite a few search criteria; you can always locate all the files you care about, simply and quickly.

Depot Setup

We'll assume that a depot (in the AccuRev repository) has already been set up for your use. Instructions for creating a depot and populating it with files to be version-controlled are included in the "Quick Evaluation" chapters in *AccuRev Technical Notes*.

Getting Work Done with the GUI

AccuRev does not force you to proceed with your software development work in any particular way. We're here to help you, not put you in a straitjacket! The workflow described in the following sections shows how you can put the AccuRev GUI to good use, but it certainly isn't the only way to get your work done.

Our example workflow includes these steps:

- *Creating a Workspace*
- *Editing Source Files*
- *Checkpointing — Saving Private Versions*
- *Concurrent Development — Incorporating Other Developers' Work*
- *Making Your Changes Public*
- *Concurrent Development — When Streams Collide*
- *Getting in Touch With Your Past*

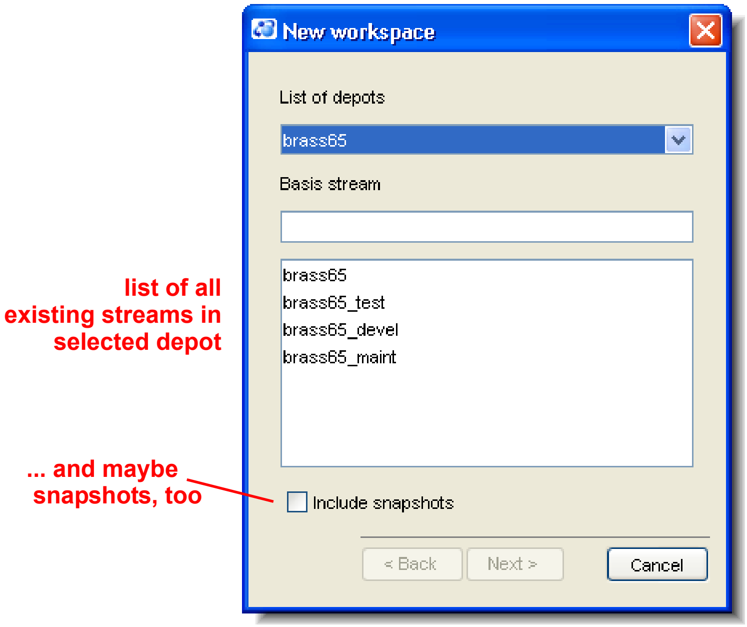
Creating a Workspace

A depot contains a hierarchy of streams. Any stream can have one or more workspaces based on ("backed by") it. You can always use the command **File > New > Workspace** on the main menu to create a new workspace. If you're currently working with some existing stream (for example, in the StreamBrowser tool), you can also use the **New Workspace** command on the stream's context menu.

Workspace creation is handled by a wizard: multiple screens that you navigate by clicking **Next** and **Previous** buttons. The following screen shots show the steps:

- **Selection of backing stream**: The first screen depends on whether you have a "current stream" context. If you do, the wizard assumes that you want the current stream to be the backing stream of the new workspace.

  Otherwise, the wizard presents a screen on which you select a particular stream in a particular depot, to be the backing stream. You can also choose a snapshot ("frozen stream") to back a workspace.



New workspace

List of depots: brass65

Basis stream:

brass65
brass65_test
brass65_devel
brass65_maint

□ Include snapshots

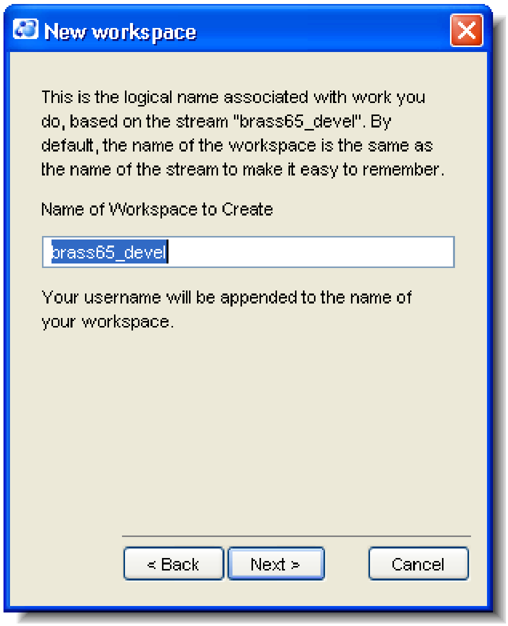
list of all existing streams in selected depot
... and maybe snapshots, too
< Back
Next >
Cancel

- **Name of Workspace to Create**: Click **Next** to accept the wizard's offer to name the workspace after the backing stream. The suffix _*<principal-name>* is appended automatically to the workspace name.

  This makes it easy to set up a backing stream and a set of like-named workspaces, one for each user:

  backing stream:
  brass65_devel

  workspaces:
  brass65_devel_mary
  brass65_devel_jjp
  brass65_devel_derek

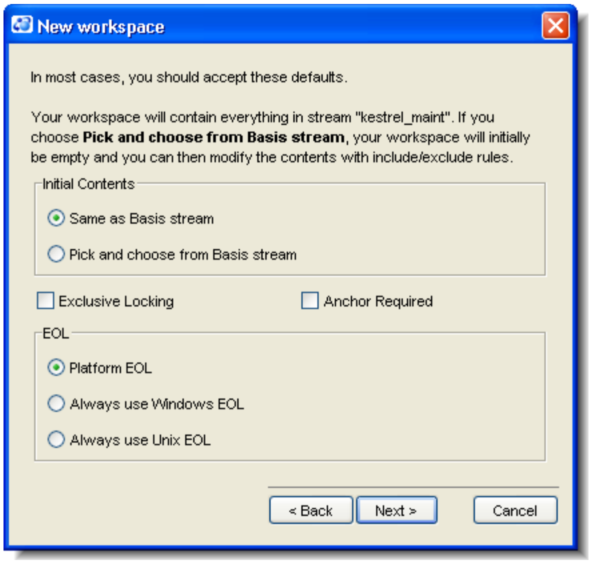


This is the logical name associated with work you do, based on the stream "brass65_devel". By default, the name of the workspace is the same as the name of the stream to make it easy to remember.

Name of Workspace to Create: brass65_devel

Your username will be appended to the name of your workspace.

- **Workspace options**: Click **Next** to accept the defaults for these workspace options:



In most cases, you should accept these defaults.

Your workspace will contain everything in stream "kestrel_maint". If you choose **Pick and choose from Basis stream**, your workspace will initially be empty and you can then modify the contents with include/exclude rules.

Initial Contents

● Same as Basis stream

○ Pick and choose from Basis stream

☐ Exclusive Locking ☐ Anchor Required

EOL

● Platform EOL

○ Always use Windows EOL

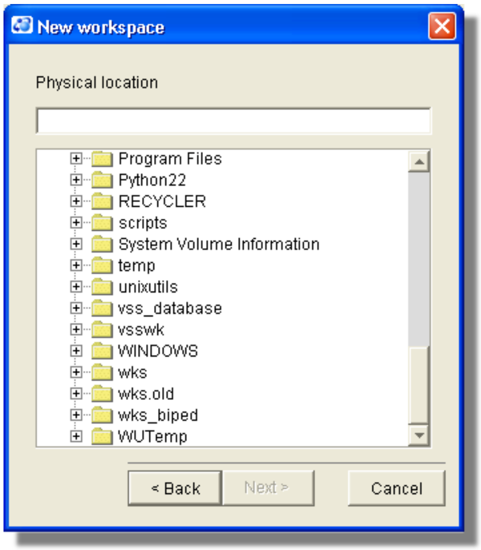
○ Always use Unix EOL

  - **Initial Contents**: By default, all the elements in the backing ("basis") stream also appear in the workspace. You can use the Include/Exclude facility to select a particular set of files and directories to appear in the workspace.
  - **Type of Workspace**: By default, you can modify any file in a workspace at any time, without having to interact with AccuRev. And several users can work on the same file concurrently in their separate workspaces. But AccuRev also supports more restrictive development processes:

    Exclusive File Locking: This feature suppresses the ability of multiple users to work on the same file at the same time. When one user starts working on a file, others are prevented from doing so. By default, this feature is turned off, enabling parallel development in this workspace.

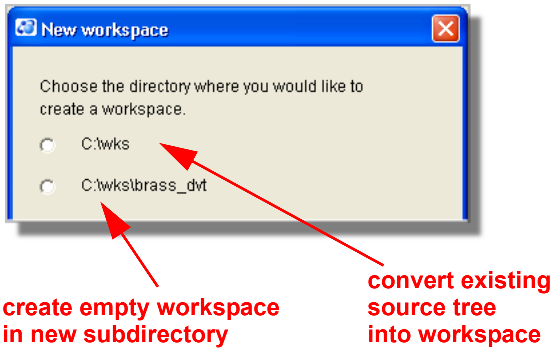
    Anchor Required: This feature requires you to run an AccuRev command, **Anchor**, before working on a file — even if no one else is currently working on it.
  - **Text-file line terminators**: By default, AccuRev uses the line terminator appropriate for the client machine's operating system. You can force Unix or Windows line terminators.

- **Physical Location**: You can create an empty workspace, or you convert an existing source tree into a workspace. In either case, navigate to an existing directory. Then, click **Next**. The wizard lets you choose whether to:
  - Convert the existing directory into a workspace
  - Create an empty workspace in a subdirectory of the existing directory. (The wizard automatically chooses a name for the subdirectory.)



Physical location

Program Files
Python22
RECYCLER
scripts
System Volume Information
temp
unixutils
vss_database
vsswk
WINDOWS
wks
wks.old
wks_biped
WUTemp



Choose the directory where you would like to create a workspace.

○ C:\wks

○ C:\wks\brass_dvt

create empty workspace in new subdirectory

convert existing source tree into workspace

Select your choice, then click **Finish**.

Editing Source Files

To edit a file, right-click it in the details pane and select **Edit** from the context menu. A double-click might bring up a text editor, but it might attempt to execute the file as a program instead. (Either way, be sure to click on the name, not on the icon.)

You can also start up a text editor or integrated development environment (IDE) independently of the AccuRev GUI, and edit files in that context.

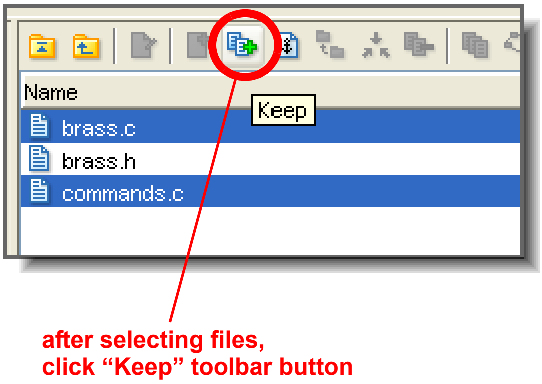
After saving your changes to the text, you may need to invoke the **View > Refresh** command in the GUI window, in order to update the files' status flags. Function key **F5** is a keyboard shortcut to this command.

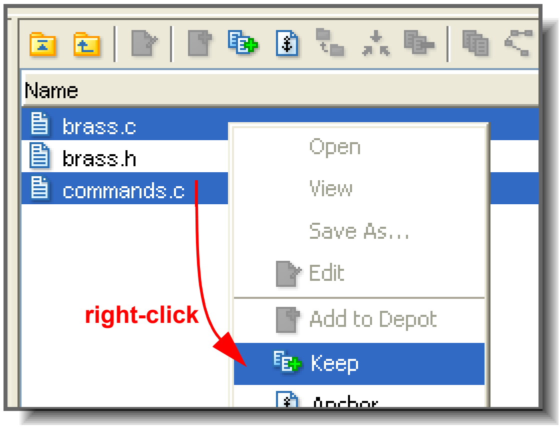
Checkpointing — Saving Private Versions

At any time, you can keep the changes you've made in one or more files. The **keep** command creates an official new version of a file. This includes making a permanent copy in the depot of the file's current contents. At any point in the future, you can revert to this version. This "save it just in case" procedure is commonly called checkpointing.

But **keep** does not make the new version public — it remains private to your workspace. Nobody else will see your changes yet. You can keep as many private versions (i.e. checkpoint the file as many times) as you want, without affecting or disrupting other people's work.

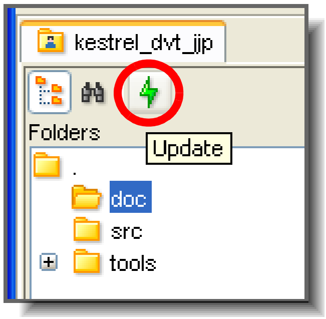
You can invoke the **keep** command from the context menu of a selected file (or group of files). There is also a **Keep** toolbar button:





If the files you wish to keep reside in several different directories, you can select all of them at once in a search view of the workspace. Use the **Modified** search. (See *Details Pane: File Status and Workspace Views* on page 5.)

Concurrent Development — Incorporating Other Developers’ Work

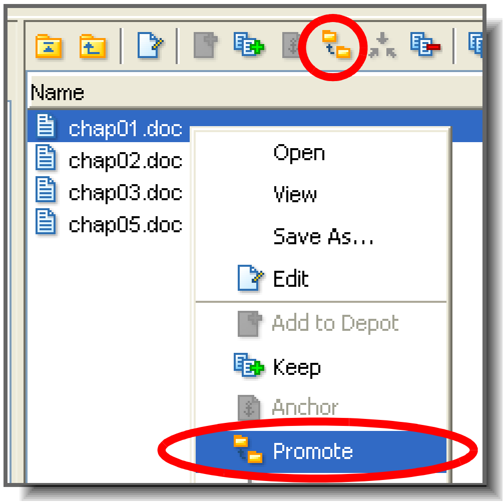


To incorporate other people's changes into your workspace, click the **Update** button, located above the navigation pane.

You won’t get the changes they have preserved with **keep** — those changes are private, as explained above. You’ll only get the changes they have made public with the **promote** command. (There’s more on **promote** just below.)

You can update your workspace as often or as infrequently as you wish.

Making Your Changes Public



To make your work available to others, you promote your kept files. You can invoke the **promote** command from the context menu of a selected file (or group of files). There is also a **Promote** toolbar button:

If the files you wish to keep reside in several different directories, you can select all of them at once from the kept view or pending view of the workspace. (See *Details Pane: File Status and Workspace Views* on page 5.)

Promoting a file makes the private version, which you previously created in your workspace with **keep**, into a public version. The public version resides in your workspace's backing stream. Anyone else whose workspace has the same backing stream can incorporate your promoted versions, using the **update** command.

Concurrent Development — When Streams Collide

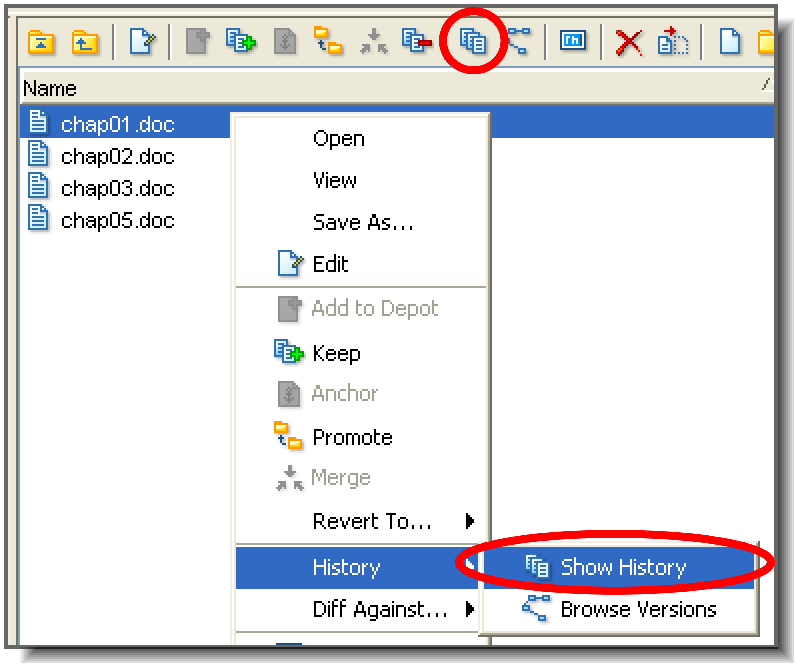
If someone else promotes a change to a file before you do, the changes are said to overlap. Before promoting your changes to the backing stream, you must merge the already-promoted changes into your work. Merging is a big topic; for more on AccuRev's Merge tool, see *The AccuRev Diff, Merge, and Patch Tools* on page 95.

Getting in Touch With Your Past

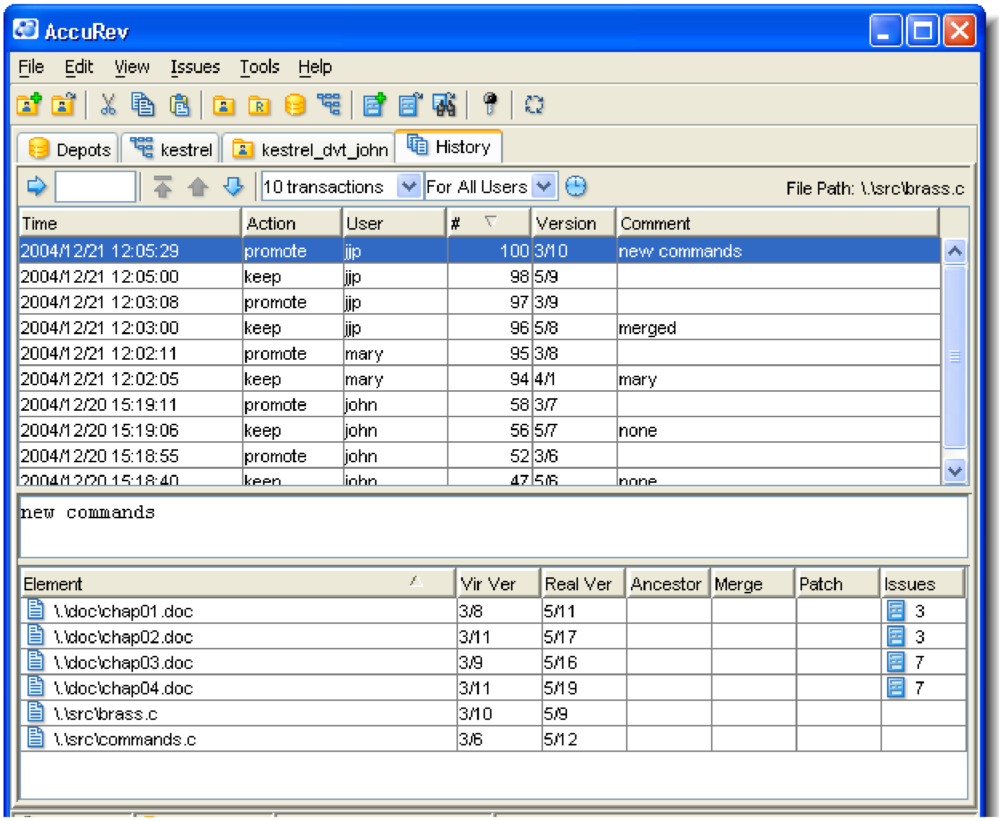
AccuRev keeps track of the complete history of each version-controlled file (or element). This history consists of the set of transactions that involved that particular element. Typically, most of an element's transactions are **keep** and **promote** actions. Transactions are also logged in other situations: when an element is first added to the depot (**create**, even though the command-name is **add**), when you rename it or move it to a different directory (**move**), when you incorporate someone else's changes into your work (**merge**), etc. In general, any change to a depot gets logged by a transaction.

To view the history of a selected element, invoke the **History > Show History** command from the element's context menu.

Alternatively, use the **Show History** button in the toolbar of the details pane.



A new tab opens, containing AccuRev's History Browser.



File Path: \.\src\brass.c

Time	Action	User	#	Version	Comment
2004/12/21 12:05:29	promote	jjp	100	3/10	new commands
2004/12/21 12:05:00	keep	jjp	98	5/9	
2004/12/21 12:03:08	promote	jjp	97	3/9	
2004/12/21 12:03:00	keep	jjp	96	5/8	merged
2004/12/21 12:02:11	promote	mary	95	3/8	
2004/12/21 12:02:05	keep	mary	94	4/1	mary
2004/12/20 15:19:11	promote	john	58	3/7	
2004/12/20 15:19:06	keep	john	56	5/7	none
2004/12/20 15:18:55	promote	john	52	3/6	
2004/12/20 15:18:40	keep	john	47	5/6	none

```
new commands
```

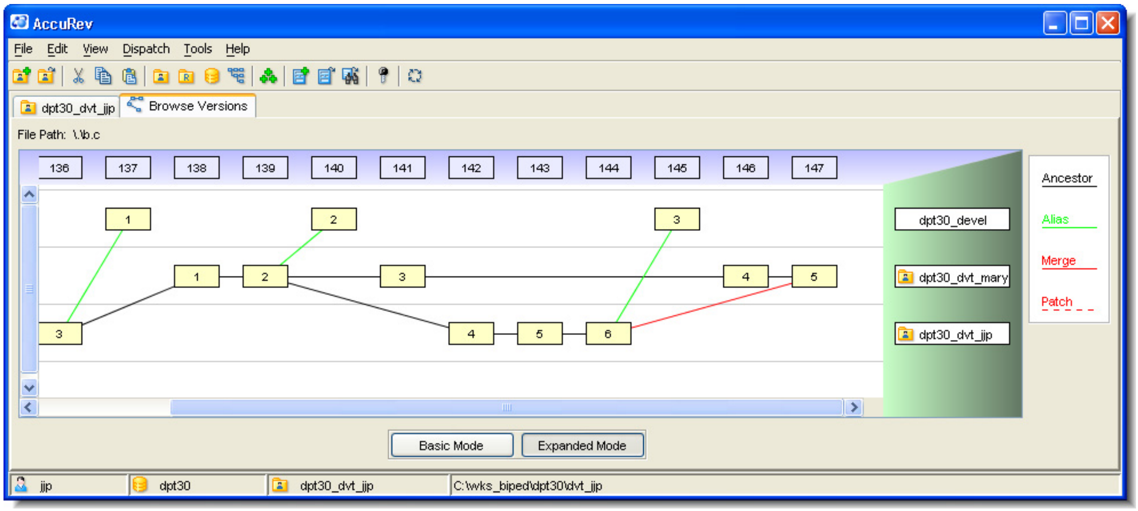
Element	Vir Ver	Real Ver	Ancestor	Merge	Patch	Issues
\.\doc\chap01.doc	3/8	5/11				3
\.\doc\chap02.doc	3/11	5/17				3
\.\doc\chap03.doc	3/9	5/16				7
\.\doc\chap04.doc	3/11	5/19				7
\.\src\brass.c	3/10	5/9				
\.\src\commands.c	3/6	5/12				

The History Browser tab has three panes. The table in the top pane summarizes the requested transactions, one transaction per row. The middle pane shows the log message (comment string) entered by the user who performed the transaction. The bottom pane lists each element involved in the transaction. For more on the History Browser, see *The History Browser* on page 115.

The Version Browser

You can also view the history of an element graphically. Select **History > Browse Versions** from the element's context menu, or use the **Browse Versions** toolbar button.

The Version Browser window displays all of an element's versions. This includes private versions you and your colleagues have created in your workspaces (**keep** command), along with public versions that have been placed in the backing stream (**promote** command). In a depot with a sophisticated stream hierarchy, the display may include versions in other streams, as well.



File Path: \.\b.c

C:\wks_biped\dpt30\dvt_jjp

Holding the mouse pointer over a version displays a help balloon containing more information about the version, including the creation log message.

The versions are connected with color-coded lines, indicating the way in which the versions were created:

- black lines indicate direct ancestry (edit existing version, then **keep** new version)
- green lines indicate promotions between streams
- solid-red lines indicate merges between streams
- dashed-red lines indicate patches — changes made in individual versions, rather than a stream's accumulated changes

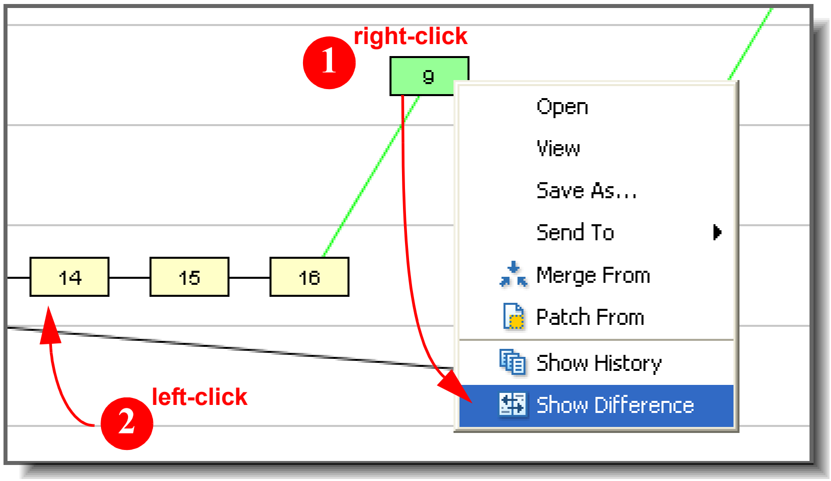
For more on ancestry, see *The Version Browser: Ancestry Tracking* on page 123.

Each version was created in a particular transaction; the white box at the top of the window shows the transaction number.

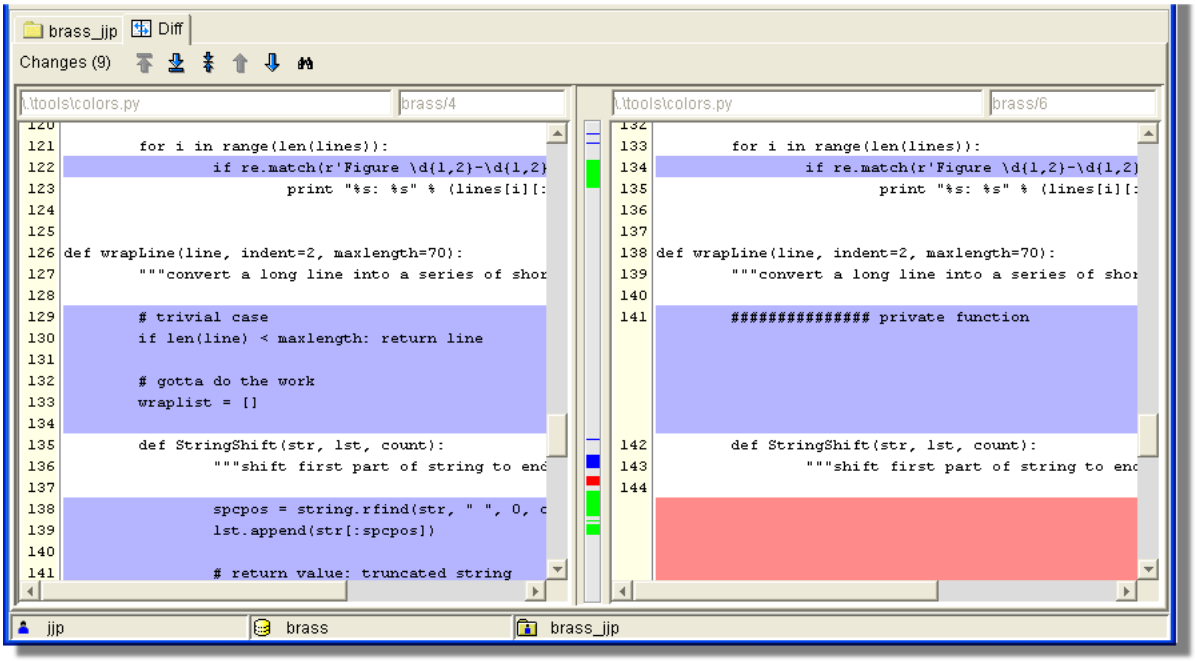
Comparing Versions

You can compare any two versions of an element:

1. Right-click a version, and select **Show Difference** from the context menu. The mouse pointer changes, with a Diff-tool icon annotating the standard arrow.
2. Left-click on any other version.



A new tab opens, in which the AccuRev Diff tool displays the two files side-by-side, highlighting their differences:



```
        for i in range(len(lines)):
                if re.match(r'Figure \d{1,2}-\d{1,2}
                        print "%s: %s" % (lines[i][:


def wrapLine(line, indent=2, maxlength=70):
        """convert a long line into a series of shor

        # trivial case
        if len(line) < maxlength: return line

        # gotta do the work
        wraplist = []

        def StringShift(str, lst, count):
                """shift first part of string to enc

                spcpos = string.rfind(str, " ", 0, c
                lst.append(str[:spcpos])

                # return value: truncated string
```

```
        for i in range(len(lines)):
                if re.match(r'Figure \d{1,2}-\d{1,2}
                        print "%s: %s" % (lines[i][:


def wrapLine(line, indent=2, maxlength=70):
        """convert a long line into a series of shor

        ############### private function
        def StringShift(str, lst, count):
                """shift first part of string to enc

```

(This is the default; you can configure the **Show Difference** command to invoke any file-difference program.) For more on the Diff tool, see *The AccuRev Diff, Merge, and Patch Tools* on page 95.

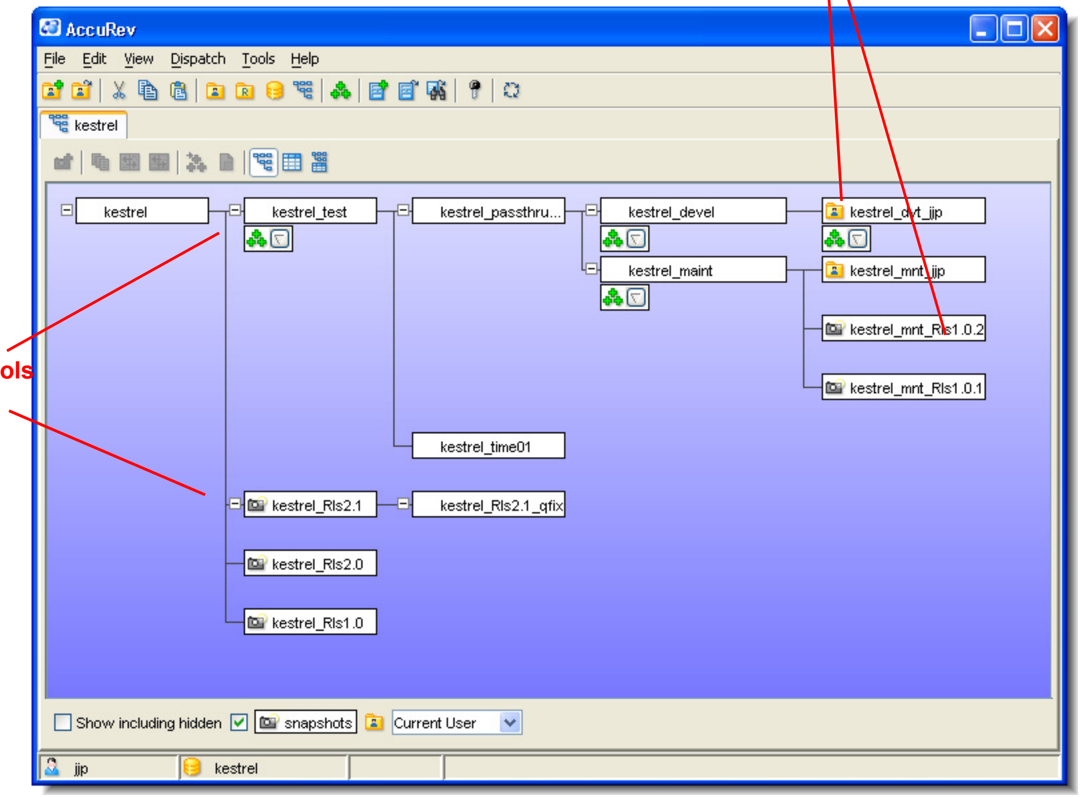
Advanced Operations

This overview document has focused on the day-to-day AccuRev tasks that are most likely to be performed by developers. The GUI makes it equally easy for project leaders (or release engineers, or quality-engineering specialists) to get their work done. We'll take a look at just one of AccuRev's advanced configuration management features.

Migrating Changes from Stream to Stream

AccuRev organizes all development of version-controlled files into streams. We've discussed the backing stream associated with developers' workspaces. More generally, a depot can have a complex hierarchy of streams, each one representing a separate development effort. There might be streams for Releases 3.2, 3.1.5, and 4.0 of a product, all active concurrently. In addition, there may be streams for special or experimental projects, for foreign-language translations, for emergency bugfixes, and so on. You can view, and work with, all of a depots streams using AccuRev's StreamBrowser. To open a StreamBrowser tab, use the **View > Streams** command or the **View Streams** toolbar button.

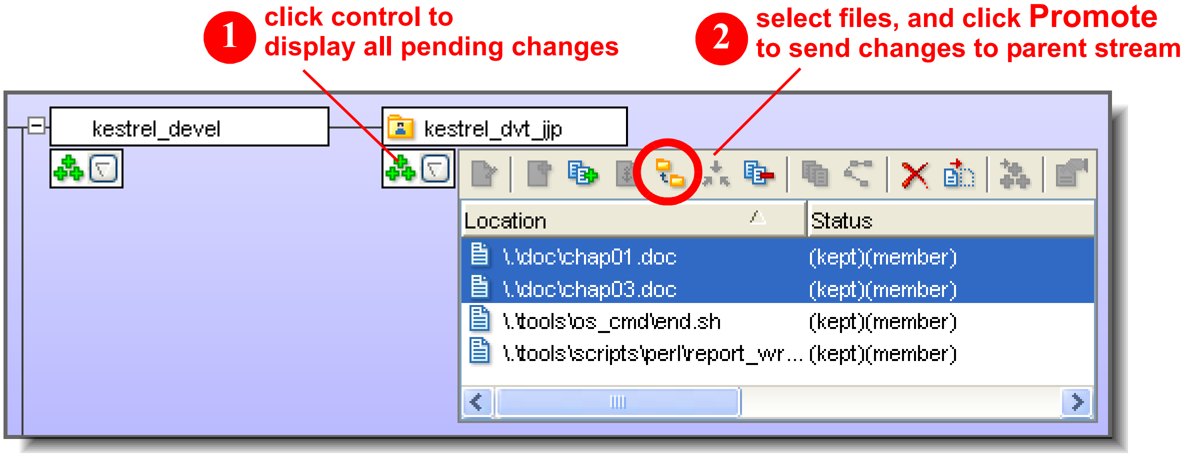
display of workspace streams and snapshots ("static streams") is optional



□ Show including hidden ☑ snapshots Current User

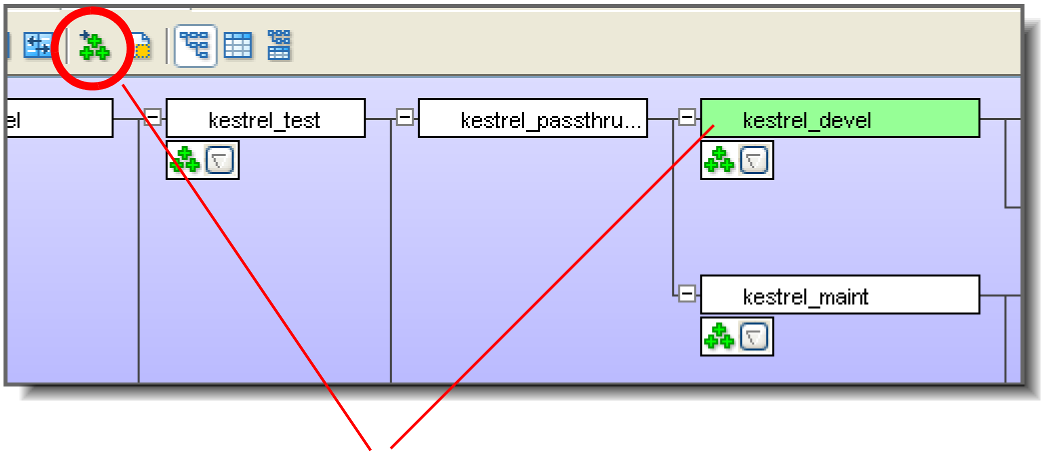
click expand/ collapse controls to determine which lower-level streams are displayed

Making Your Changes Public on page 11 describes how a developer uses the **promote** command to send changes from his own workspace to the backing stream. In a deep stream hierarchy, several promotions are required to propagate the changes all the way to the top. Each such promotion is simple.

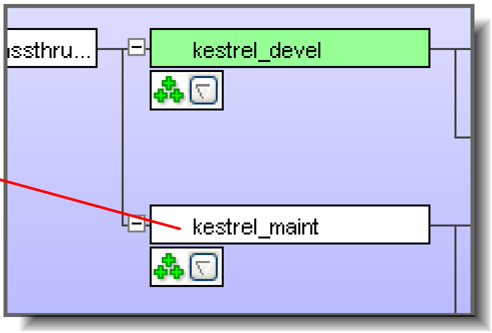

1 click control to display all pending changes
2 select files, and click Promote to send changes to parent stream
kestrel_devel
kestrel_dvt_jjp

Location	Status
\.\doc\chap01.doc	(kept)(member)
\.\doc\chap03.doc	(kept)(member)
\.\tools\os_cmd\end.sh	(kept)(member)
\.\tools\scripts\perl\report_wr...	(kept)(member)

It is often desirable to send changes to other streams, as well. For example, a manager might mandate that a developer's fix for a security hole be incorporated immediately into all active development efforts — i.e. into all active streams. The Change Palette makes it easy to migrate changes between streams, regardless of their location in the depot's stream hierarchy. This includes automatic determination of what changes need to be migrated, and control over the performing of any necessary conflict resolutions (merges). The following "storyboard" shows how to propagate changes from stream **kestrel_devel** to stream **kestrel_maint**:

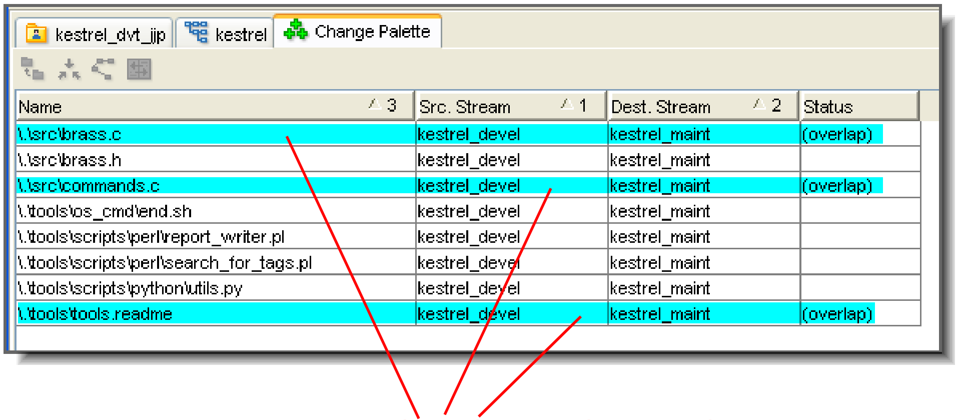


1 select “from” stream, and click **Send to Change Palette** toolbar button



2 select “to” stream

3 **Change Palette** tab appears



Name 3	Src. Stream 1	Dest. Stream 2	Status
\.\src\brass.c	kestrel_devel	kestrel_maint	(overlap)
\.\src\brass.h	kestrel_devel	kestrel_maint	
\.\src\commands.c	kestrel_devel	kestrel_maint	(overlap)
\.\tools\os_cmd\end.sh	kestrel_devel	kestrel_maint	
\.\tools\scripts\perl\report_writer.pl	kestrel_devel	kestrel_maint	
\.\tools\scripts\perl\search_for_tags.pl	kestrel_devel	kestrel_maint	
\.\tools\scripts\python\utils.py	kestrel_devel	kestrel_maint	
\.\tools\tools.readme	kestrel_devel	kestrel_maint	(overlap)

these three versions must be merged before they can be promoted to **kestrel_maint** stream

all other versions can be promoted to **kestrel_maint** immediately

Day-to-Day Usage of AccuRev

This document presents enough information for the individual user to work with AccuRev on a day-to-day basis. We provide a brief overview and discuss a handful of commands. It's a short chapter, because AccuRev is an elegantly simple configuration management system.

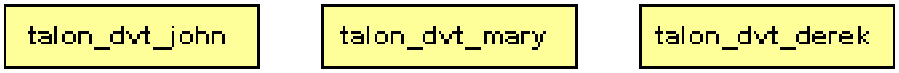
For a more complete overview of AccuRev usage, see *Overview of the AccuRev GUI* on page 1.

The AccuRev Usage Model

AccuRev's flexibility makes it easy to use for a variety of development scenarios. But like every software system, AccuRev has usage models that were foremost in the minds of its architects. This section describes the most common usage model.

AccuRev is a configuration management (CM) system, designed for use by a team of people (users) who are developing a set of files. This set of files might contain source code in any programming language, images, technical and marketing documents, audio/video tracks, etc. The files — and the directories in which the files reside — are said to be "version-controlled" or "under source control".

For maximum productivity, the team's users must be able to work independently of each other — sometimes for just a few hours or days, other times for many weeks. Accordingly, each user has his own private copy of all the version-controlled files. The private copies are stored on the user's own machine (or perhaps in the user's private area on a public machine), in a directory tree called a workspace. We can picture the independent workspaces for a three-user team as follows:

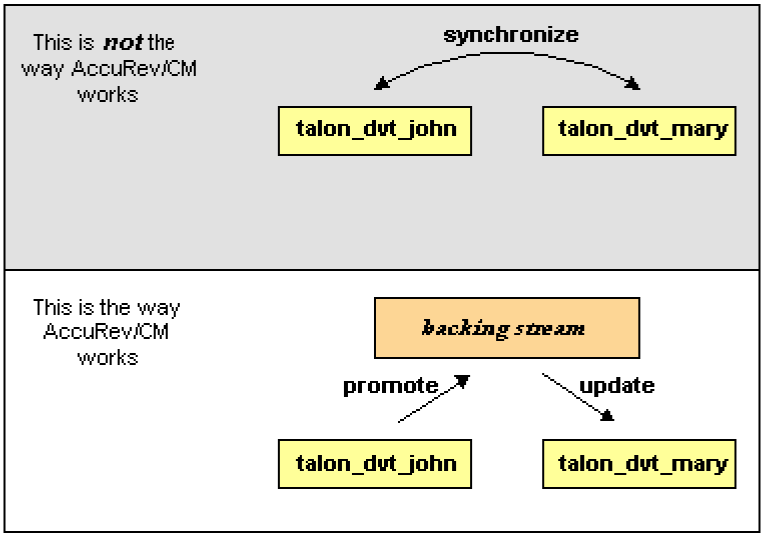


This set of users' workspaces uses the convention of having like names, suffixed with the individual usernames. AccuRev enforces this username-suffix convention. **talon_dvt** might mean "development work on the Talon product"; **john**, **mary**, and **derek** would be the users' operating system login names.

From AccuRev's perspective, development work in this set of workspaces is a continual back-and-forth between "getting in sync" and "getting out of sync":

- Initially, the workspaces are completely synchronized: they all have copies of the same set of version-controlled files.
- The workspaces lose synchronization as each user makes changes to some of the files.
- Periodically, users share their changes with each other. When **john** incorporates some or all of **mary**'s changes into his workspace, their two workspaces become more closely (perhaps completely) synchronized.

You might assume that the workspace synchronization process involves the direct transfer of data from one workspace to another. But this is not the way AccuRev organizes the work environment. Instead of transferring data directly between private areas (that is, between users' workspaces), AccuRev organizes the data transfer into two steps:



1. One user makes his changes public — available to all the other members of his team. This step is called promotion.
2. Whenever they wish, other team members incorporate the public changes into their own workspaces. This step is called updating.

The first step involves a public data area, called a stream. AccuRev has several kinds of streams; the kind that we're discussing here is called a backing stream. We'll see below how the data in this public stream "is in back of" or "provides a backstop for" all the private workspaces of the team members.

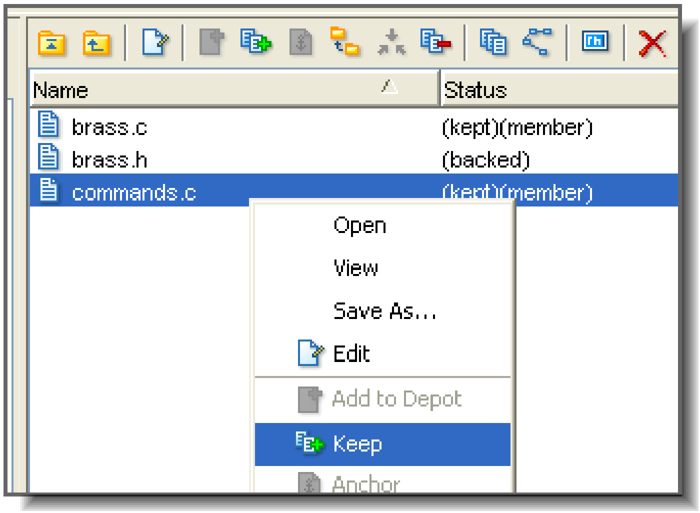
Change and Synchronization: The Four Basic Commands

With the usage model described above, you'll be able to accomplish most of your AccuRev work with four simple commands: **Keep**, **Promote**, **Update**, and **Merge**. We describe these commands in the following sections. Each section has a subsection titled "The Fine Print", in which we present additional usage details, notes on the way AccuRev implements certain features, and other tidbits of interest. You might want to skip over these sections on your first reading of this material.

Keep: Preserving Changes in Your Private Workspace

An AccuRev workspace is just a normal directory tree, in which you make changes to version-controlled files. You can work with the files using text editors, build and test tools, IDEs, etc., just as if the files weren't version-controlled at all. For example, you might edit a source file and invoke the editor's "Save" command a dozen times over the course of an hour or two. These operations don't involve AccuRev at all — they simply have the operating system change the contents and the timestamp of the file in your workspace.

You don't need to perform a "check out" operation or otherwise get permission from AccuRev before editing a file in your workspace. (Some legacy CM systems do impose such a regimen.)



Every so often, you want AccuRev to preserve the current contents of the file as an official new version of the file. You accomplish this using AccuRev's **Keep** command. This figure shows how to invoke the **Keep** command from a file's context (right-click) menu in the AccuRev File Browser tool, which has a Windows Explorer-like interface. You can also invoke **Keep** with the toolbar button.

You can continue modifying the file, then using **Keep** to preserve the latest changes, as often as you like. Other team members won't complain about "thrashing", because these new versions stay within your workspace; without affecting any other user's workspace.

AccuRev retains all the versions that you **Keep**. This makes it possible for you to roll back to any previous version you created.

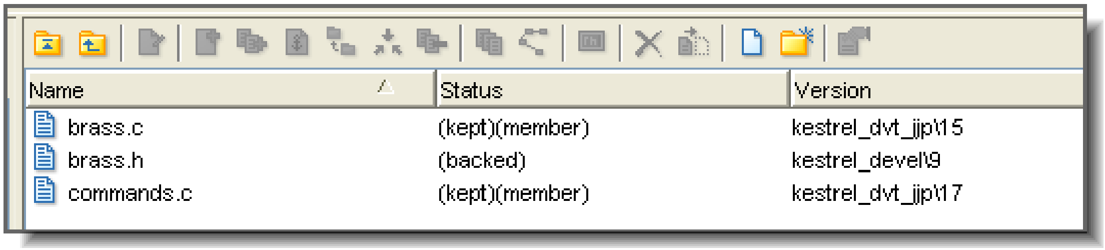
Several other operations are similar to **Keep**, in that they create a new version of a file in your workspace, without affecting any other user's workspace. The most important are:

- **Rename**/**Move**: You can rename a file or move it to a different directory (or both), using AccuRev commands. Other users will continue to see the file at its original pathname in their workspaces.
- **Defunct**: You can remove a file from your workspace with the AccuRev command **Defunct**. Other users will continue to see the file in their workspaces.

The Fine Print

We said above that AccuRev "retains all the versions that you **Keep**". But where? Each time you **Keep** a file, its current contents are copied to the AccuRev repository, located on the machine where the AccuRev Server runs. You don't need to care about the name and precise location of this copy. Each version you create has a version-ID, such as **talon_dvt_john/12** ("the **12**th version of this file created in workspace **talon_dvt_john**").

AccuRev keeps track of the status of each file in a workspace. After you **Keep** a file, the Status column in the AccuRev File Browser contains the indicator **(kept)**. It also contains the indicator **(member)**, meaning that the file belongs to the set of files you're actively working on. (See **Active and Inactive Files** below.) The Version column displays the version-ID.



Name	Status	Version
brass.c	(kept)(member)	kestrel_dvt_jjp\15
brass.h	(backed)	kestrel_devel\9
commands.c	(kept)(member)	kestrel_dvt_jjp\17

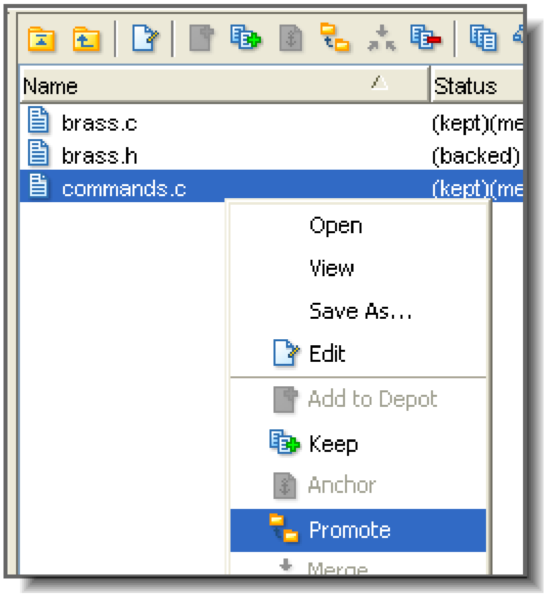
A change to the data within a file, recorded by **Keep**, is termed a content change; the change made by **Rename**/**Move** or **Defunct** is termed a namespace change. (Many CM systems don't handle namespace changes at all, or have very limited capabilities in this area.) As noted above, AccuRev saves a new copy of the file in the repository whenever you make a content change. But it doesn't need to copy the file when you make a namespace change; rather, the AccuRev Server just records the change in its database.

To perform version control on directories, AccuRev only needs to keep track of namespace changes — renaming, moving, or deleting a directory. Unlike some legacy CM systems, AccuRev doesn't need to record a new directory version when you make a content change — for example, adding a new file to the directory.

Promote: Making Your Private Changes Public

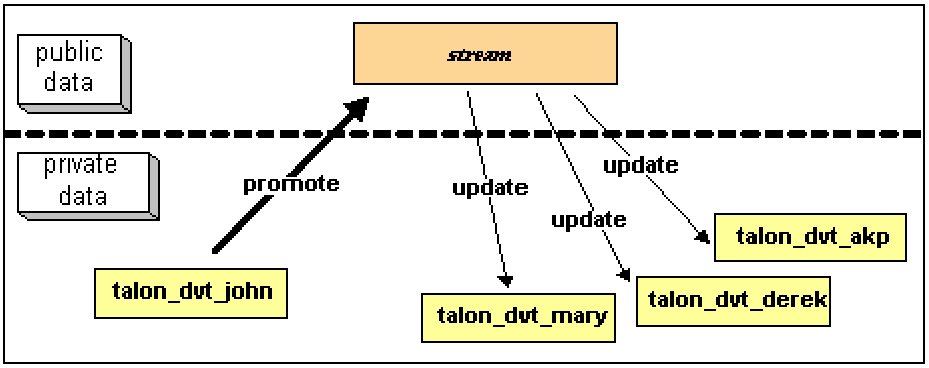
At some point, after you've used **Keep** to create one or more new, private versions of a file in your workspace, you typically want to share the changes you've made with the other team members. To make your (most recent) new version "available to the public", you promote it. This figure shows how to invoke the **Promote** command from a file's context (right-click) menu in the File Browser. You can also invoke **Promote** with the toolbar button.

Promoting your new version of a file does not automatically "push" it into the workspaces of the other team members. When a user decides that he's ready to incorporate versions of files that other team members have **Promote**d, he "pulls" them into his workspace with the **Update** command (details below).



Name	Status
brass.c	(kept)(me
brass.h	(backed)
commands.c	(kept)(me

Streams



The **Promote** command sends data to — and the **Update** command gets data from — a sophisticated AccuRev data structure called a stream. The stream acts as a “central data exchange” for the set of workspaces used by a development team. A stream also has a bit of “traffic cop” built in, preventing team members’ efforts from colliding and providing other mechanisms to control the flow of data.

A stream is not, as you might initially suppose, a set of copies of promoted files. Rather, it’s more like a list of version-IDs.

- the 4th version created in workspace **talon_dvt_john** of file **command.c**
- the 7th version created in workspace **talon_dvt_mary** of file **brass.c**
- ... etc.

In CM-speak, a stream is a configuration of a collection of version-controlled files. The term “stream” is apt, because it implies the ongoing change of a development project. Each time a user promotes a version of file **brass.c**, the stream configuration changes for that file — for example, from “the 5th version created in workspace **talon_dvt_derek**” to “the 7th version created in workspace **talon_dvt_mary**”.

Promotion and Parallel Development

Sometimes, AccuRev doesn’t allow you to promote a file to the development team’s stream, because another team member has already promoted the same file (after modifying it and performing a **Keep** on it). AccuRev is preventing you from overwriting your colleague’s change to the team’s shared stream. This situation is called an overlap: two users working at the same time on the same goal, to create the stream’s next version of a particular file.

Before you can promote your changes to the stream, you must first perform a merge on the file that has an overlap (details below).

Active and Inactive Files

As you work with a file using the commands described above, AccuRev considers the file to alternate between being *active* in your workspace and *inactive*:

- The file is initially *inactive*.
- When you create a new version in your workspace, using **Keep**, **Rename**/**Move**, or **Defunct**, the file becomes *active*.

- When you make your private version public, using the **Promote** command, the file becomes *inactive* again.

Later, you might restart this cycle, making the file active again by creating another new version of it. Alternatively, an update of your workspace might overwrite your inactive file with a newer version that another team member promoted.

AccuRev keeps track of the set of active files in your workspace. Officially, this set is called the default group. You might find it easier to think of it as the workspace's "active group" or its "active set".

The Fine Print

The **Promote** command doesn't copy the promoted version to the AccuRev repository. It doesn't need to. Promotion just gives an additional name to a version that *already exists* in the repository — having been placed there by a previous **Keep** command (or **Rename**/**Move** or **Defunct**). For example, promoting "the 7th version created in workspace **talon_dvt_mary**" might give that version the additional name "the 3rd version promoted to stream **talon_dvt**".

Just to emphasize the previous point: a stream does not reside in the file system, but in the database managed by the AccuRev Server. Promoting a version to a stream does not create a copy of a file; it just creates an additional file-reference in the Server database.

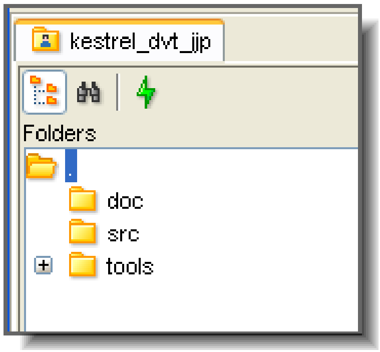
It might seem strange at first that deleting a file with the **Defunct** command makes the file *active*. The File Browser continues to list the file — with a **(defunct)** status — even though the file has been removed from your workspace's disk storage. This design feature enables AccuRev to implement the file-deletion operation using the same private-change/public-change scheme as all other changes.

We've discussed *the* stream behind a set of workspaces. But a typical development project has many streams, organized into a hierarchy. Promoting a version to a higher-level stream from a lower-level stream makes that version "even more public" — for example, available to users outside your local development team.

Update: Incorporating Others' Changes into Your Workspace

As users work independently of each other, the contents of their workspaces increasingly diverge. Typically, some of the differences between workspaces are inconsistencies. For example, changes that John makes in a report-library routine might cause errors in the report program that Mary's writing. To minimize the time and effort required to resolve inconsistencies during the "integration" phase of a project, it makes sense to have users synchronize their workspaces on a regular basis.

With AccuRev, synchronization does not mean incorporating data into your workspace directly from one or more other workspaces. Instead, synchronization involves copying data into the workspace from the stream to which all team members **Promote** their changes. This operation is performed by the **Update** command. This figure shows the **Update** toolbar button. You can also invoke this command as **File > Update** from the main menu.



> NOTE: The stream's role as a provider of data — through **Updates** — to a set of workspaces motivates the term backing stream. Think of restocking a store's shelves with merchandise retrieved from "the back room".

So an update operation on your workspace copies versions of certain files from the backing stream to the workspace, overwriting/replacing the files currently in the workspace. But which files? **Update** changes a file if (1) there is a newer version in the backing stream, and (2) the file is *not* currently active in your workspace.

Update won't overwrite an active file, even if there's a new version of it in the backing stream. No matter how good someone else's code is, you don't want his changes to wipe out the changes that you've been making! This situation is another instance of an overlap, which we described in the **Promote** section above. (You can encounter an overlap both (1) if you're trying to make your private changes public (promotion), or (2) if you're trying to bring already-public changes into your private workspace (updating).) In all such situations, AccuRev resolves the overlap situation with a merge operation (details below).

Update handles namespace changes as well as content changes. Thus, if your colleague renamed a file and promoted the change, an update will cause the file to be renamed in your workspace. And if your colleague removed a file (**Defunct** command), an update will cause the file to disappear from your workspace.

The Fine Print

Here's how AccuRev prevents an update from "clobbering" your changes. The first thing **Update** does is to analyze your workspace, determining whether each version-controlled file is "active" or "inactive". Initially, all the files in a workspace are inactive — each one is a copy of some version in the repository. (For each version-controlled file, AccuRev keeps track of *which* particular version.)

A file is deemed to be active in your workspace if you've created a new version of it, using the **Keep**, **Rename**/**Move**, or **Defunct** command. (A couple of additional commands "activate" a file; one of them is discussed below.) When **Update** copies versions from the repository into your workspace, it skips over all such active files.

> Note: **Update** can tell if you've modified a file but have not yet stored the changes in the repository as a new **Keep** version. It uses file sizes, timestamps, and checksums to determine this. The presence of any such files prevents the update from proceeding. You can use the **Anchor** command to activate such files, enabling **Update** to do its work.

Merge: When Changes Would Collide ...

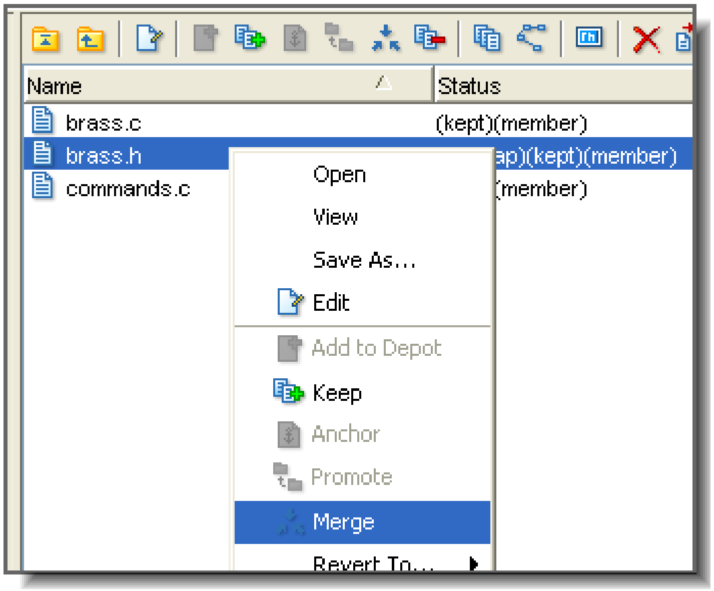
The preceding sections, on the **Promote** and **Update** commands, both discuss the situation in which two users concurrently work on the same file. Their changes to the file are said to overlap. Both **Promote** and **Update** decline to process a file with overlap status, because doing so would cause one user's changes to overwrite the other's changes.

For example:

- Team members John and Mary both **Keep** one or more new private versions of **brass.c** in their respective workspaces.
- Mary **Promote**s her latest new version of **brass.h** to the backing stream.
- At this point, AccuRev will decline to do either of the following:
  - **Promote** John's version of **brass.h** to the backing stream.
  - Overwrite John's copy of **brass.h** during an update. (The **Update** command skips over this file, but continues its work on other files.)

To enable either a promotion or an update of **brass.h**, John must incorporate, or merge, the version in the backing stream — which contains Mary's changes — into his own copy of the file. The **Merge** command is essentially a fancy text editor, which combines the contents of two versions of a text file. The resulting "merged version" replaces the file in John's workspace.

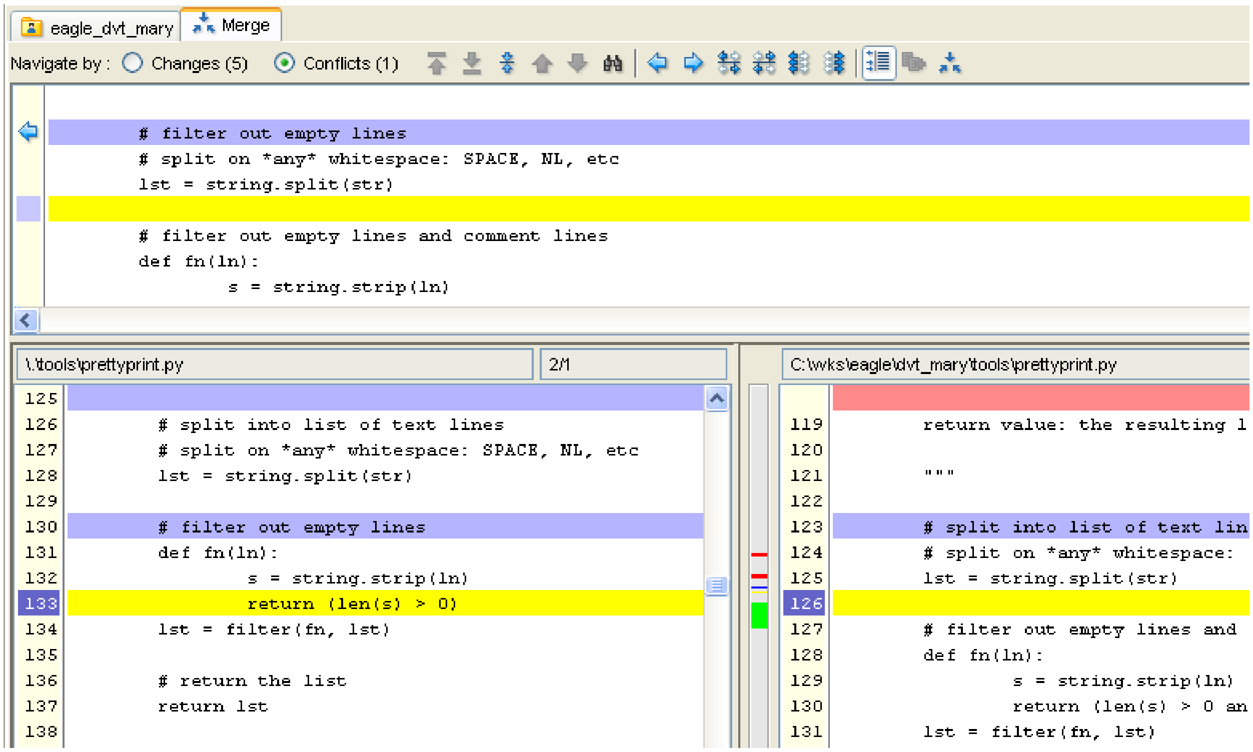
This figure shows how to invoke the **Merge** command from a file's context (right-click) menu in the File Browser. You can also invoke **Merge** with the toolbar button.



Often, a merge operation is unambiguous, and so can be performed automatically. For example, suppose Mary's changes to file **brass.h** all occur in lines 1–50, and all of John's changes occur in lines 125–140. In this case, merging the two versions involves replacing some or all of John's first 50 lines with Mary's. Now, the edited version of **brass.h** in John's workspace contains both users' changes.

> Note: we don't claim that the two sets of changes are semantically consistent with each other. That's what the build-and-test cycle is for!

If both John and Mary have made changes to the same part of the file — say, lines 85–87 — then John must decide how to resolve this conflict. The graphical **Merge** tool makes this easy:



```
        # filter out empty lines
        # split on *any* whitespace: SPACE, NL, etc
        lst = string.split(str)

        # filter out empty lines and comment lines
        def fn(ln):
                s = string.strip(ln)
```

\.\tools\prettyprint.py 2/1

```

        # split into list of text lines
        # split on *any* whitespace: SPACE, NL, etc
        lst = string.split(str)

        # filter out empty lines
        def fn(ln):
                s = string.strip(ln)
                return (len(s) > 0)
        lst = filter(fn, lst)

        # return the list
        return lst
```

C:\wks\eagle\dvt_mary\tools\prettyprint.py

```

        return value: the resulting l

        """

        # split into list of text lin
        # split on *any* whitespace:
        lst = string.split(str)

        # filter out empty lines and
        def fn(ln):
                s = string.strip(ln)
                return (len(s) > 0 an
        lst = filter(fn, lst)
```

After performing a merge, AccuRev automatically **Keep**s the merged version to preserve the results of the merge operation. You can then **Promote** the merged version to the backing stream. After that, other team members can use **Update** — or perhaps **Merge** — to bring all the changes into their workspaces.

The Fine Print

The graphical Merge tool performs a “3-way merge”, which uses the common ancestor of the two versions being merged. This algorithm helps to automate the merge operation, often completely eliminating the need for human intervention.

AccuRev performs merge operations on text files only, not on binary files.

AccuRev keeps track of all merge operations. This greatly simplifies subsequent merge operations on files that have been merged previously: you don’t need to resolve the same conflicts over and over again.

The most common overlap situation involves AccuRev’s preventing you from promoting a file, because someone else “got there first” in creating a version in the backing stream. AccuRev can also detect deep overlaps, in which another user “got there first” in creating a version in the *parent* of the backing stream, or in other higher-level streams.

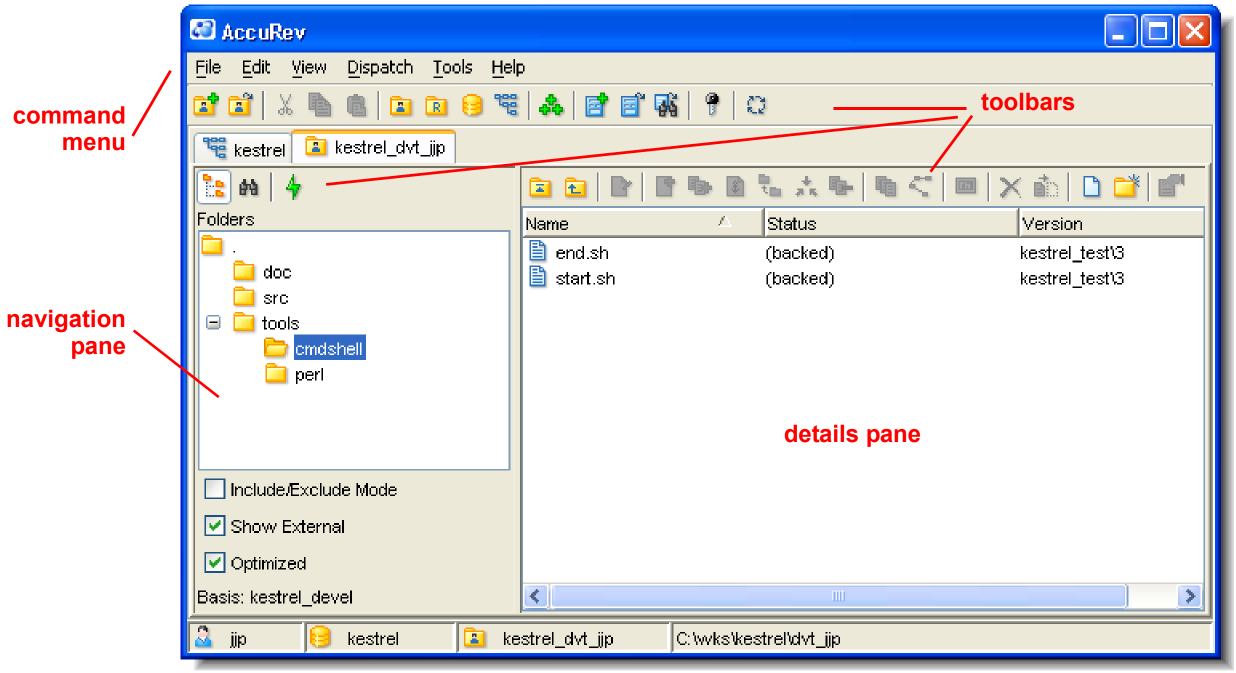
Learning More about AccuRev

Armed with the four commands **Keep**, **Promote**, **Update**, and **Merge**, you’ll be able to work effectively in team parallel development environment. To make full use of AccuRev’s configuration management capabilities, you’ll need to dig a bit deeper. But no matter what your CM challenges are, we think you’ll find that AccuRev meets them with an architecture and user interface that are intuitive and easy to learn.

The File Browser

AccuRev's job is to keep track of your files. Accordingly, one of AccuRev's main GUI tools is the File Browser. The File Browser makes it easy to view, monitor the status, and change the contents of files located in AccuRev workspaces. You can have any number of File Browser tabs open concurrently in an AccuRev GUI window — each one displaying the contents of a difference workspace.

The File Browser display resembles that of Windows Explorer. It contains a navigation pane and a details pane. You can execute AccuRev commands by selecting from the pull-down command menu or by clicking toolbar buttons.



Name	Status	Version
end.sh	(backed)	kestrel_test\3
start.sh	(backed)	kestrel_test\3

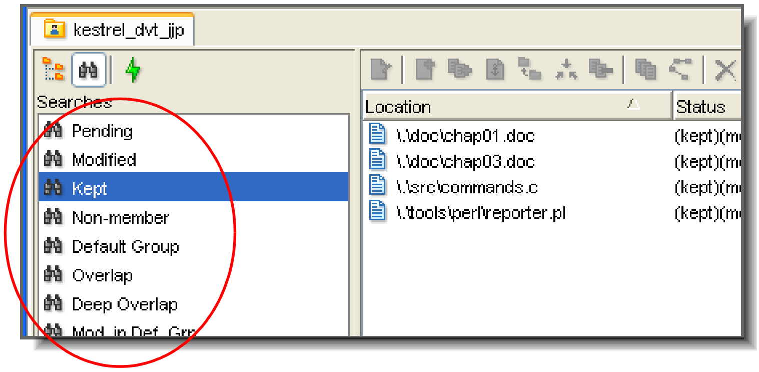
☐ Include/Exclude Mode

☑ Show External

☑ Optimized

Basis: kestrel_devel

jjp | kestrel | kestrel_dvt_jjp | C:\wks\kestrel\dvt_jjp



Location	Status
\.\doc\chap01.doc	(kept)(m
\.\doc\chap03.doc	(kept)(m
\.\src\commands.c	(kept)(m
\.\tools\perl\reporter.pl	(kept)(m

As shown above, the navigation pane (usually) shows the directory hierarchy of one workspace; the details pane (usually) shows the contents of one of the workspace's directories. We say "usually", because the File Browser has a powerful searches view. The details pane can locate all the files, throughout the entire workspace, that meet a particular search criterion. For example, it can display all the files you've edited and saved with the **Keep** command. (In the AccuRev CLI, this facility is described using the term "filter".)

When a workspace is opened in the GUI, the File Browser automatically enters Searches View and performs a Pending search. For more on Searches View, see *Searches and File Statuses* on page 53.

Each of the File Browser's panes has its own toolbar, separate from the GUI window's overall toolbar.

Alternatives to the File Browser

The File Browser is not intended to completely replace the Windows Explorer or comparable tools on Unix systems. For example, the File Browser's details pane does not include such columns as Size, Date Modified, and Attribute. These are operating-system-level file properties; the File Browser reports only a few AccuRev-specific properties.

Some people do most of their work inside an integrated development environment (IDE), such as Visual Studio or Forte for Java. These environments have their own "explorer" or "file browser" built in, typically organizing files into separate "projects". You can use AccuRev commands within an IDE if an IDE integration for that environment is available from AccuRev, Inc. As of AccuRev Version 3.5, the following IDE integrations are available:

- Visual Basic 6.0
- Visual C++ 6.0
- Visual Studio .NET
- Rational Rose
- additional IDEs that support the SCC interface
- Sun One Studio and NetBeans
- Eclipse 2.x, 3.0

Even if you use an IDE most of the time, you may want to use the File Browser occasionally:

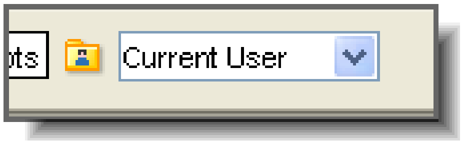
- The IDE does not display all of a file's configuration-management properties, such as the current version-ID.
- Certain AccuRev commands cannot be executed through the IDE integration, only from the AccuRev File Browser.

It's important to keep in mind that you can use a wide variety of tools to make content changes to a file, but you must use AccuRev commands to make valid namespace changes: renaming, moving to another directory, or deleting. Some of the IDE integrations support automatic renaming

Opening a File Browser Tab

There are several ways to open a File Browser tab in the AccuRev GUI. When you start a new GUI session, a File Browser tab opens automatically on the workspace you visited most recently in the previous session. To open a File Browser on any of your workspaces, use one of these techniques:

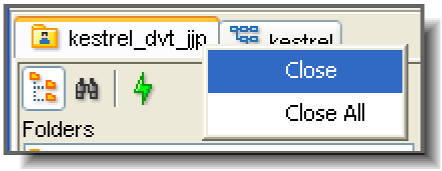
- Select **File > Open > Workspace** from the command menu, and select a workspace.
- Select **View > Workspaces** from the command menu (or click the **View Workspaces** toolbar button) to open a **Workspaces** tab. Right-click the desired workspace, and select **Open** from the context menu. Or just double-click the desired workspace.
- On a StreamBrowser tab, make sure the desired workspace is displayed. (To open another user's workspace, change the listbox setting at the bottom on the tab from **Current User** to the appropriate username. There is also an **All Workspaces** setting.) Right-click the desired workspace, and select **Open** from the context menu. Or just double-click the desired workspace.



As of Version 3.5.5, the File Browser tab always opens in Searches View. See *Searches and File Statuses* on page 53.

Leaving a File Browser Tab

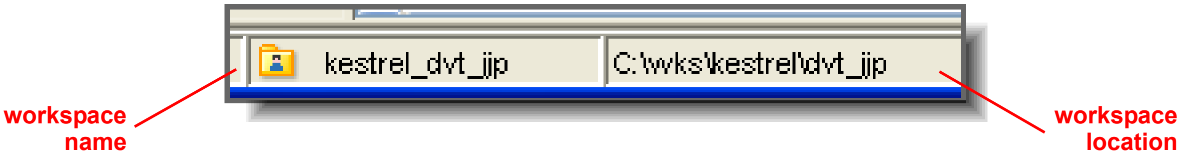
You can leave a File Browser, switching to another tab in the AccuRev GUI, then return to that File Browser tab later. Alternatively, you can close the tab altogether: right-click the title, then select **Close** from the context menu.



When returning to a File Browser tab, it's a good idea to refresh the display (**View > Refresh** or function key **F5**). This ensures that the File Browser display reflects any work you performed "between visits" to the tab.

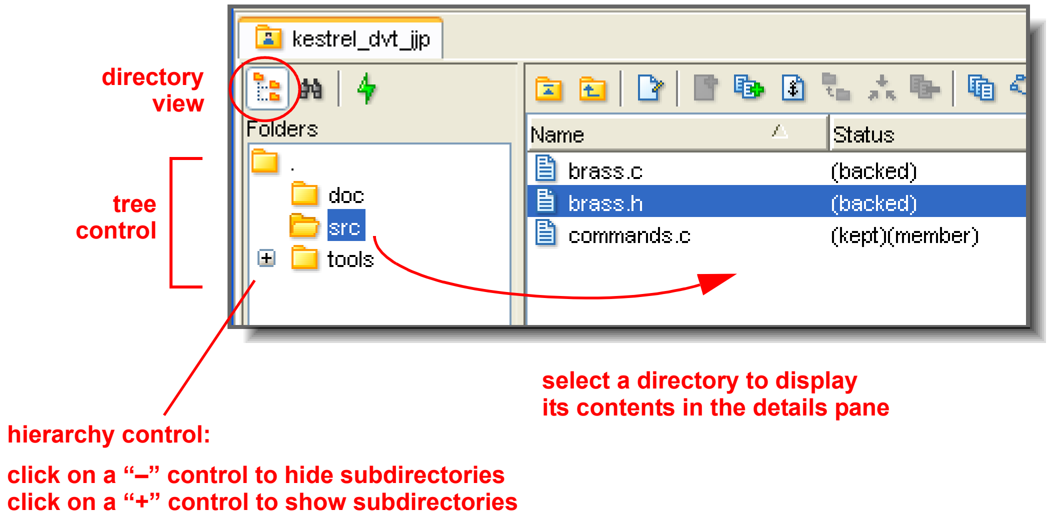
File Browser Layout

If you're familiar with Windows Explorer or similar tools, you'll find that using the File Browser is easy. As with those tools, there's a navigation pane and a details pane. Below these panes are overall indicators for the AccuRev GUI window. When you're working in the File Browser, indicators show the name and location of the current workspace.



Navigation Pane

You use the navigation pane to control which files appear in the details pane. The directory view provides for "traditional" navigation: when you select a particular directory (or "folder") using the standard hierarchy control, the files in that directory appear in the details pane.



Name	Status
brass.c	(backed)
brass.h	(backed)
commands.c	(kept)(member)

directory view

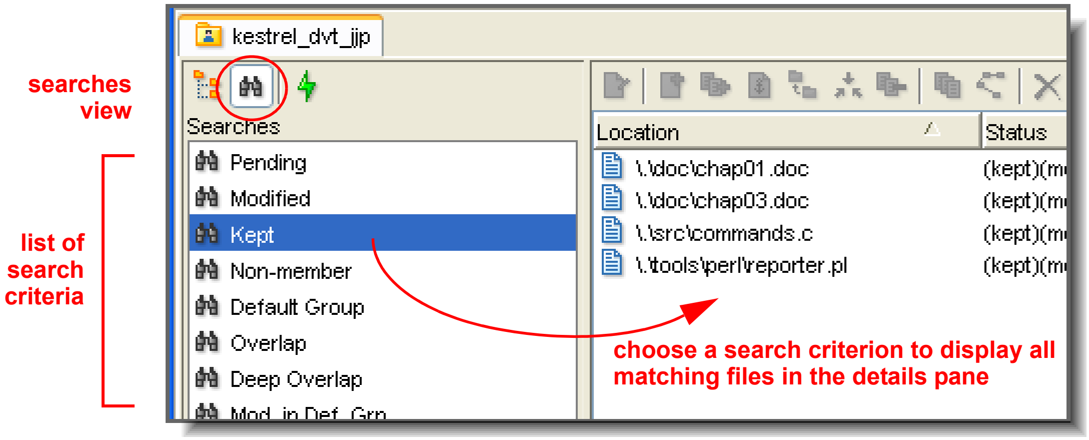
tree control

select a directory to display its contents in the details pane

hierarchy control:

click on a "–" control to hide subdirectories
click on a "+" control to show subdirectories

The File Browser also has a searches view, which organizes the workspace's files by AccuRev status, instead of by directory location. When you switch to searches view and choose a particular search criterion, all the files that the search locates (in the entire workspace, regardless of directory location) appear in the details pane:



Location	Status
\.\doc\chap01.doc	(kept)(m
\.\doc\chap03.doc	(kept)(m
\.\src\commands.c	(kept)(m
\.\tools\perl\reporter.pl	(kept)(m

searches view

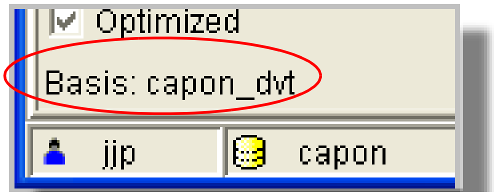
list of search criteria

choose a search criterion to display all matching files in the details pane

The searches view enables you to see "just the files you care about", instead of all the files in your workspace. Which files do you care about? In general, a workspace contains a copy of the entire source base, which might include hundreds, or perhaps thousands, of files. But for a given development project, you'll probably modify only a handful of the files. You may not need the other files at all; or you may need them for general reference, or to enable you to perform software builds and tests in the workspace.

> **Note**: As of Version 3.5, there's another way to concentrate on just the files you care about: the Include/Exclude facility. See *Working in Include/Exclude Mode* on page 61.

Regardless of which view you're using, an indicator at the bottom of the navigation pane shows the name of the workspace's backing stream.



☑ Optimized

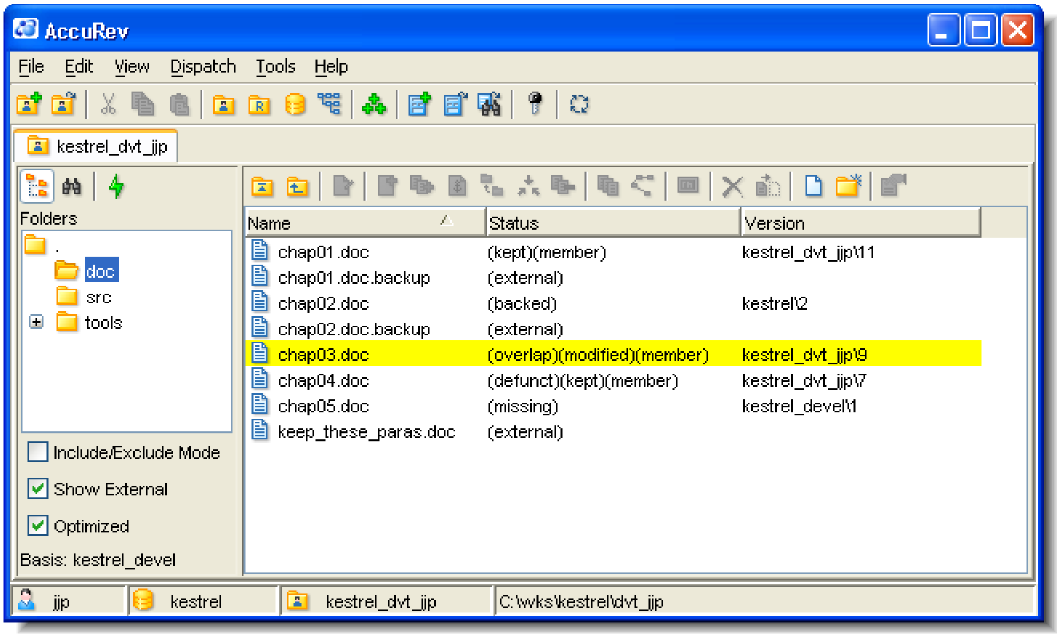
Basis: capon_dvt

jjp | capon

The following section discusses the details pane and the various AccuRev file-status indicators. In section *Searches and File Statuses* on page 53, we discuss the various search criteria that you can choose in the searches view.

Details Pane

The details pane of the File Browser displays three columns of information for the files and directories in a workspace:



Name	Status	Version
chap01.doc	(kept)(member)	kestrel_dvt_jjp\11
chap01.doc.backup	(external)	
chap02.doc	(backed)	kestrel\2
chap02.doc.backup	(external)	
chap03.doc	(overlap)(modified)(member)	kestrel_dvt_jjp\9
chap04.doc	(defunct)(kept)(member)	kestrel_dvt_jjp\7
chap05.doc	(missing)	kestrel_devel\1
keep_these_paras.doc	(external)	

☐ Include/Exclude Mode

☑ Show External

☑ Optimized

Basis: kestrel_devel

jjp | kestrel | kestrel_dvt_jjp | C:\wks\kestrel\dvt_jjp

Name

The name of the object, either a file or a subdirectory, in this workspace. AccuRev allows you to rename and relocate objects, so a file might have a different name and/or a different directory location in another workspace.

Status

One or more keywords, indicating the AccuRev status of the file (or subdirectory) in this workspace. Here's a list of all the status keywords; section *Working in the Details Pane* on page 36 describes how the status of an element changes during typical development scenarios.

Presence of the element in the workspace:

- **(defunct)** — the element has been marked for removal from the workspace stream with the **Defunct** command. (The element has already been removed from the workspace directory tree (local disk storage); it gets removed from the workspace stream (in the depot) when you **Promote** the element to the backing stream.)

- **(missing)** — the workspace "should" include a version of this element, but doesn't. This occurs when you delete version-controlled files from the workspace using operating system commands, or using the AccuRev **Delete** command.

  In a sparse workspace, this status also applies to elements that are not currently loaded into the workspace. As of Version 3.5, the ability to create a new sparse workspace has been disabled; use the Include/Exclude facility in a new workspace instead (see page 61).
- **(excluded)** — the element does not appear in the workspace because it has been excluded, using the Include/Exclude facility (new in Version 3.5). For file elements, it's more likely that the exclusion was explicitly set on the directory in which the file resides, or in a higher-level directory that includes the file. See *Working in Include/Exclude Mode* on page 61.
- **(external)** — the file or directory has not been placed under version control. (It's in the workspace directory tree, but not in the workspace stream.) See *Controlling the Display of External Objects* on page 35.

Changes to the element in the stream:

- **(modified)** — the file has been modified in the workspace since the most recent **Update** or **Keep**.
- **(kept)** — a new version of the element has been created with **Keep** or **Rename**, and the file has not subsequently been modified or promoted to the backing stream.
- **(member)** — the element is "active" in the workspace (is in the workspace stream's default group).

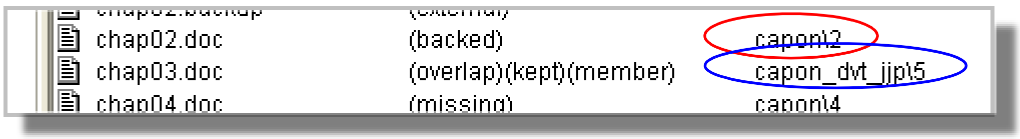
Relationship to the backing stream:

- **(backed)** — the element has not been changed since the last time you **Promote**d it, or since the most recent **Update** of your workspace.
- **(stale)** — the element has changed in the backing stream, but the change has not been incorporated into the workspace with an **Update**.
- **(overlap)** — the element has changed both in the backing stream and in your workspace. This indicates that a merge is required before you can promote your changes to the backing stream. Files with this status appear with a yellow highlight.

Version

The version-ID, indicating which version of an AccuRev element is in this workspace. For elements that you have modified in this workspace, but not yet promoted to the backing stream, the version-ID indicates a version in the workspace's built-in stream. These are described as "active elements" or in AccuRev parlance, as "members of the workspace's default group". For inactive elements, which you haven't modified, the Version column indicates a version from some higher-level stream, which your workspace stream inherits.

“inactive” element:
version inherited from higher-level stream



```
chap02.backup    (external)
chap02.doc       (backed)                   capon\2
chap03.doc       (overlap)(kept)(member)    capon_dvt_jjp\5
chap04.doc       (missing)                  capon\4
```

“active” element:
version in workspace stream

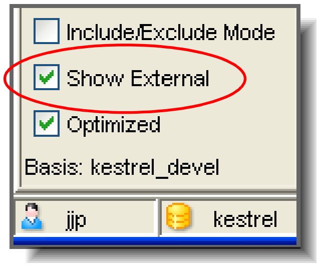
If this column is blank, the file or subdirectory has not been placed under version control. Such objects have the status **(external)**.

Controlling the Display of External Objects

By default, the details pane shows both elements (files and subdirectories that have been placed under version control) and external objects. You can control whether external objects are shown with the **Show External** checkbox at the bottom of the navigation pane.

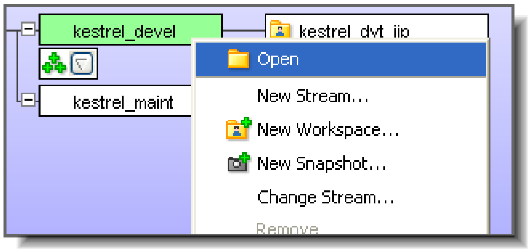
(For an explanation of the **Optimized** checkbox, see *Search Performance Optimizations* on page 59.)

controls display of external objects



□ Include/Exclude Mode
☑ Show External
☑ Optimized
Basis: kestrel_devel

Displaying the Contents of a Dynamic Stream or Snapshot



So far, we’ve discussed how the File Browser displays the contents of a workspace. But the File Browser can also display the contents of a dynamic stream or a snapshot: in the StreamBrowser, right-click the dynamic stream or snapshot, and select **Open** from the context menu. (Or just double-click it.)

Dynamic streams and snapshots are different from workspaces. They exist only in the AccuRev repository, not on the disk storage of any client machine. And they don’t contain actual files, but only the element versions that have been created in the repository by the **Promote** command.

The versions that the File Browser displays as the contents of a dynamic stream or snapshot can have only these few statuses:

- **(member)** — the element is “active” in the stream (is in the stream’s default group). This version has not yet been promoted to the stream’s parent.
- **(overlap)** — the element is active in this stream, and the parent stream’s version has changes that are not included in this stream’s version. A merge is required before this stream’s version can be promoted to the parent stream.
- **(defunct)** — the element is active in this stream: it was removed from a workspace with the **Defunct** command, and the change was then promoted to this stream.

- **(backed)** — the element is not active in this stream. This stream “inherits” the version of the element that appears in the parent stream.

The statuses that relate to an actual file in a user’s workspace are not relevant in a dynamic stream or snapshot.

Working in the Details Pane

Working with a workspace’s files in the File Browser’s details pane follows this pattern:

1. Select one or more files and/or directories.
2. Invoke an AccuRev command, either:
   - ... by clicking one of the File Browser’s toolbar buttons, or
   - ... by right-clicking a selected object and choosing a command from the context menu.

The file-selection gestures are the same ones used by Windows Explorer:

- To select a file, click it with the left mouse button.
- To select a contiguous range of files, click-and-drag with the left mouse button.
- To select all the files in a directory, press **Ctrl-A**.
- To add or subtract a file from an existing selection, hold down the **Ctrl** key and click that file with the left mouse button.
- To extend an existing selection to a particular file, hold down the **Shift** key and click that file with the left mouse button.
- Typing a character selects the next file or directory whose name begins with that character.

The File Browser’s toolbar — and a file’s right-click context menu — include the following AccuRev commands. (Note: you invoke one of the most important commands, **Update**, from the navigation pane, not the details pane. See *Updating the Workspace* on page 68.)

Go to root

Have the details pane display the contents of the depot’s (and workspace’s) top-level directory.

Up one level

Have the details pane display the contents of the current directory’s parent.

Edit

Open a text editor on the currently selected file. (Select exactly one file before invoking this command.) You can use environment variable AC_EDITOR_GUI or EDITOR to control which text editor gets invoked.

The File Browser automatically updates the file’s status indicators after an edit session. If you have changed the file’s contents, AccuRev automatically places a **(modified)** status indicator in the **Status** column.

Note: you can also change the contents of files using operating system commands, scripts, and third-party tools, including integrated development environments (IDEs). In this case, invoke the command **View > Refresh** or press function key **F5** to update the file’s status indicators.

Add to Depot

Place the selected files/directories under version control. All the selected objects must currently have **(external)** status. For each one, AccuRev creates version 1 in the workspace stream.

Keep

Create a new version in the workspace stream for each selected file. All of the selected objects must be file elements; none can have **(external)** status and none can be a directory.

Typically, you invoke the **Keep** command on files that you’ve been working on, and thus have **(modified)** status. But this is not a requirement. If you keep a file that you have not modified, a new version is created with identical contents.

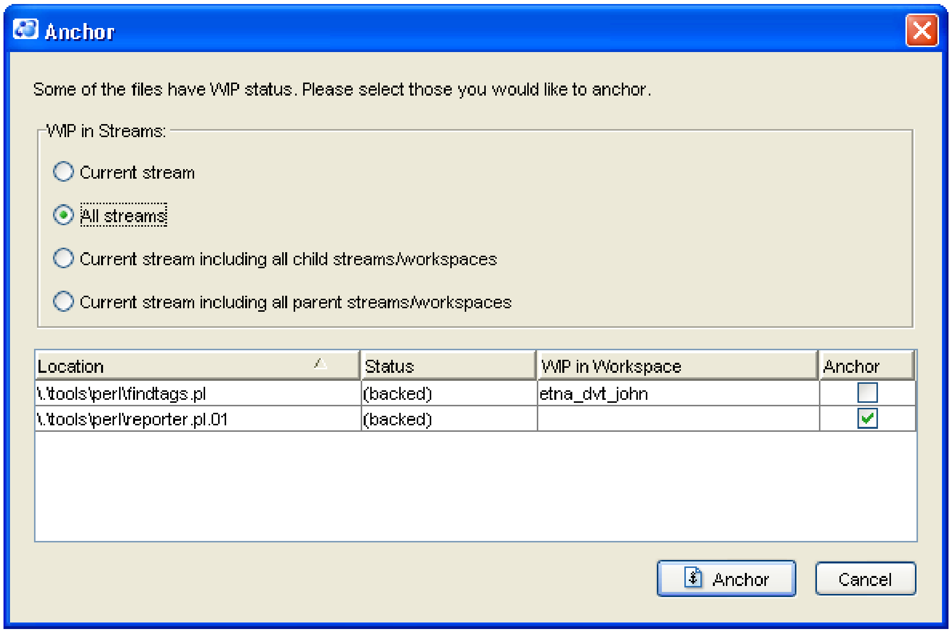
The first time you use the **Keep** command on a file, the files becomes active in the workspace (“is placed in the workspace’s default group”). After that, you can modify and **Keep** a file as many times as you like. The file remains active in the workspace until you invoke the **Promote** or **Revert to Backed** command on it.

Anchor

Make the selected files/directories active in the workspace (“place it in the workspace’s default group”), without modifying them. Typically, you anchor a file in your workspace to prevent it from being overwritten with a newer version by a subsequent **Update** command. (**Update** overwrites inactive files only, not active ones.)

File(s) that you **Anchor** must currently be inactive in the workspace, from AccuRev’s perspective; that is, they must have **(backed)** status. **Anchor** creates a new version in the workspace stream, identical in contents to the existing version. This new version simply records the fact that the file is officially active in the workspace.

AccuRev features anchor-required workspaces, in which you *must* **Anchor** a file before modifying it. When you anchor a set of elements in an anchor-required workspace, a dialog box appears if one or more of them are active in a sibling workspace:



Anchor

Some of the files have WIP status. Please select those you would like to anchor.

WIP in Streams:

○ Current stream

● All streams

○ Current stream including all child streams/workspaces

○ Current stream including all parent streams/workspaces

Location	Status	WIP in Workspace	Anchor
\.\tools\perl\findtags.pl	(backed)	etna_dvt_john	□
\.\tools\perl\reporter.pl.01	(backed)		☑

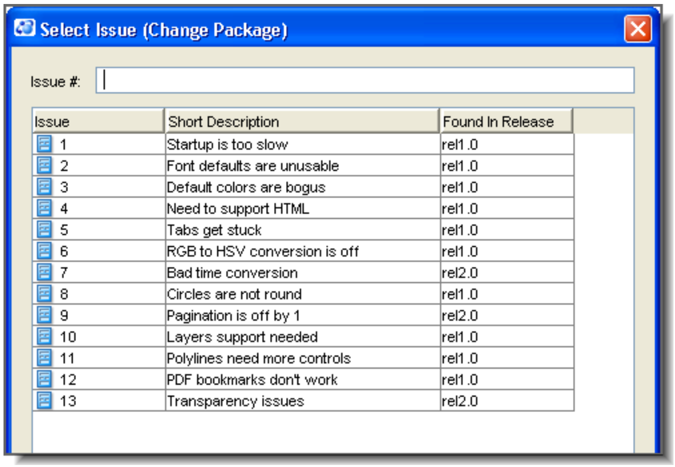
This enables you to select/deselect individual elements to be anchored. Note that the elements that are active in the sibling workspace are initially deselected. This makes it easy to avoid the situation where multiple users are working on the same file(s) concurrently.

Promote

Send the active version of the selected files/directories to the workspace's backing stream (parent stream). The new version in the backing stream acts as a reference to the version created in your workspace. (In AccuRev parlance, it's a "virtual version" that "is an alias for" the version in your workspace.)

The selected element(s) become inactive in the workspace (are removed from the workspace's default group). The status of the elements becomes **(backed)**.

Integration between configuration management and issue management: If either or both of the integrations between AccuRev's configuration management and issue management facilities is enabled, a particular Dispatch query is executed and its results displayed. You can select one of the issue records and click **Ok**; or you can type a number in the **Issue #** input field. The integration(s) record information about the **Promote** transaction in the issue record you've designated.



Select Issue (Change Package)

Issue #:

Issue	Short Description	Found In Release
1	Startup is too slow	rel1.0
2	Font defaults are unusable	rel1.0
3	Default colors are bogus	rel1.0
4	Need to support HTML	rel1.0
5	Tabs get stuck	rel1.0
6	RGB to HSV conversion is off	rel1.0
7	Bad time conversion	rel2.0
8	Circles are not round	rel1.0
9	Pagination is off by 1	rel2.0
10	Layers support needed	rel1.0
11	Polylines need more controls	rel1.0
12	PDF bookmarks don't work	rel1.0
13	Transparency issues	rel2.0

Clicking **Ok** without specifying an issue record bypasses the integration.

For more information, see *Integrations Between Configuration Management and Issue Management* on page 45 of the *Dispatch Issue Management Manual*.

Merge

For each selected file, perform a merge operation, involving two versions: the one in your workspace and the one in the workspace's backing stream (parent stream). The resulting "merged version" is saved as a new version in the workspace stream.

The merge operation takes into account both the contents of the versions to be merged and the names of those versions. Thus, at the completion of a **Merge** command, a file might have a different name!

Revert to Backed

For each selected file/directory, remove all changes you've made since the last time you **Promote**d it, or since the last **Update** of the workspace — whichever is more recent. For files, this includes both content changes and namespace changes. For directories, this includes namespace changes only.

The selected elements become inactive in the workspace (are removed from the workspace's default group). The status of the elements becomes **(backed)**.

When you invoke this command in a dynamic stream, the depot's **pre-promote-trig** trigger fires. That's because, like a **Promote**, the command changes which version of the element appears in the stream.

Revert to Most Recent Version

> Note: if you've modified a file but not yet performed a **Keep** on it, this command works like **Revert to Backed**.

(on context menu only, not toolbar) For each file element with **(modified)** status, replace the file with the most recent version you created with **Keep**. Use this command when you've saved one or more intermediate versions of the file(s), and you want to discard further changes you've made since a **Keep**.

The selected elements remain active in the workspace (are *not* removed from the workspace's default group). The **(modified)** status of the elements changes to **(kept)**.

Show History

Open a History Browser tab, containing the transactions involving the selected file or directory.

Browse Versions

Open a Version Browser tab, showing all the versions of the selected file or directory, and their interrelationships (ancestry).

Cut

Mark the currently selected element to be moved to another directory in the same depot. To finish the relocation, right-click the destination directory (in either the navigation or details pane) and select **Paste**.

Paste

Specify the destination for an element that has been marked for relocation with the **Cut** command.

Rename

Change the name of the selected file or directory. The new name can be in a different directory within the depot. Thus, this command can perform a “move” as well as a “rename”. This activates the object (includes it in the workspace’s default group). Its status changes to **(kept)(member)**.

Note: if you change the contents of a file, then rename it without first performing a **Keep**, the version created by the **Rename** command does not contain the content changes, just the name change. The file’s new status reflects this: **(modified)(member)**.

Delete

Remove the selected files/directories from the workspace directory tree — the workspace’s local disk storage. This command does not affect the depot — AccuRev still thinks the deleted objects should be there, so it continues to list them, with **(missing)** status.

To remove a file or directory from the depot’s stream hierarchy, so that its removal will be reflected in other users’ workspaces (after you **Promote** the change and they perform an **Update**), use the **Defunct** command instead of **Delete**.

Defunct

Remove the selected files/directories from the workspace directory tree (local disk storage), and also change their status to **(defunct)**. This also activates the elements in the workspace stream — they’re “in the act of being deleted”. Accordingly, the elements also get **(kept)(member)** status.

When you **Promote** a defunct element, it disappears entirely from the workspace stream, and from the File Browser display. The element becomes **(defunct)**, and also active, in the backing stream.

When you **Update** a workspace below a stream with a **(defunct)** file, the file is removed from the workspace’s local disk storage. For a **(defunct)** directory, **Update** removes the entire subtree below that directory from the workspace’s local disk storage.

Populate

Run the **populate** command on the currently selected element. A dialog box lets you enable the overwrite and/or recursive option.

New File

Create an empty file in the current directory, and optionally place it under version control (**Add to Depot** checkbox).

New Folder

Create an empty subdirectory in the current directory, and optionally place it under version control (**Add to Depot** checkbox).

Send to Issue

Record the selected versions in the change package section (Changes tab) of a particular issue record.

Send to Change Palette

(dynamic stream only): Load the selected versions into the Change Palette, so that they can be promoted to another stream.

Properties

Displays information about the selected element: pathname, file type of current version, and element-ID.

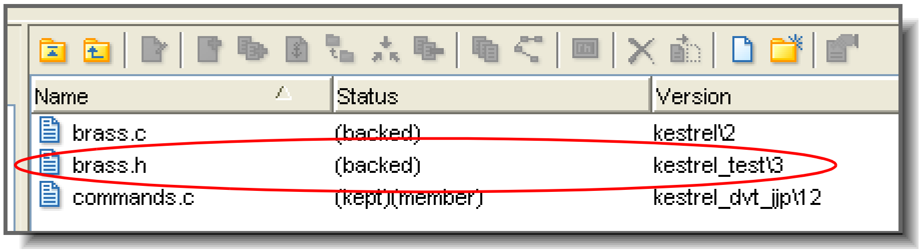
Each element displayed in the details pane also has a context menu, which offers a similar, but not identical, set of commands as the toolbar. See *File Browser: Element's Context Menu in Details Pane* on page 146.

Changes in File Status / Creation of New Versions

As a version control system, AccuRev keeps track of changes that users make to files and directories. The following sections describe common usage scenarios, showing how AccuRev change-tracking is reflected in the File Browser's details pane: in changes to the status of objects, and in creation of new versions of objects.

Editing a File's Contents

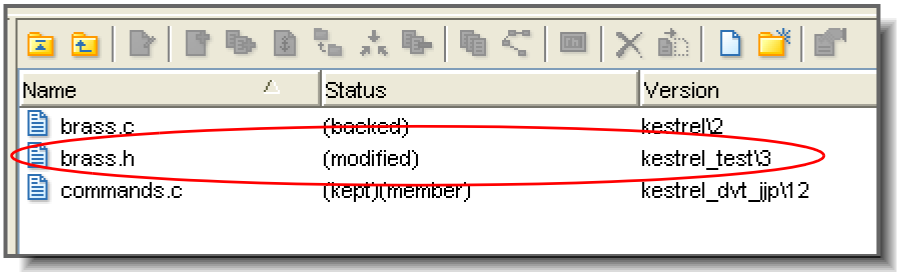
Perhaps the most common scenario is "non-conflicting development": modifying a file that no one else is working on concurrently. Initially, the Status column in the details pane shows the file's status as **(backed)**: your workspace contains an unmodified copy of a version in the backing stream. The Version column contains a version-ID, indicating which backing-stream version it is.



Name	Status	Version
brass.c	(backed)	kestrel\2
brass.h	(backed)	kestrel_test\3
commands.c	(kept)(member)	kestrel_dvt_jjp\12

Invoke the **Edit** command on the file, using the toolbar button or the file's context menu. This launches a text editor session on the file. You can use environment variable AC_EDITOR_GUI or EDITOR to control which text editor is launched.

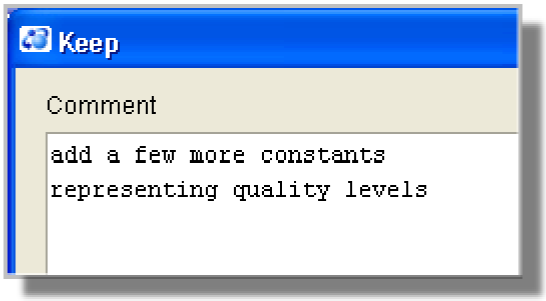
When you end the edit session, be sure to refresh the File Browser display, using function key **F5** or the **Refresh** button on the GUI's main toolbar or the **View > Refresh** command on the GUI's main menu. The file's status changes to **(modified)**. This indicates that you've modified the file in your workspace since the last time you synchronized the file in your workspace with the depot. (That is, you've changed the file at the operating-system level, but not at the AccuRev level.)



Name	Status	Version
brass.c	(backed)	kestrel\2
brass.h	(modified)	kestrel_test\3
commands.c	(kept)(member)	kestrel_dvt_jjp\12

You can edit the file (and update the File Browser display) as many times as you wish. The file's status remains **(modified)**.

To have AccuRev record a new version of the file, invoke the **Keep** command on it. If you wish, enter a comment in the dialog box that appears. The comment string becomes a permanent annotation to the version, viewable with the History Browser.

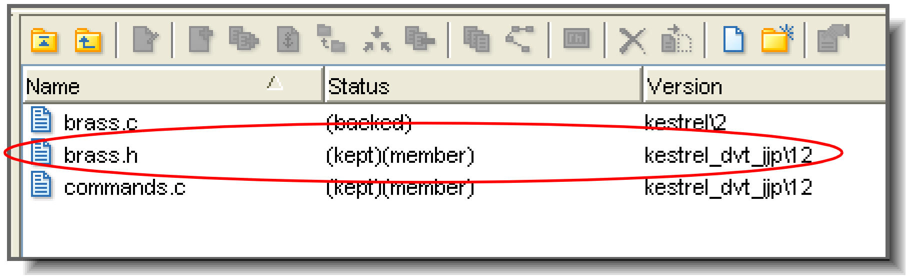


Comment

```
add a few more constants
representing quality levels
```

The **Keep** command creates a new version of the file in the workspace stream. This version is said to record a content change to the file. AccuRev also tracks namespace changes — see below.

The file's status changes to **(kept)(member)**; this indicates that you've recorded a new version, and that the file is currently active in your workspace (is a member of the workspace's default group).



Name	Status	Version
brass.c	(backed)	kestrel\2
brass.h	(kept)(member)	kestrel_dvt_jjp\12
commands.c	(kept)(member)	kestrel_dvt_jjp\12

The Version column shows the version-ID of the newly created version. Note that this version is recorded in your private workspace stream (in this example, **kestrel_dvt_jjp**); previously the Version column indicated that your workspace contained a version from the public backing stream (**kestrel_dvt**).

You can continue modifying the file with the **Edit** command, and saving new versions in the depot with the **Keep** command. The file's status will alternate between **(modified)(member)** and **(kept)(member)**. The persistence of the **(member)** indicator reflects the fact that the file remains active in your workspace until you promote your changes to the backing stream or undo your changes. These operations are described below.

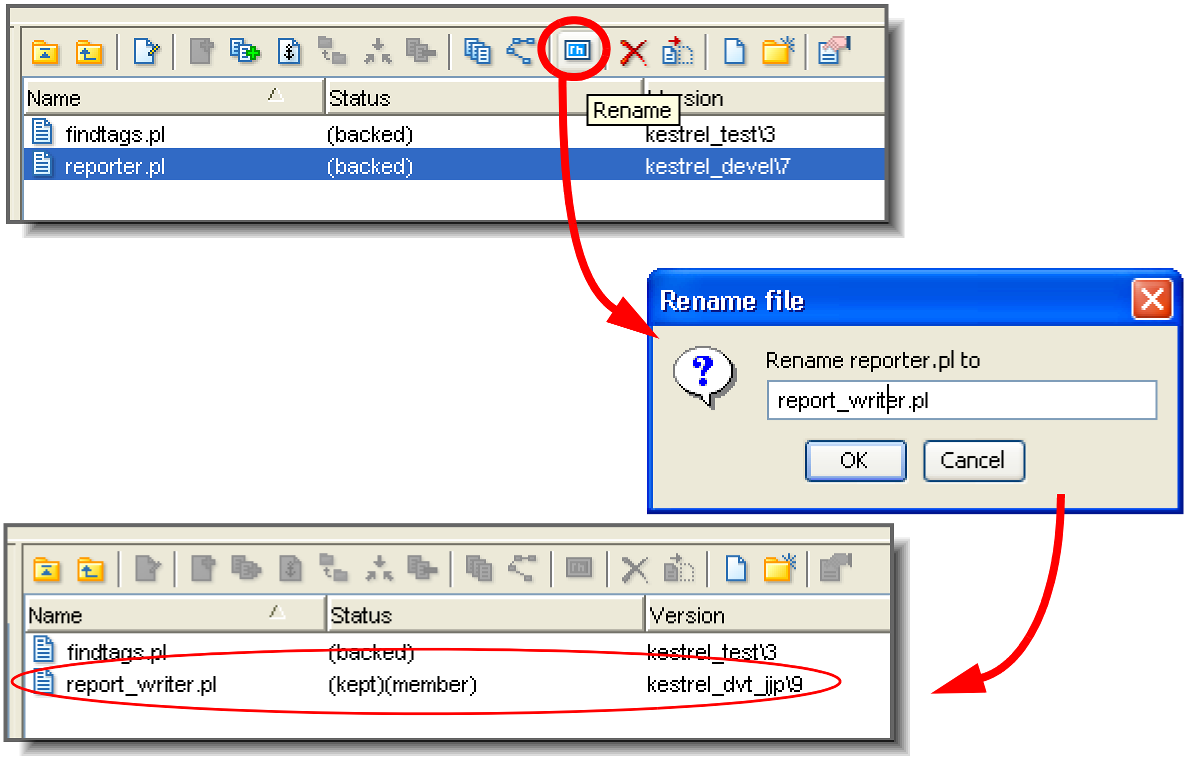
Renaming or Moving a File

In addition to tracking changes to the contents of files, AccuRev tracks namespace changes:

- To rename a file within the same directory, invoke the **Rename** command.
- To move a file to a different directory in the depot, right-click the file and select **Cut** from the context menu. Then right-click the destination directory in the navigation pane (not the details pane) and select **Paste** from the context menu.

  You can also use the **Rename** command to move a file to a different directory; specify a relative pathname (such as **..\otherdir\myfile.c**) or a depot-relative pathname (such as **/./lib/header/base.h**) as the new name for the file.

AccuRev records a namespace change to a file in the same way it records a content change: by creating a new version of the file in the workspace stream.



Name	Status	Version
findtags.pl	(backed)	kestrel_test\3
reporter.pl	(backed)	kestrel_devel\7

Rename reporter.pl to

report_writer.pl

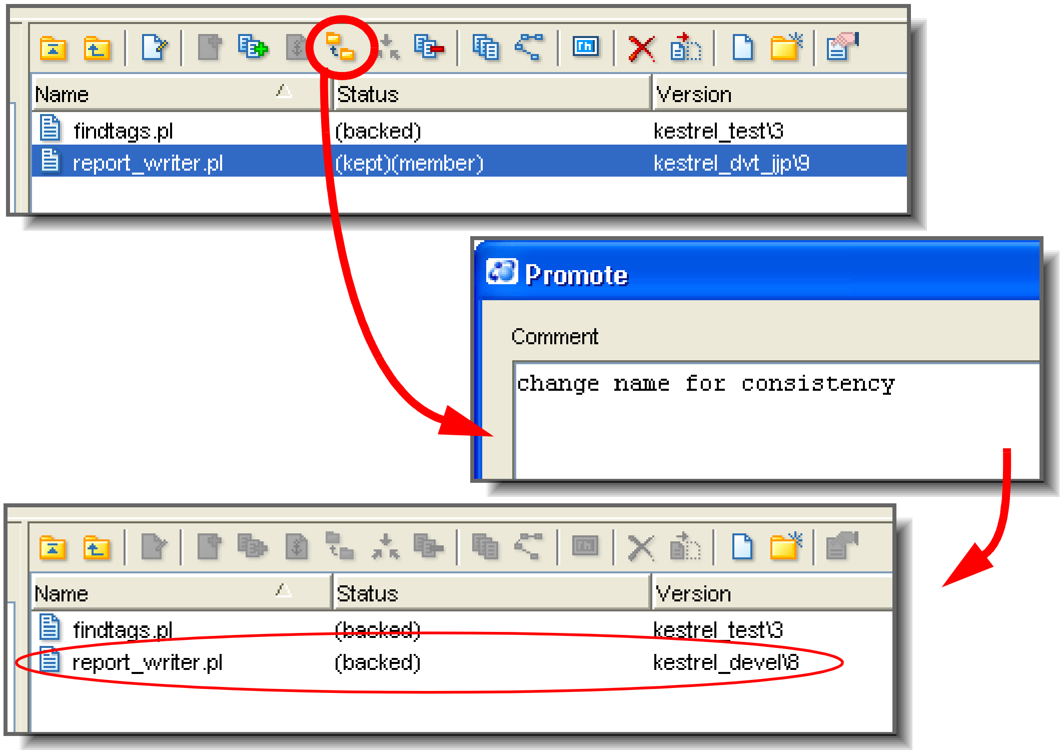
Name	Status	Version
findtags.pl	(backed)	kestrel_test\3
report_writer.pl	(kept)(member)	kestrel_dvt_jjp\9

As with a content change, the file's status changes to **(kept)(member)**. (If the file also has content changes that have not yet been saved with **Keep**, the status becomes **(modified)(member)**). Note that making a namespace change to a file "activates" it — creates a new version in the workspace stream and makes it a member of the workspace's default group — just like **Keep**ing a content change.

You can also rename and/or move a directory — see *Changing a Directory* on page 46.

Following Through by Promoting the Changes

Initially, the content changes and/or namespace changes you make to a file are recorded only in your workspace's private stream. This keeps your work isolated from your colleagues' work. When you're ready to share your changes to a file with your colleagues, you **Promote** the active version from your workspace stream to the backing stream. This makes your changes available to all workspaces that are based on the same backing stream.



Name	Status	Version
findtags.pl	(backed)	kestrel_test\3
report_writer.pl	(kept)(member)	kestrel_dvt_jjp\9

Comment

```
change name for consistency
```

Name	Status	Version
findtags.pl	(backed)	kestrel_test\3
report_writer.pl	(backed)	kestrel_devel\8

Note how the version-ID in the Version column changes:

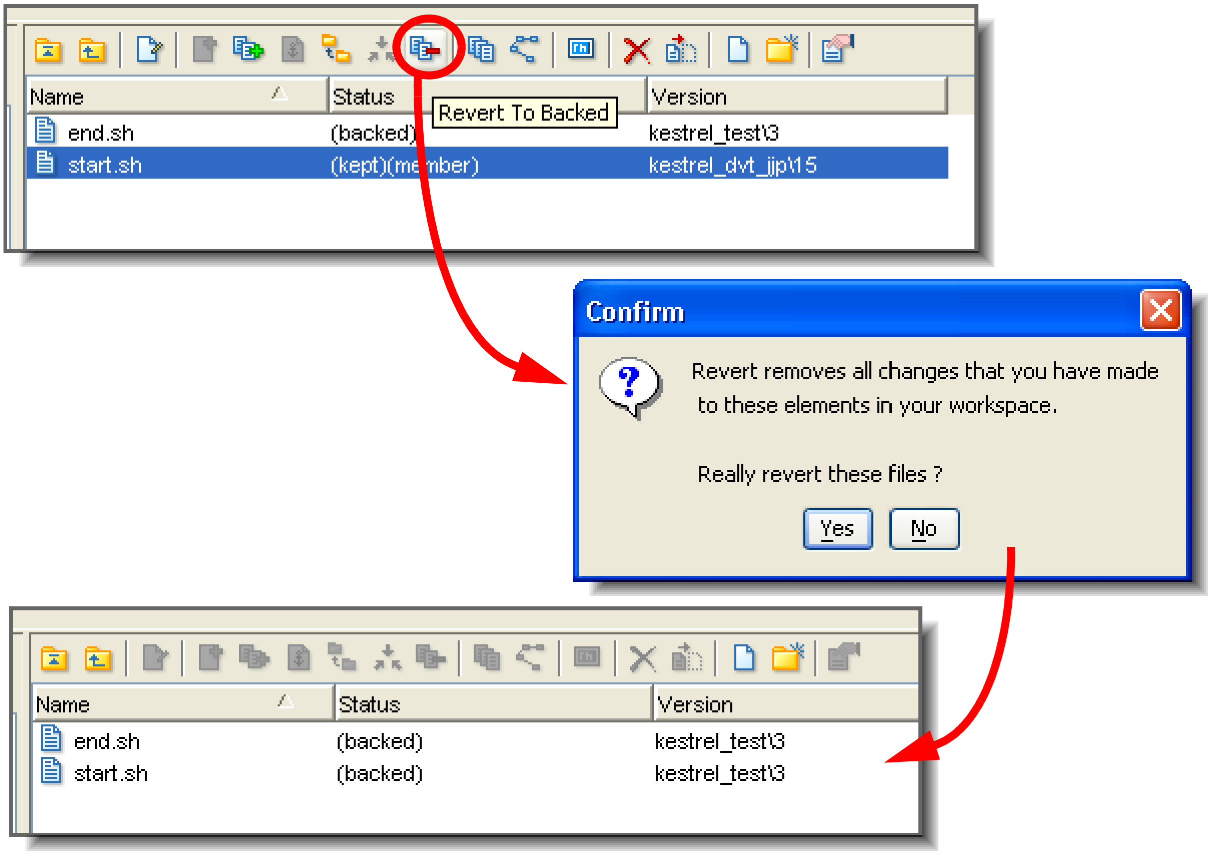
- Before the promotion, it indicates the particular version of the file that is active in your workspace (in this example, version **kestrel_dvt_jjp\9**).
- After the promotion, it indicates that version's newly created version-ID in the backing stream (in this example, **kestrel_devel\8**).

Having been promoted, the file is no longer active in the workspace (is no longer in the default group of the workspace stream). Accordingly, it loses its **(member)** status and returns to being **(backed)** — its original status before you started working on the file. That is, the workspace returns to "inheriting" the currently-active version of the file in the backing stream — which happens to be the version you just promoted there!

> Note: strictly speaking, you must **Keep** a file's changes before you can **Promote** them. But as a convenience, the File Browser enables the **Promote** command for files that are **(modified)**, but not **(kept)**. Invoking **Promote** performs both a keep transaction and a promote transaction.

Following Through by Undoing the Changes

Inevitably, you sometimes decide *not* to share your changes to a file with your colleagues — instead, you decide to discard the changes altogether. (Not all ideas are good ideas ...) The **Revert to Backed** command undoes all the content and/or namespace changes you've made to an active file. The file's status reverts to **(backed)**, and your workspace "rolls back" to using the version that it contained the last time the file's status was **(backed)**. It might be a version that your brought into your workspace with a recent **Update** command; or it might be a version that you created in your workspace, then **Promote**d to the backing stream.



Name	Status	Version
end.sh	(backed)	kestrel_test\3
start.sh	(kept)(member)	kestrel_dvt_jjp\15

Confirm

Revert removes all changes that you have made to these elements in your workspace.

Really revert these files ?

Name	Status	Version
end.sh	(backed)	kestrel_test\3
start.sh	(backed)	kestrel_test\3

> Note: if your changes to a file included moving it to a different directory, invoking **Revert to Backed** causes the file to disappear from its new location and return to its original location.

A file’s context menu (but not the File Browser’s toolbar) also contains a variant command, **Revert to Most Recent Version**. This command is useful if you’ve modified a file’s contents repeatedly, creating one or more intermediate versions in your workspace with the **Keep** command. **Revert to Most Recent Version** discards any content changes you’ve made since the most recent **Keep**. The file’s status reverts from **(modified)** to **(kept)**. The file remains active in the workspace, so it retains its **(member)** status.

Changing a Directory

In addition to tracking changes to files, as discussed in the preceding sections, AccuRev tracks changes to directories. AccuRev’s model for directory-level changes is simple, but somewhat different from the model used by the operating system (and by some other version-control systems). AccuRev considers the following to be changes to a directory:

- Renaming a directory
- Moving a directory to another location in the depot
- Deleting a directory

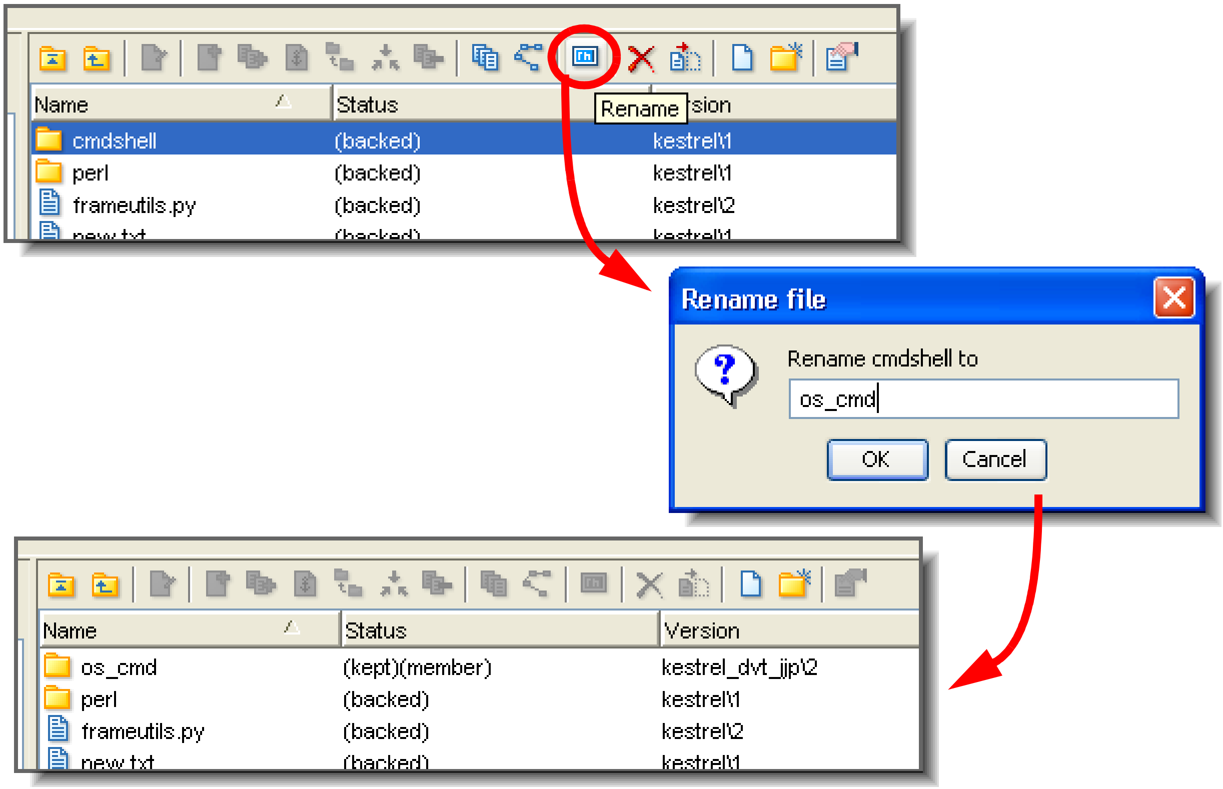
The following are *not* changes to a directory:

- Creating a new file (it’s a change to the file itself)

- Renaming an existing file (it's a change to the file itself)
- Deleting a file (it's a change to the file itself; see *Deleting a File — Intentionally and Permanently* on page 50)

Note that it is only changes involving a directory's pathname that are considered to be changes to the directory itself. Changes to a directory's contents are not considered to be changes to the directory — they are changes to the affected elements within the directory.

You change a directory's pathname in the same way you change a file's pathname — with the **Rename** command or with **Cut** and **Paste** commands (see *Renaming or Moving a File* on page 43). When you make such a change, AccuRev records a new version of the directory in the workspace stream.



Name	Status	sion
cmdshell	(backed)	kestrel\1
perl	(backed)	kestrel\1
frameutils.py	(backed)	kestrel\2
new.txt	(backed)	kestrel\1

Rename file

Rename cmdshell to

os_cmd

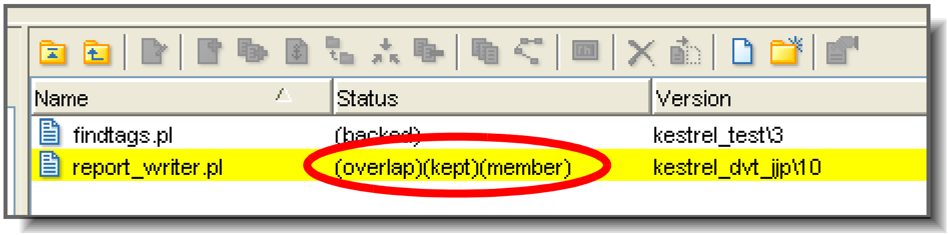
Name	Status	Version
os_cmd	(kept)(member)	kestrel_dvt_jjp\2
perl	(backed)	kestrel\1
frameutils.py	(backed)	kestrel\2
new.txt	(backed)	kestrel\1

The new version-ID appears in the Version column of the details pane (in this example, **kestrel_dvt_jjp\2**). And the directory's status changes to **(kept)(member)** — just as for a file's namespace change.

Merging Your Changes with Someone Else's Changes

AccuRev supports concurrent development: two or more users can start with the same version of a file and make changes to that file independently — both content changes and namespace changes. After one of the users **Promote**s his changes to the backing stream, each of the others must **Merge** her own changes with the newest version in the backing stream. Merging is described in chapter *The AccuRev Diff, Merge, and Patch Tools* on page 95. We describe here how a merge scenario is reflected in the File Browser display.

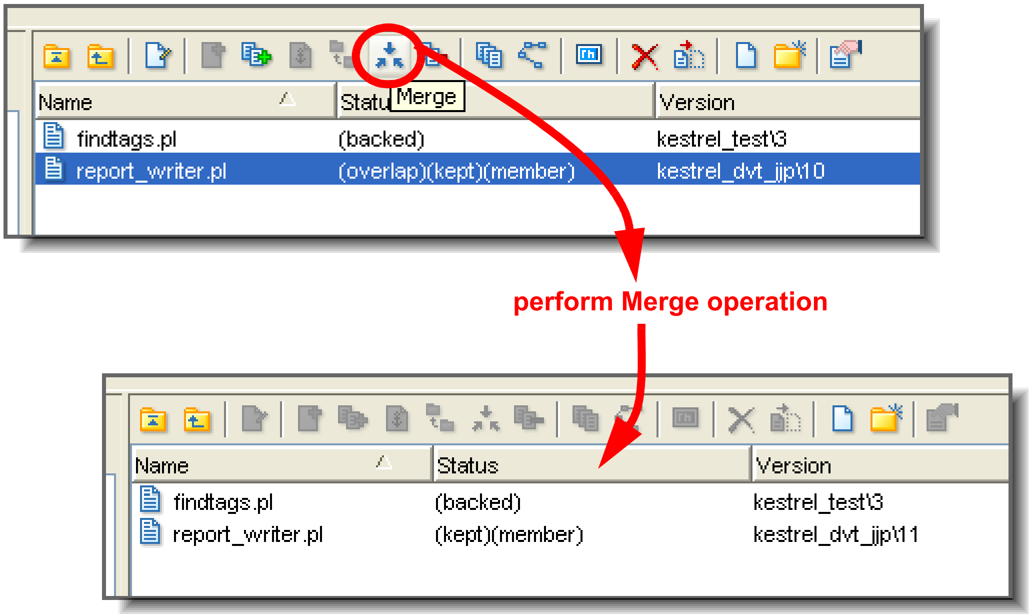
As you work on a file, **Keep**ing intermediate versions in your workspace, the file's status alternates between **(modified)** and **(kept)**. At any point, you may notice that an additional indicator, **(overlap)**, appears in the Status column. To make sure you notice, the File Browser displays the entry with a yellow highlight.



Name	Status	Version
findtags.pl	(backed)	kestrel_test\3
report_writer.pl	(overlap)(kept)(member)	kestrel_dvt_jjp\10

This means that a new version of the file has entered the backing stream. Typically, one of your colleagues has edited the file and promoted her version to the backing stream. It may also be that someone has promoted a new version of the file to a higher-level stream, and the backing stream dynamically inherits the new version "from on high".

Whenever a file's status is **(overlap)**, the **Merge** command is enabled. The execution of a **Merge** command concludes with the **Keep**ing of a new version in your workspace.



Name	Status	Version
findtags.pl	(backed)	kestrel_test\3
report_writer.pl	(overlap)(kept)(member)	kestrel_dvt_jjp\10

perform Merge operation

Name	Status	Version
findtags.pl	(backed)	kestrel_test\3
report_writer.pl	(kept)(member)	kestrel_dvt_jjp\11

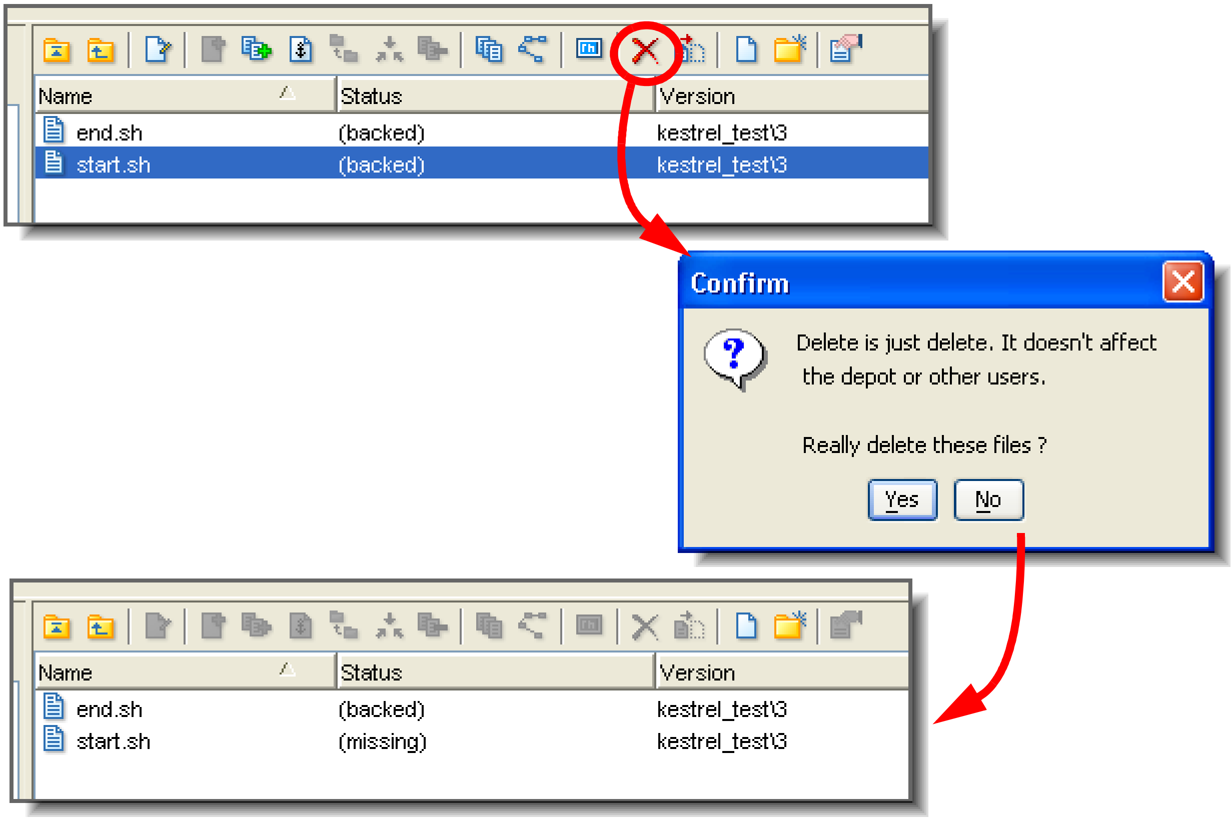
When you **Promote** this merged version to the backing stream, the file's status returns to **(backed)**. This step is exactly the same as in section *Following Through by Promoting the Changes* on page 44.

Deleting a File — Accidentally or Temporarily

An AccuRev workspace is an ordinary directory tree (the workspace directory tree), typically located on your machine's hard drive. Nothing prevents you (or perhaps, some rogue cleanup script) from using operating system commands to delete one or more of the files under version control. On occasion, you may even want to delete some files temporarily — for example, to test the robustness of your build or installation procedure.

By definition, deleting a file at the operating system level makes it disappear from disk storage. Operating system tools, such as Windows Explorer or the Unix **ls** command, will detect that the file no longer exists. But the file does *not* disappear from the File Browser display. AccuRev knows that the file *should* be in the workspace, because the file element still exists in the workspace's built-in stream. (The workspace stream is located in the AccuRev depot. It's unaffected by the operating system's delete-file commands.)

Accordingly, when a version-controlled file is deleted at the operating system level, the File Browser continues to list it, but indicates the file's status as **(missing)**.



Name	Status	Version
end.sh	(backed)	kestrel_test\3
start.sh	(backed)	kestrel_test\3

Confirm

Delete is just delete. It doesn't affect the depot or other users.

Really delete these files ?

Name	Status	Version
end.sh	(backed)	kestrel_test\3
start.sh	(missing)	kestrel_test\3

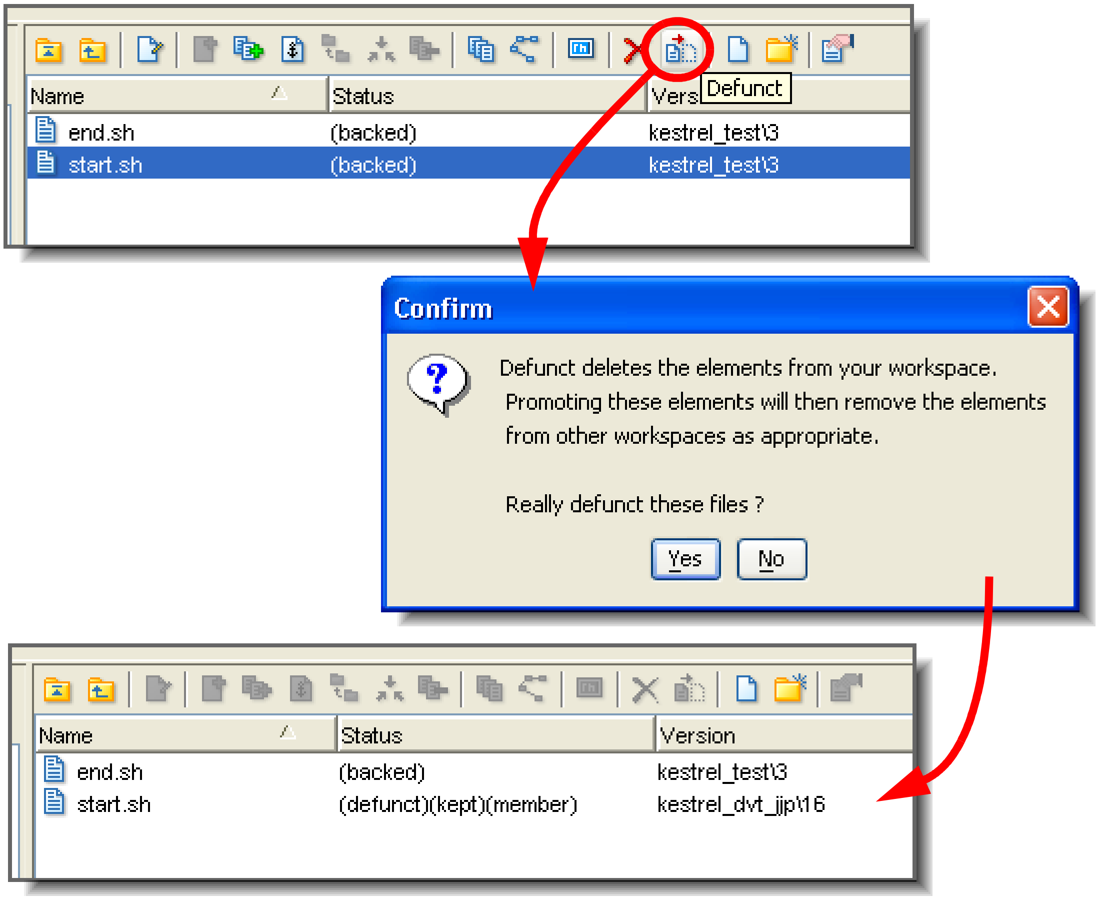
Note that the Version column continues to indicate which version of the file should be in the workspace. To restore that version, invoke the **Populate** command from the missing file's context menu.

Deleting a directory at the operating system level is similar to (but potentially more destructive than) deleting a file. The entire directory subtree is deleted from disk storage, and the directory is listed by the File Browser as **(missing)**. To recover the entire directory tree, invoke the **Populate** command from the directory's context menu. Be sure to select the Recursive option from the Populate Files dialog box.

Deleting a File — Intentionally and Permanently

Sometimes, you want to delete a file permanently. That is, you want the file to disappear from your workspace, and from other users' workspaces, too. The file might be related to a product feature that was cancelled. Or perhaps a code reorganization rendered the file unnecessary.

Deleting a file at the AccuRev depot level (rather than simply at the operating system level) is called defuncting, and is implemented by the **Defunct** command.



Name	Status	Vers
end.sh	(backed)	kestrel_test\3
start.sh	(backed)	kestrel_test\3

Confirm

Defunct deletes the elements from your workspace.
Promoting these elements will then remove the elements from other workspaces as appropriate.

Really defunct these files ?

Name	Status	Version
end.sh	(backed)	kestrel_test\3
start.sh	(defunct)(kept)(member)	kestrel_dvt_jjp\16

Defuncting a file removes it from the workspace's disk storage (that is, deletes the file at the operating system level). In addition, the **Defunct** command is recorded in the depot. It may be surprising at first, but AccuRev manages the defuncting of a file in the same way as it manages the creation of new versions. To AccuRev, defuncting is just another kind of change that can happen to a file:

- Defuncting "activates" a file in the workspace stream, recording the fact that you've made a change to the file. In addition to getting **(defunct)** status, the file gets the **(kept)** and **(member)** statuses, just as if you had performed a **Keep** command.
- AccuRev records the change as a new version of the file in the workspace stream (in this example, version **kestrel_dvt_jjp\16**).
- At first, the defuncting of a file is isolated to your own workspace. The file continues to exist in other users' workspaces.

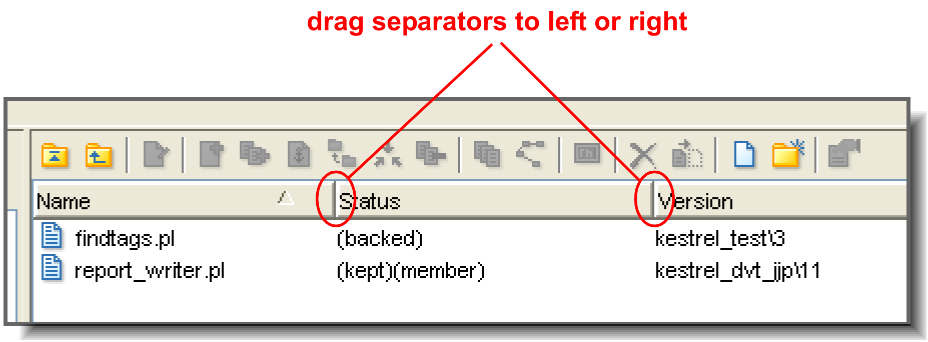
- To “share” the defuncting of a file, you **Promote** the change to the backing stream. This causes the file to disappear from your workspace stream, and from the File Browser display. The file will disappear from other users’ workspaces when they invoke the **Update** command.
- As always, you propagate the change — in this case, removal of the file — throughout the depot by promoting the defuncted file from the backing stream to the depot’s higher-level streams.

Note that the **Defunct** command does not actually remove the file element from the AccuRev repository, even if you promote the change to the depot’s base stream. Because AccuRev is TimeSafe, old versions of the file continue to exist in snapshots of streams, in History Browser displays, etc. You can restore such elements with the CLI command **undefunct**.

Rearranging Columns / Sorting Rows

The details pane display is a multiple-column table. As with all such tables in AccuRev, you can resize and rearrange its columns, and you can sort its rows.

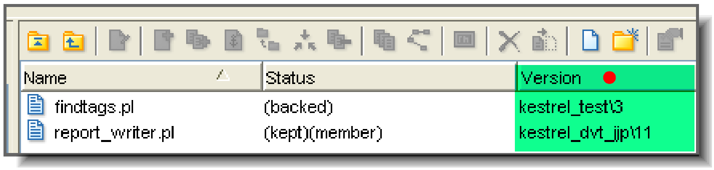
To resize the columns, click and drag the column separators.



drag separators to left or right

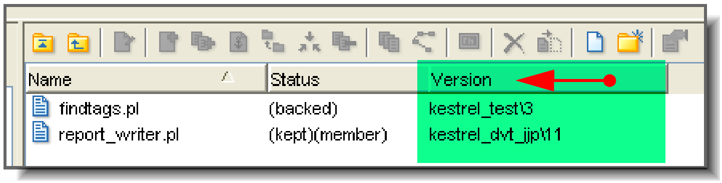
Name	Status	Version
findtags.pl	(backed)	kestrel_test\3
report_writer.pl	(kept)(member)	kestrel_dvt_jjp\11

To rearrange the columns, click and drag the column headers.



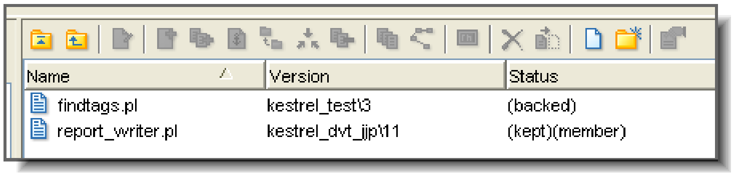
1. click

Name	Status	Version
findtags.pl	(backed)	kestrel_test\3
report_writer.pl	(kept)(member)	kestrel_dvt_jjp\11



2. drag

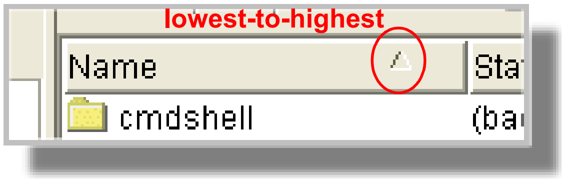
Name	Status	Version
findtags.pl	(backed)	kestrel_test\3
report_writer.pl	(kept)(member)	kestrel_dvt_jjp\11



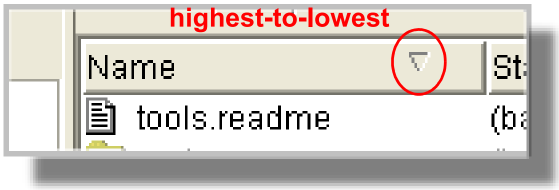
3. column snaps into place

Name	Version	Status
findtags.pl	kestrel_test\3	(backed)
report_writer.pl	kestrel_dvt_jjp\11	(kept)(member)

Initially, the rows of the details pane, each representing a file or directory, are sorted on one column (the Name column). A direction icon in the column header indicates whether the sort is lowest-to-highest or highest-to-lowest.



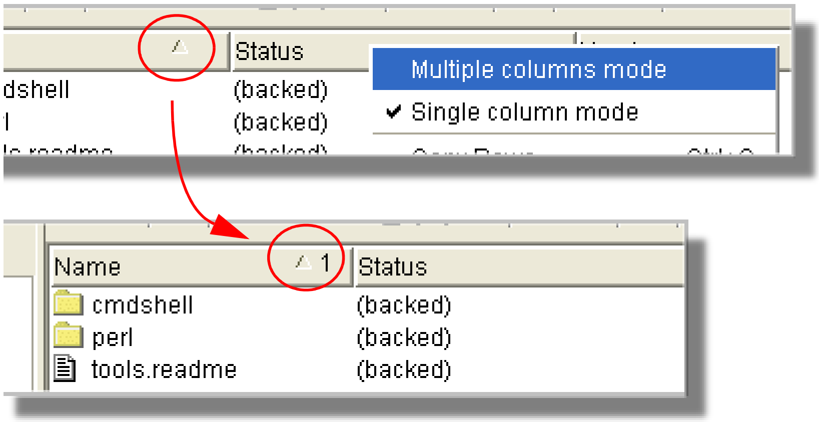
This is termed single column mode. Left-click on any column header to continue in this mode:



- Click on the current sort column to reverse the sort order.
- Click on another column to make it the sort column.

Note: when sorting on the Name column, a lowest-to-highest sort places all directories before all files; a highest-to-lowest sort places all files before all directories.

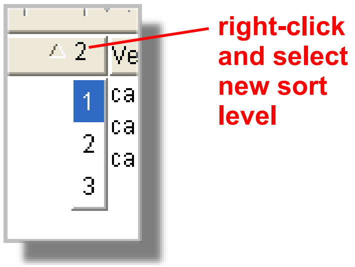
You can switch at any time to multiple columns mode, in which you define a primary sort column, a secondary sort column, and so on. Right-click any column header to switch sort modes. A “1” appears next to the direction icon in the current sort column, indicating that this is now the primary sort column.



Click another column to make it the secondary sort column. A direction icon annotated with “2” appears in this column, and the rows are reordered according to the two-level sort.

In multiple columns mode, you can:

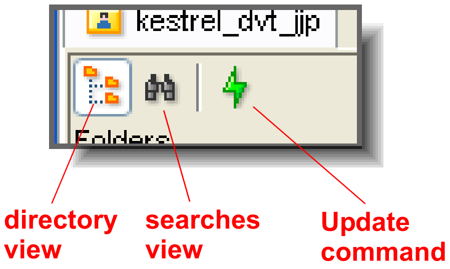
- Left-click on any column header without a direction icon: this defines an additional sort level, using that column.
- Left-click on any column header that already has a direction icon: this reverses the sort order at that sort level.
- Right-click on any sort column’s level number (1, 2, etc.) to change the sort level of that column.



The rows are reordered automatically each time you perform any of these actions.

Working in the Navigation Pane

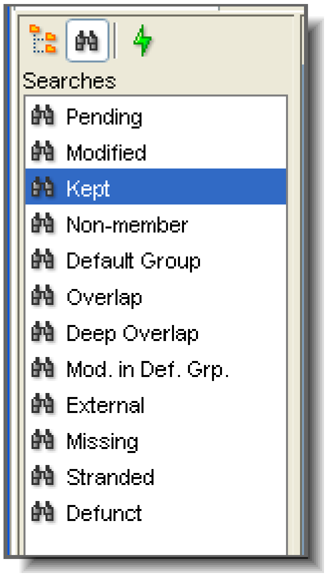
The toolbar at the top of the File Browser’s navigation pane includes three buttons:



- The **Directory View** button configures the details pane to display one directory’s contents at a time. A standard hierarchy control in the navigation pane enables you to switch among the workspace’s directories.

  All the descriptions and examples in the preceding sections of this chapter have used directory view.

- The **Searches View** button configures the details pane to display a selected subset of the depot’s files and directories. The navigation pane lists the available search criteria; when you choose one, AccuRev and displays only those objects selected by the search criterion.

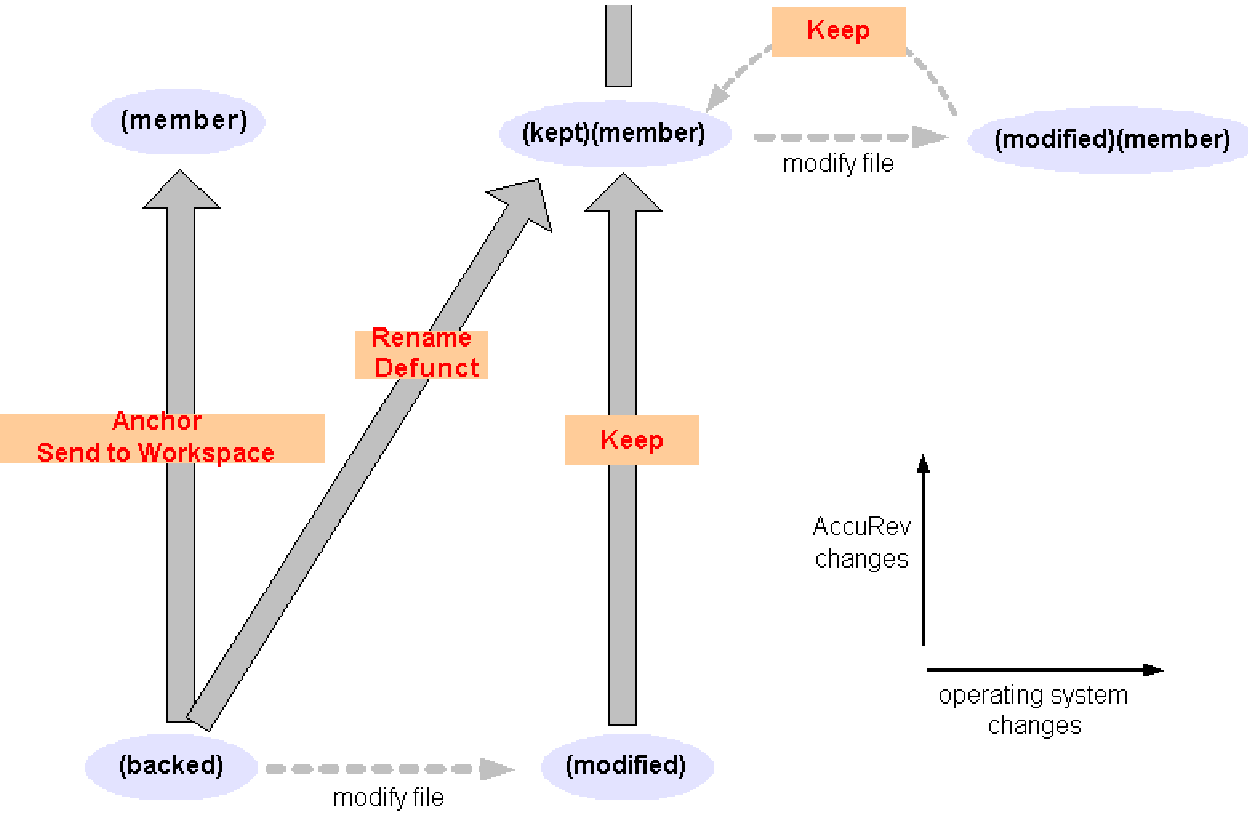


- The **Update** button invokes the Update command, copying new versions of elements from the backing stream into the workspace.

Searches and File Statuses

All of the search criteria available in searches view select files and directories based on their AccuRev status. For example, a **Kept** search selects all elements — files and directories — in the workspace that are listed with the **(kept)** status indicator (and perhaps other indicators, too). The searches are not mutually exclusive: some objects may be selected by more than one search criterion — for example, **Kept** and **Pending**: every object selected by a **Kept** search will also appear in a **Pending** search.

Before describing the search criteria, we present a diagram that shows how the statuses indicate an element’s development progress. This diagram leverages the central AccuRev concept of promotion, which suggests that as more work is performed on an element, it “rises to a higher level” of development. The diagram also depicts the fact that AccuRev tracks changes to an element in two “dimensions”: it records changes made by AccuRev commands as new versions in the depot; and it detects that a file’s contents have changed at the operating system level.



This diagram captures the following facts:

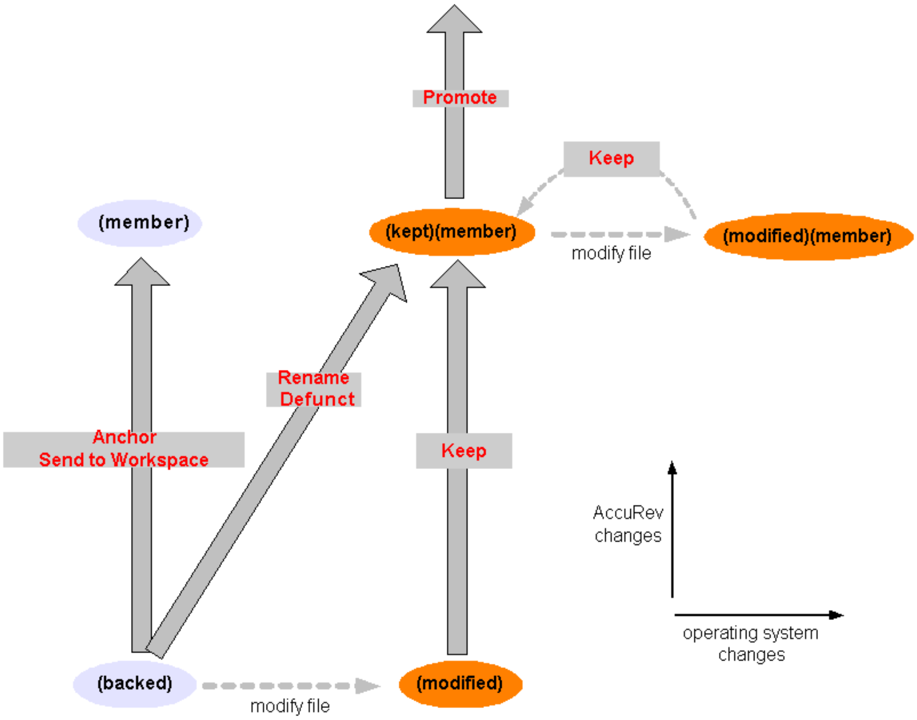
- Before you start working on an element, it's unchanged along both the AccuRev dimension and the operating-system dimension. Its status is **(backed)**.
- When you modify the contents of a file, it changes along the operating-system dimension only, and becomes **(modified)**.
- The following involve changes along the AccuRev dimension. This makes the element active in the workspace, so its status indicates that it is a **(member)** of the default group.
  - When you **Keep** a file that you've modified, its status becomes **(kept)**.
  - An element can also achieve **(kept)** status through a namespace change — **Rename** or **Defunct** command. This is a change along both the AccuRev and operating-system dimensions. (The CLI command **undefunct**, which reinstates a previously defuncted object, works similarly. This command is not in the GUI.)
  - The **Anchor** command change a file along the AccuRev dimension only. The file's status becomes **(member)**.
- You can create any number of intermediate versions of a file in the workspace, by repeatedly modifying the file then **Keep**ing it. Throughout this process, the file remains a **(member)** of the workspace's default group. In addition, its status toggles between **(kept)** and **(modified)**.

- You can **Promote** an element whose status is **(kept)**. This returns the element's status to **(backed)** — the workspace now uses the newly promoted version in the backing stream.

The diagram makes it easy (we hope!) to see how each search criterion works:

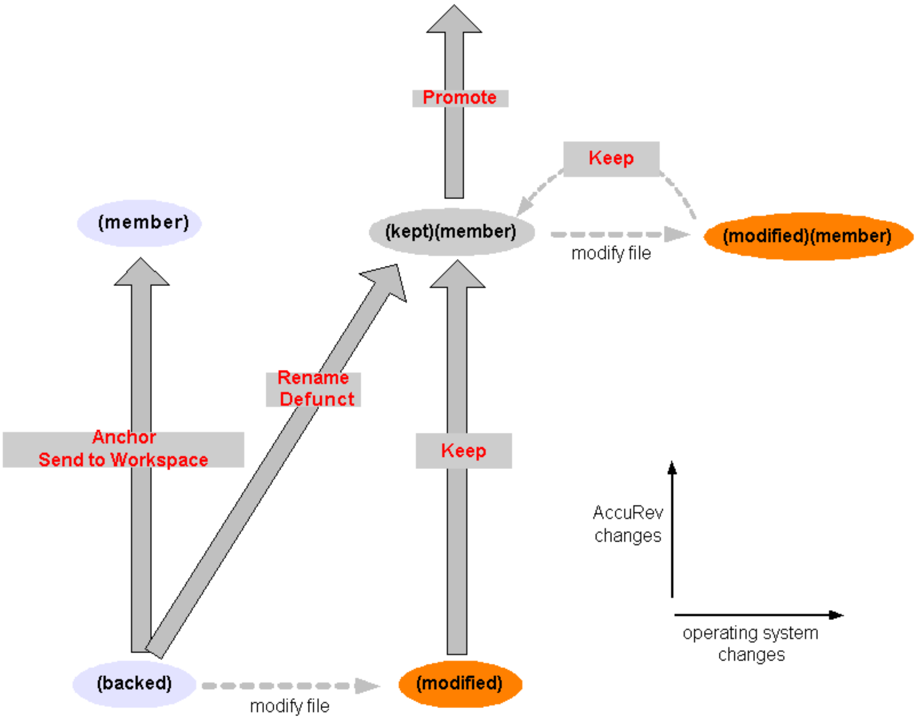
Pending

Selects elements whose status includes either **(modified)** or **(kept)**. These are elements that have had an AccuRev content or namespace change, or a change in file contents.



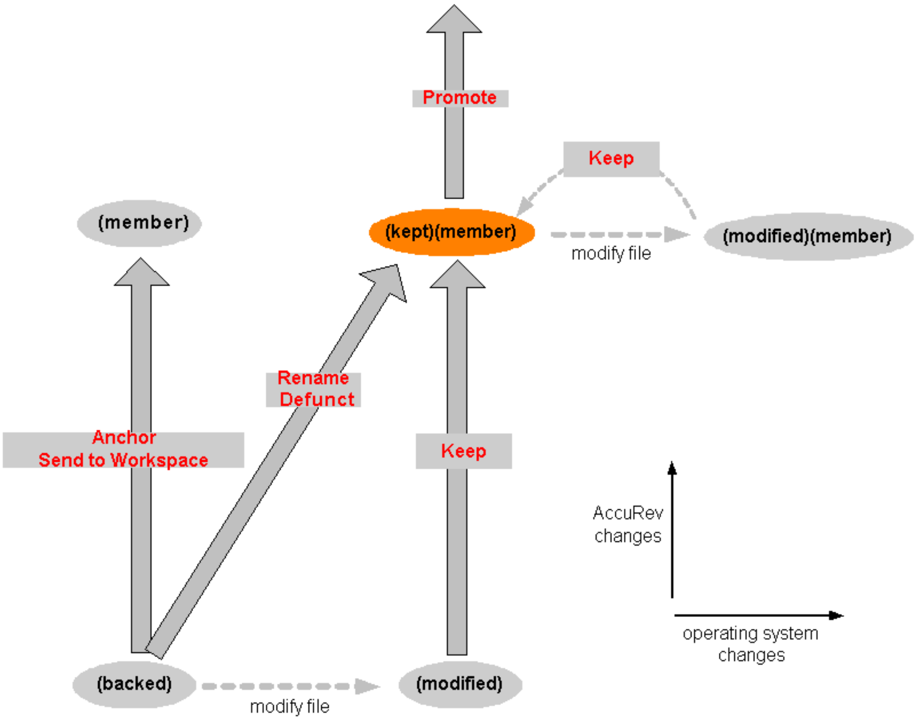
Modified

Selects files whose status includes **(modified)**. This indicates a content change that has not yet been preserved with **Keep** (but may have followed a previous **Keep**).



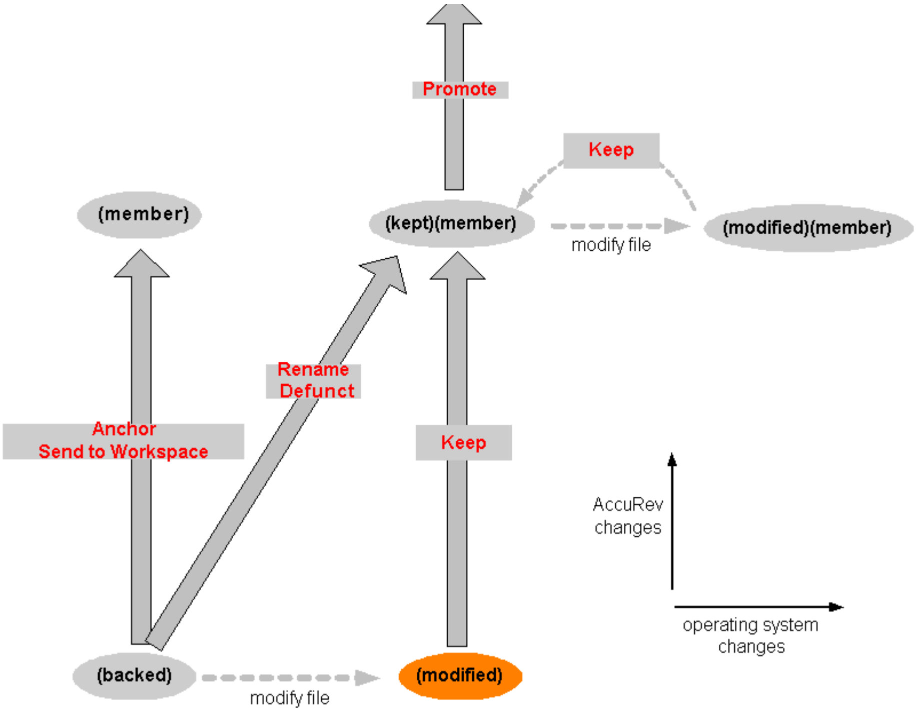
Kept

Selects files whose status includes **(kept)**. These are files whose content changes have all been preserved with **Keep**, or elements with namespace-level changes (**Rename** or **Defunct** command).



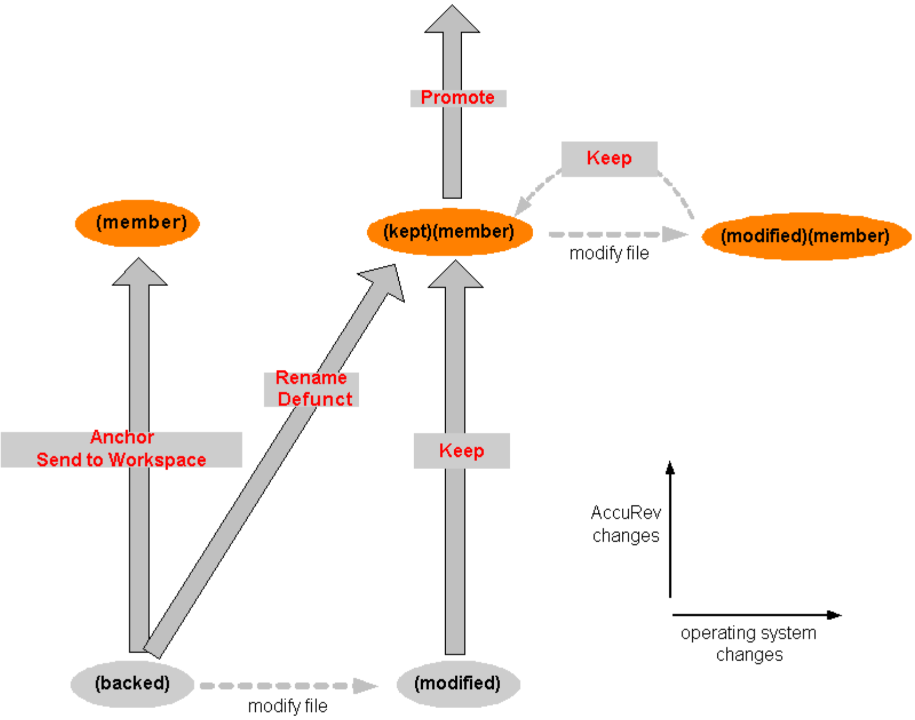
Non-member

Selects files whose status includes **(modified)** but not **(member)**. These are files that have content changes, but are not in the workspace's default group. (They have not been activated with a **Keep** or **Anchor** command since the file's last update or promotion.)



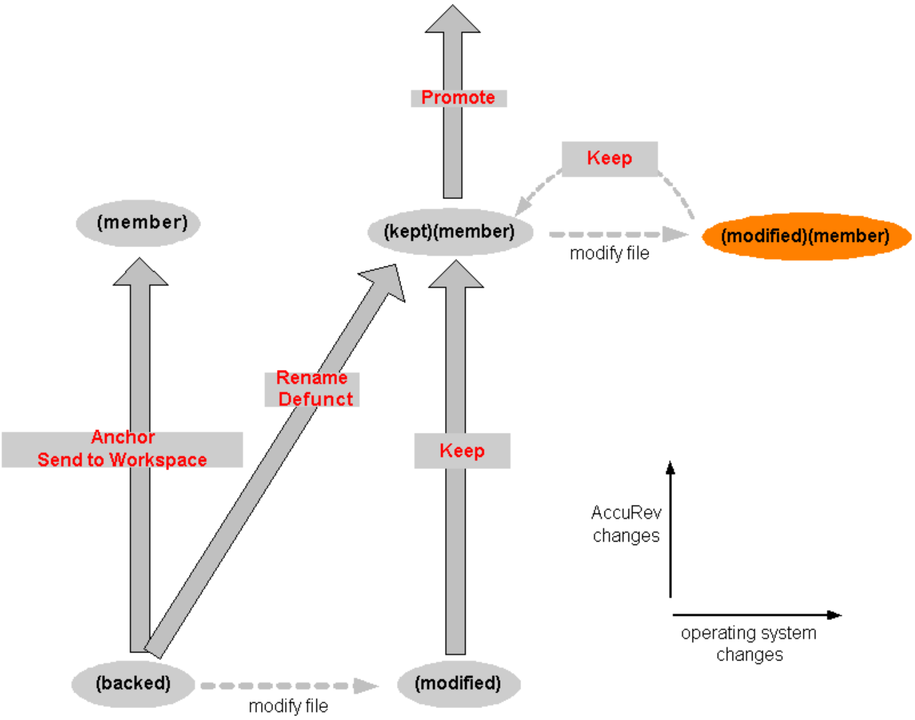
Default Group

Selects elements whose status includes **(member)**. These are elements in the workspace's default group, for which you've entered one of these commands: **Keep**, **Rename**, **Defunct**, **Anchor**.



Modified in Default Group

Selects elements whose status includes both **(modified)** and **(member)**. This is the intersection of the sets of files selected by the Modified and Default Group search criteria.



The following search criteria are not accounted for in the status diagrams above:

Overlap

Selects the subset of Pending files whose status includes **(overlap)** — the current version in the parent (backing) stream is not an ancestor of your workspace version. This means there might be changes in the backing stream version that are not present in your workspace version. You need to perform a **Merge** with the backing stream version before you can **Promote** your workspace version.

Deep Overlap

Reports **(overlap)** files in the current workspace or stream. Also reports **(overlap)** versions in the *parent* stream, in the *grandparent* stream, ... all the way up the depot hierarchy.

External

Selects files and directories that exist in the workspace directory tree, but have never been placed under version control with the **Add** command.

Missing

Selects elements that *should* be in the workspace, but aren't. That is, there's a version of the file or directory in the workspace stream, but the file or directory was removed from disk storage (the workspace tree) by an operating system command or some non-AccuRev program.

In a sparse workspace, only a subset of the depot's elements are loaded into the workspace tree. In such a workspace, elements that have not been loaded have the status **(missing)**, and so are listed by this search.

Stranded

Selects the elements in the default group of a workspace or dynamic stream that have become stranded. A default group member becomes stranded when there is no pathname to the element in that workspace or stream.

Here's the most typical scenario for stranding a file in a workspace:

- Create a new version of a file with **Keep**. This places the file element in the workspace's default group.
- Remove the file's parent directory with **Defunct**. With the parent directory gone, there's no pathname in the workspace to the file element.

In the details pane, stranded files are identified by their element-IDs (since they have no pathname in the workspace or stream).

Defunct

Selects the elements in the default group of a workspace or dynamic stream whose status is **(defunct)**.

Search Performance Optimizations

Many of the searches described above require that AccuRev consider every file in your workspace, even the ones that you haven't placed under version control (e.g. editor backup files, files produced by software builds). If your workspace contains many thousands of files, such operations can be time-consuming.

Here are some situations that require the File Browser to examine every file in the workspace:

- When applying the **External** search criterion, each filename must be sent to the AccuRev Server, to determine whether the file has already been placed under version control.
- When applying the **Modified** search criterion, the checksum of each file in the workspace must be compared to the checksum of the corresponding "latest" version in the depot.

The searches that require a full-workspace examination are: **Pending**, **Modified**, **Non-member**, **Overlap**, **Deep Overlap**, **External**, and **Missing**.

To speed performance, the File Browser can use several optimizations. Each can significantly reduce the number of files to be considered by a search.

Pathname Optimization

You can configure the File Browser to ignore files based on their pathnames. For example, you might know that files with a **.exe** suffix are not version-controlled, or that the entire contents of directories whose names start with **build_** are not version-controlled. Ignored files are omitted from the details pane when the File Browser displays search results.

The environment variable ACCUREV_IGNORE_ELEMS controls this optimization. For example, you might set this variable to the following value before starting the AccuRev GUI:

```
*.exe  */build_*
```

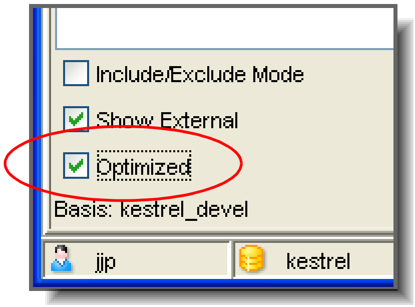
Any file or directory that matches any of the specified patterns will be ignored by the File Browser when it applies the **Pending**, **Modified**, **Non-member**, or **External** search criterion. For details on this optimization, see *Using the ACCUREV_IGNORE_ELEMS Environment Variable* on page 39 of the *AccuRev Technical Notes* manual.

Timestamp Optimization

Note: as of Version 3.3.1, the timestamp optimization is available both in searches view and in directory view.

AccuRev considers the development work you perform to be continually changing the synchronization between your workspace and the backing stream. As you modify existing files and create new ones, your workspace becomes out-of-sync with the backing stream. Each time you execute an **Update** command, your workspace becomes more closely (sometimes, completely) synchronized with the backing stream.

When searching for workspace changes that have not yet been recorded in the depot, AccuRev can assume that all such changes are contained in files with timestamps *after* the time of the workspace's most recent update. It makes this assumption if the **Optimized** checkbox at the bottom of the navigation pane is checked. Any file or directory with an older timestamp is ignored, and is omitted from the details pane when the File Browser displays search results.



☐ Include/Exclude Mode
☑ Show External
☑ Optimized
Basis: kestrel_devel
jjp | kestrel

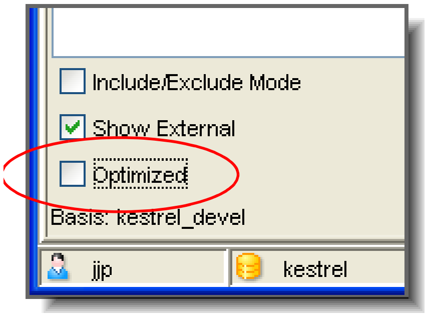
Given the way in which the **Update** command operates, this assumption about timestamps is usually valid — and thus, so is the performance optimization. (For details, see *Updating the Workspace* on page 68.) In general, if the only way you change source files is with a text editor, the assumption will be valid: each new change gets timestamped with the current time.

But there are tools that can introduce "a new change with an old timestamp" into a workspace:

- The operating system's "copy file" command can preserve old timestamps when creating a new copy of a file. Similarly, the **tar** (Unix) and **zip** (Unix and Windows) utilities can preserve old timestamps when they copy files out of an archive.
- Less likely but possible, a severely-lagging system clock on an AccuRev client machine can cause edited files to get timestamps that precede the most recent update. (AccuRev commands won't execute if the client machine's clock is not synchronized with the server machine's clock. But something bad might happen to the client machine's clock at a time when no AccuRev commands are being executed.)

If either of these situations applies to you, clear the **Optimized** checkbox before choosing one of the relevant search criteria. Turning off timestamp optimization can slow performance significantly, but it guarantees that no file will be overlooked because of its timestamp.

Note: clearing the **Optimized** checkbox turns off timestamp optimization, but not pathname optimization.

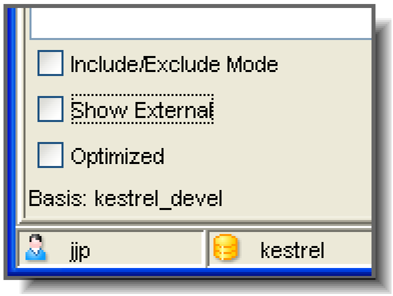


☐ Include/Exclude Mode
☑ Show External
☐ Optimized
Basis: kestrel_devel
jjp | kestrel

If you know exactly which modified files have old timestamps, you don't need to turn off the timestamp optimization. Instead, just update the timestamps to the current time, using the CLI command **accurev touch** (see page 161 in the *AccuRev User's Manual (CLI)*).

External File Exclusion

To have the File Browser ignore all file system objects with **(external)** status, clear the **Show External** checkbox. By default, this checkbox is checked, so that the File Browser shows both elements and external objects.

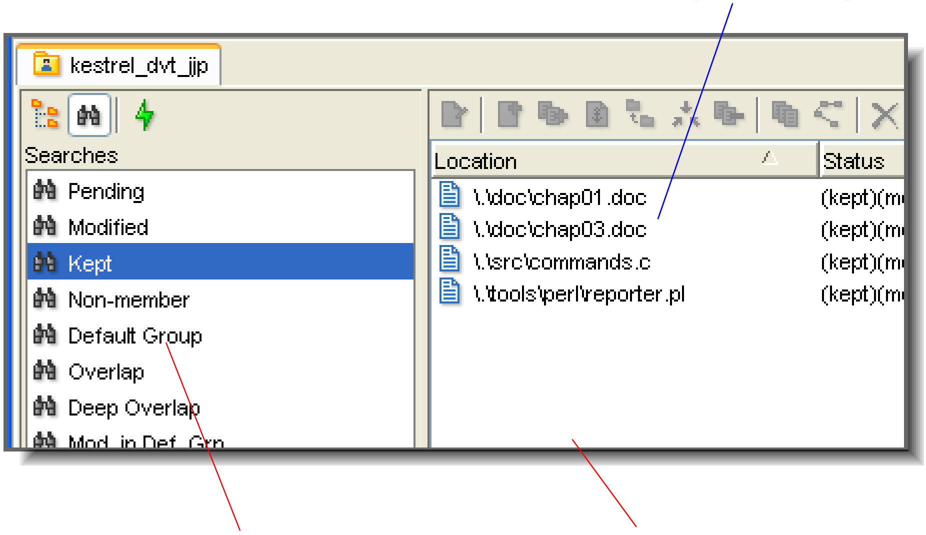


☐ Include/Exclude Mode
☐ Show External
☐ Optimized
Basis: kestrel_devel
jjp | kestrel

Operating on Files Selected by a Search

After using a search to select a subset of your workspace's files, the File Browser displays all of the selected files in the details pane. Typically, the selected files are located in a number of different directories, throughout the workspace. Accordingly, each file is identified by its complete depot-relative pathname, rather than by a simple filename.

location displayed as
depot-relative pathname



Location	Status
\.\doc\chap01.doc	(kept)(m
\.\doc\chap03.doc	(kept)(m
\.\src\commands.c	(kept)(m
\.\tools\perl\reporter.pl	(kept)(m

1. choose a search criterion
in navigation pane

2. details pane displays all items
selected by that search criterion

You can work with these files (and directories) using the same techniques and commands described in *Working in the Details Pane* on page 36. That is, the details pane works the same way whether you're using directory view or searches view. Here are a couple of notes:

- Pressing **Ctrl-A** to select all items in the details pane is particularly useful in searches view. For example, to promote all files that you've saved with **Keep**, you can (1) switch to searches view and choose the **Kept** search criterion, (2) press **Ctrl-A** to select all the kept files, (3) invoke the **Promote** command.
- A file can disappear from the details pane if you change it in a search-related way. For example, after you promote all the files displayed by a **Kept** search, the files' status changes from **(kept)** to **(backed)**. Accordingly, the files disappear from the **Kept** search display.

Working in Include/Exclude Mode

By default, a workspace contains a copy of each file under version control in a particular depot. (Roughly speaking: it's more accurate to say that a workspace contains a copy of each version in a particular stream.) But sometimes, you don't want and need a copy of *every* file — for example, your current assignment doesn't include "rebuilding the world" each night. The depot might contain many thousands of files, of which you might need only a small subset.

In this case, it makes sense to configure your workspace to contain a specified subset of the depot's files, rather than all of them. The benefits can be quite significant:

- less clutter, allowing you to concentrate on the files that are important to you
- less disk space required to store your workspace on your machine
- faster backups of your workspace
- faster AccuRev processing of your workspace, especially during the **Update** command

Prior to Version 3.5, AccuRev supported this capability through sparse workspaces. A sparse workspace started out empty; you added certain elements to the workspace using the **Populate** command. Those elements were maintained in the regular manner, using **Keep**, **Promote**, and **Update**. Other elements in the depot were ignored.

This scheme was satisfactory for many purposes, but there were some drawbacks. For example, an **Update** would not bring newly created elements into your workspace, just new versions of the elements that you had already **Populate**'d.

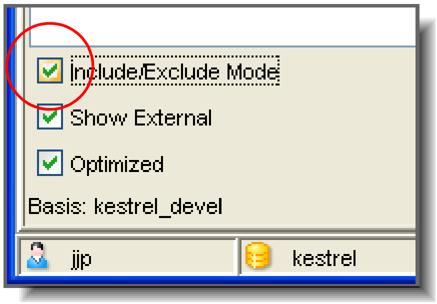
As of Version 3.5, AccuRev features a more powerful facility, called include/exclude. You can no longer create a sparse workspace (although you can continue to use existing sparse workspaces). You use include rules and exclude rules to specify which elements are to be represented in a workspace:

- An include rule can specify an individual file, an individual directory's contents, or the contents of an entire directory tree. (This parallels the choices for using **Populate** in a sparse workspace.)
- An exclude rule can specify an individual file or an entire directory tree.
- Rules at lower levels of the directory hierarchy can refine rules at a higher level. For example, a graphic artist might add a rule to exclude everything below the directory **src**, but then add another rule to include the single subdirectory **src/gui/images/icons**.

The include/exclude facility has significant advantages over sparse workspaces:

- Include rules and exclude rules are official attributes of the workspace, maintained in the AccuRev repository. Since AccuRev knows that a directory is included in the workspace, it "remembers" to bring newly created elements in that directory into the workspace during updates.
- You can use include rules and exclude rules to configure streams as well as workspaces. The rules are inherited down the stream hierarchy. For example, to make the **marketing** directory invisible to all developers, you can exclude that directory from a stream that all developers' workspaces are based on (either directly or through multiple stream levels).

Enabling Include/Exclude Mode



☑ Include/Exclude Mode

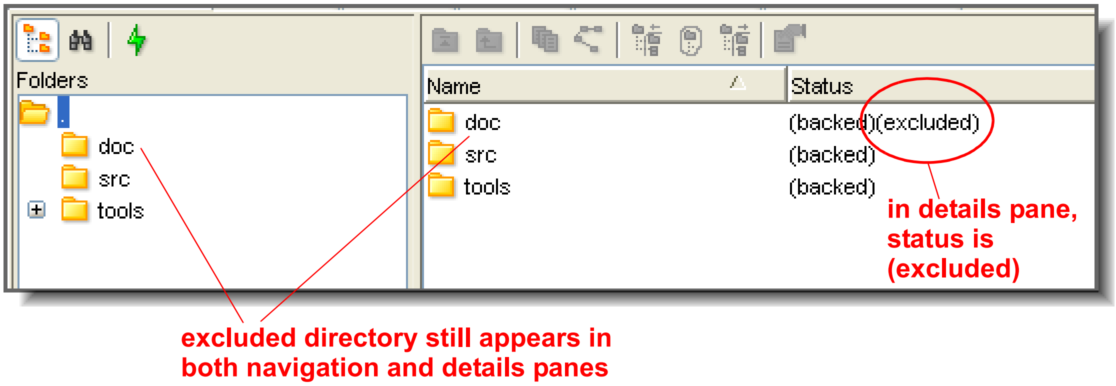
☑ Show External

☑ Optimized

Basis: kestrel_devel

To enable include/exclude mode in a workspace or stream, use the checkbox at the bottom of the File Browser's navigation pane. In this mode, the File Browser display changes in several ways.

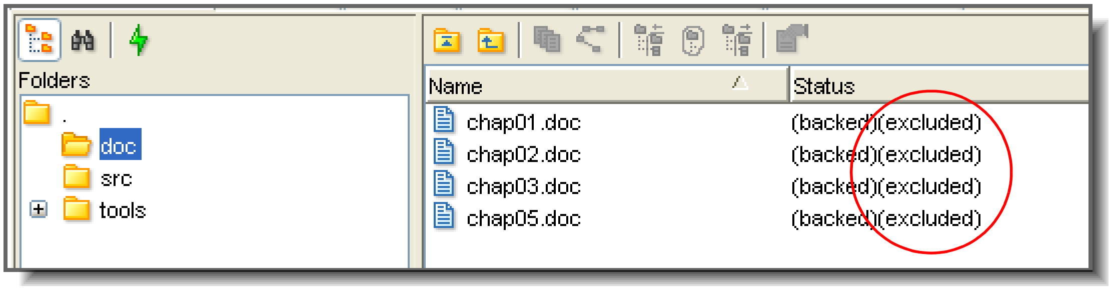
Both the navigation and details panes show *all* elements in the workspace or stream. Elements that are currently excluded have an **(excluded)** indicator in the Status column. For example, if top-level directory **doc** is excluded, the File Browser still shows it in both the navigation and details panes:



Name	Status
doc	(backed)(excluded)
src	(backed)
tools	(backed)

Folders
.
doc
src
tools
in details pane, status is (excluded)
excluded directory still appears in both navigation and details panes

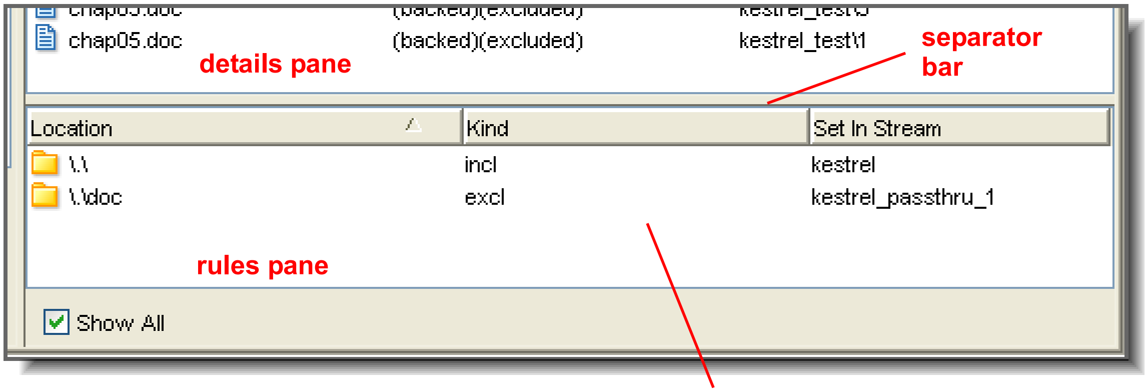
You can navigate into an excluded directory. All the elements therein will have **(excluded)** status.



Name	Status
chap01.doc	(backed)(excluded)
chap02.doc	(backed)(excluded)
chap03.doc	(backed)(excluded)
chap05.doc	(backed)(excluded)

The File Browser gets a new pane, the rules pane, in include/exclude mode. The table in this pane shows include and exclude rules that affect the current workspace or stream:

- **Show All** checkbox cleared: only the rules that were explicitly set for the current workspace
- **Show All** checkbox checked: also includes rules inherited from higher-level streams. The **Set in Stream** column indicates which higher-level stream.

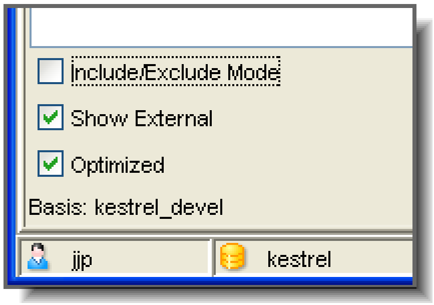


Location	Kind	Set In Stream
\.\	incl	kestrel
\.\doc	excl	kestrel_passthru_1

The details pane has a different toolbar in include/exclude mode; most of the command buttons you use for day-to-day development disappear, replaced by buttons for maintaining the include and exclude rules.

See below for sections that describe how to work in include/exclude mode.

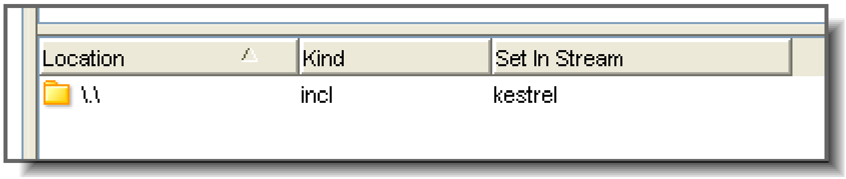
Leaving Include/Exclude Mode



To leave include/exclude mode in a workspace or stream, clear the checkbox at the bottom of the File Browser's navigation pane. The rules pane disappears, and the details pane reverts to:

- *not* displaying elements that have been excluded from the current workspace or stream
- including its standard toolbar

Adding Rules



Location	Kind	Set In Stream
\.\	incl	kestrel

In each depot, there is one hard-coded base rule: the depot's base stream has an include rule that specifies the depot's top-level (root) directory. This rule makes the depot's entire directory hierarchy visible in the depot's base stream.

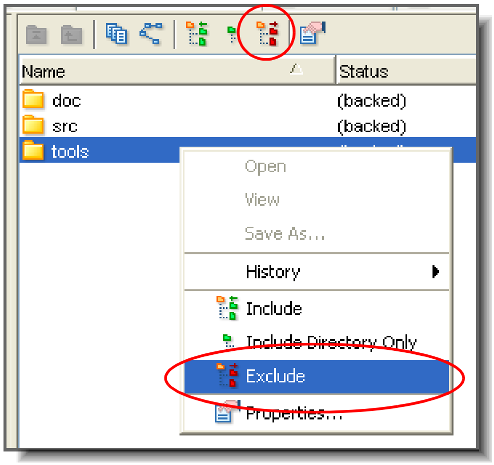
Any number of rules can be added. Each rule applies to a particular pathname within the depot's directory hierarchy ("Location"), and applies at a particular level in the depot's stream hierarchy ("Set in Stream"). A rule set in a dynamic stream gets inherited by lower-level streams; but a rule for the *same location* in a lower-level stream or workspace overrides a rule in a higher-level stream.

Example: Excluding a Directory

To exclude a particular directory from the current workspace or stream, add a new rule for it:

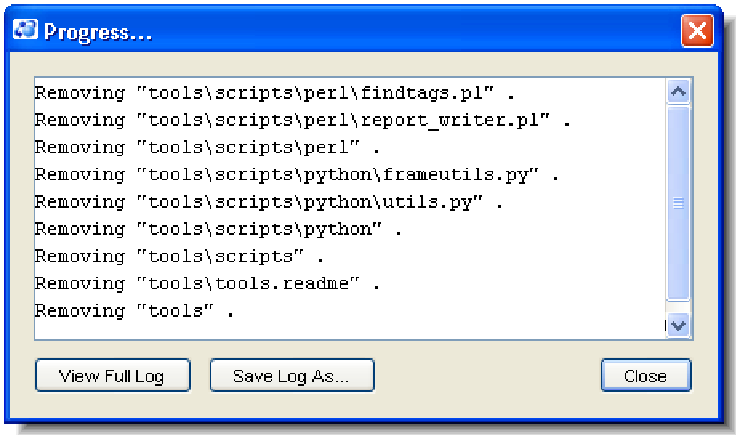
1. Make the directory appear in the details pane, by going to its parent directory in the navigation pane.

2. Select the directory to be excluded and click the **Exclude** button in the toolbar. Alternatively, right-click the directory and select **Exclude** from the context menu.



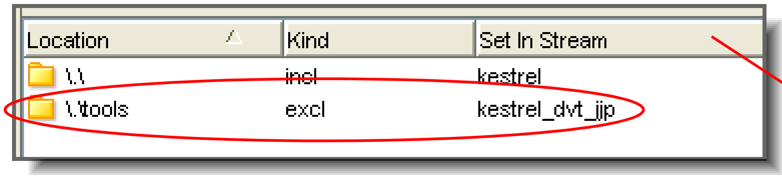
Alternative: if a rule already exists for the directory — set at the current level in the stream hierarchy or at a higher level — you can invoke **Exclude** from the context menu of that rule in the rules pane.

The entire directory tree below the specified directory is removed from the workspace or stream.



```
Removing "tools\scripts\perl\findtags.pl" .
Removing "tools\scripts\perl\report_writer.pl" .
Removing "tools\scripts\perl" .
Removing "tools\scripts\python\frameutils.py" .
Removing "tools\scripts\python\utils.py" .
Removing "tools\scripts\python" .
Removing "tools\scripts" .
Removing "tools\tools.readme" .
Removing "tools" .
```

The new exclude rule now appears in the rules pane:



Location	Kind	Set In Stream
\.\	incl	kestrel
\.\tools	excl	kestrel_dvt_jjp

this column appears only if **Show All** is checked

If you're in a dynamic stream, AccuRev reminds you that elements won't be removed from workspaces below that stream until they are **Update**'d. But keep in mind that element exclusion is instantly inherited by *streams* below that stream.

> Note: if you want to remove some, but not all, of a directory tree from a workspace or stream, you must use an **Include Directory Only** rule, not an **Exclude** rule. See *Using the "Directory Only" Form of an Include Rule* on page 66.

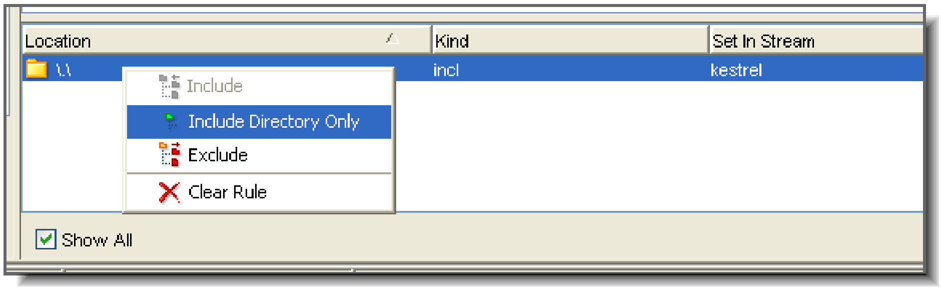
Example: Emptying Out a Workspace

If you're accustomed to working in a sparse workspace, you'll may want to do the include/exclude equivalent in a new workspace:

- remove all elements from the workspace
- add back to the workspace just the elements that you want to appear in it

The first step is accomplished by setting an **Include Directory Only** rule for the depot's top-level directory. Since the top-level directory never appears in the File Browser's details pane, you must invoke the command from the rules pane, not the details pane:

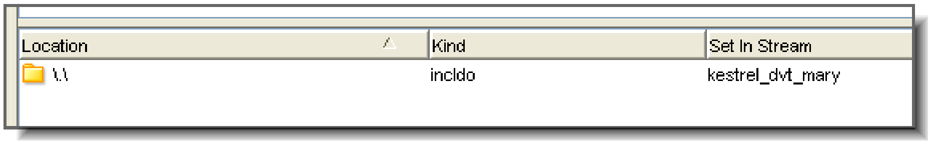
1. Make sure that **Show All** is checked in the rules pane, causing the depot's base rule to be displayed.
2. Right-click the base rule, and select **Include Directory Only**.



Location	Kind	Set In Stream
\.\	incl	kestrel

☑ Show All

This creates a new rule, at the workspace level, for the top-level directory. This rule overrides, and replaces in the rules pane, the rule you just right-clicked:



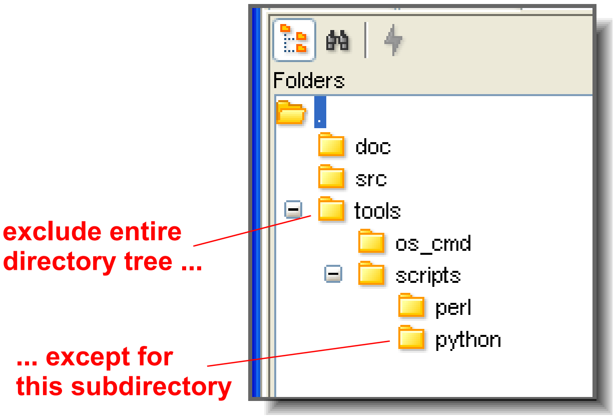
Location	Kind	Set In Stream
\.\	incldo	kestrel_dvt_mary

If you subsequently remove this new rule (see *Removing Rules* below), the overridden rule will reappear in the rules pane, and will reapply to your workspace.

Using the "Directory Only" Form of an Include Rule

Sometimes, you want to exclude most, but not all, of a directory tree from a workspace or stream. For example, you might want to exclude the entire **tools** directory tree, except for subdirectory **python**. This is easy to accomplish, since the depot's entire directory hierarchy is always visible in include/exclude mode:

1. Add an **Include Directory Only** rule for the **tools** directory.

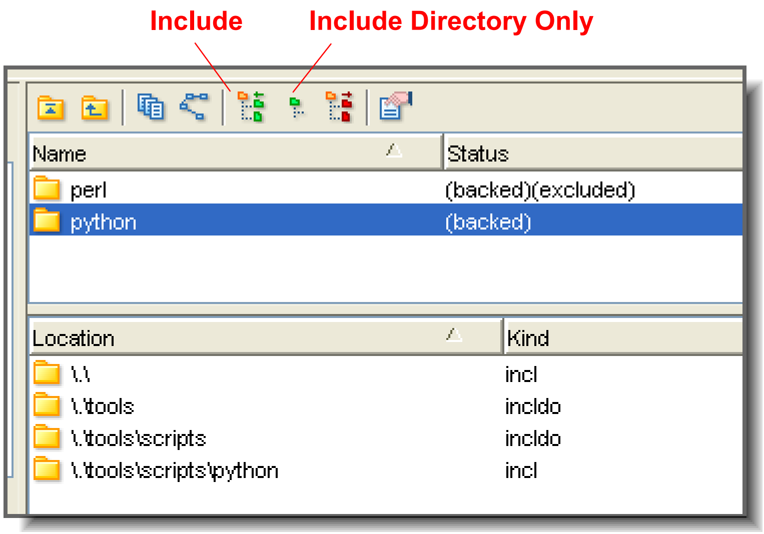


Folders
.
doc
src
tools
os_cmd
scripts
perl
python
exclude entire directory tree ...
... except for this subdirectory

2. Add an **Include Directory Only** rule for the **scripts** directory.

3. Add an **Include** rule for the **python** directory.

Here's the resulting details pane display:



Name	Status
perl	(backed)(excluded)
python	(backed)

Location	Kind
\.\	incl
\.\tools	incldo
\.\tools\scripts	incldo
\.\tools\scripts\python	incl

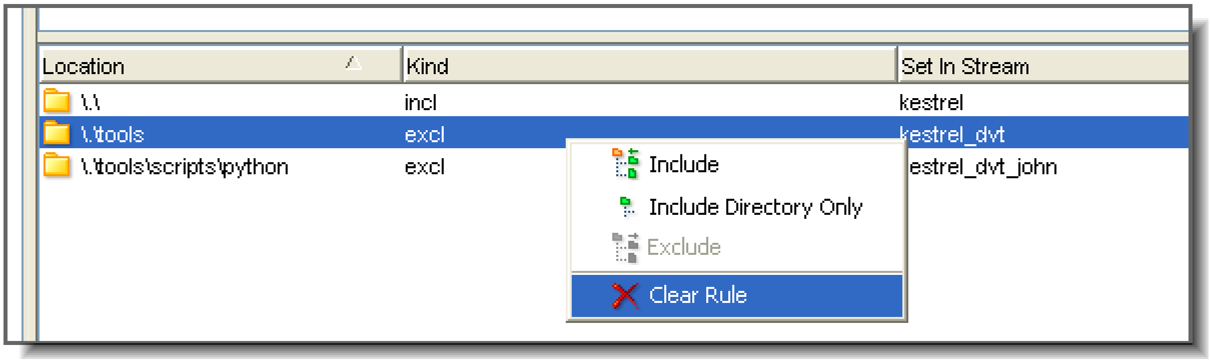
Note that the **Include Directory Only** command actually constitutes both an inclusion and an exclusion: "include the specified directory, but exclude everything under that directory". Subsequent **Include Directory Only** commands can override, in part, the "exclude everything under" portion of the command.

Removing Rules

To remove an existing rule that appears in the rules pane:

1. Make sure that the File Browser tab shows the stream or workspace in which the rule was explicitly set (**Set In Stream** column). If necessary, open a new File Browser on that stream or workspace.

2. Right-click the rule, and select **Clear Rule** from the context menu.



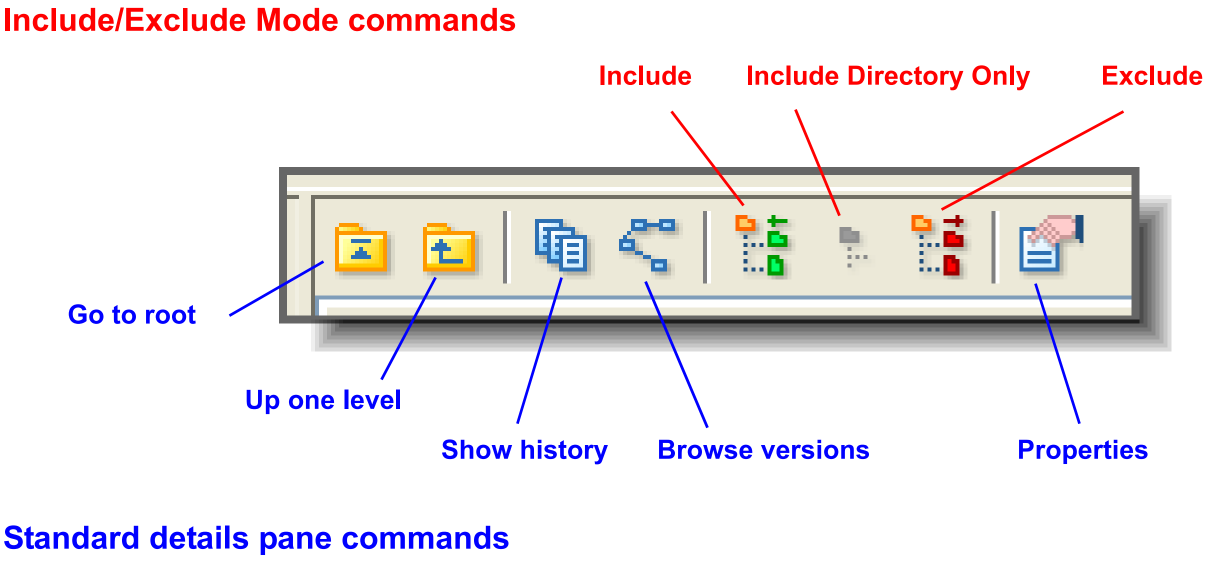
Location	Kind	Set In Stream
\.\	incl	kestrel
\.\tools	excl	kestrel_dvt
\.\tools\scripts\python	excl	estrel_dvt_john

When you remove a rule from a stream, the effect is immediate on the stream itself and on streams below it. The effect does not take place on workspaces below the stream until they are **Update**'d.

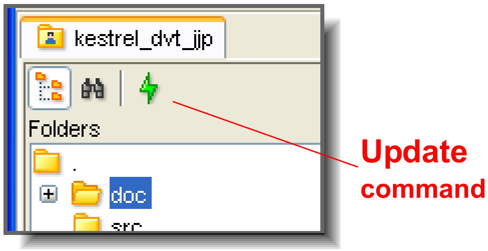
When you remove a rule from a workspace, the effect is immediate on the workspace itself: files are copied into the workspace tree if you remove an exclude rule; files are deleted from the workspace tree if you remove an include rule.

Details Pane Toolbar in Include/Exclude Mode

The File Browser's details pane has its own toolbar (see *Working in the Details Pane* on page 36). When you switch to Include/Exclude mode, the toolbar changes: most of the buttons for AccuRev's version-control commands — **Keep**, **Promote**, **Merge**, etc.— disappear. Instead, buttons for the **Include**, **Include Directory Only**, and **Exclude** commands appear. A few of the standard toolbar buttons remain, to aid you in navigating the depot and determining element history.

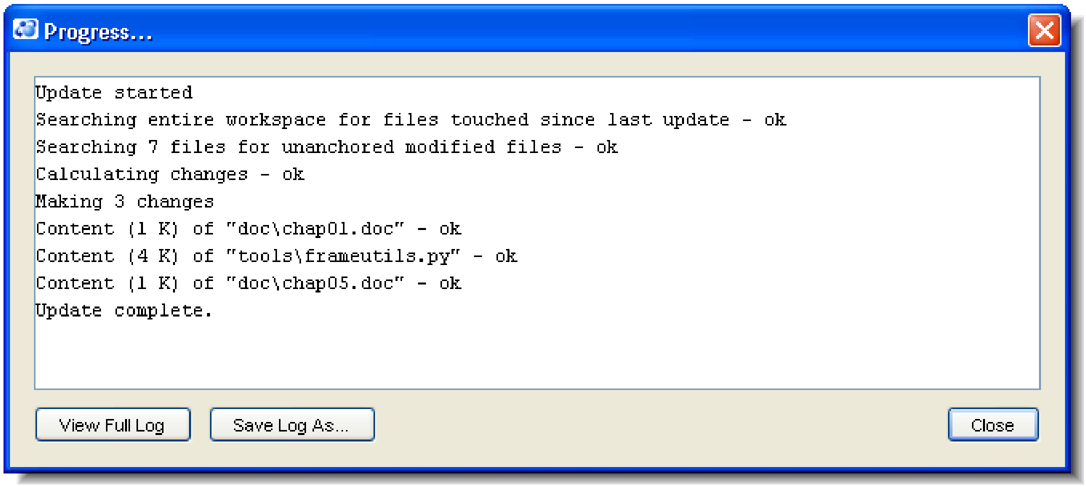


Updating the Workspace



The principal purpose of an AccuRev workspace is to provide an *isolated* location for performing development tasks. The changes you make do not affect your colleagues until you decide to make them public, using the **Promote** command. Likewise, the changes that others make do not immediately affect your workspace. You must execute an **Update** command to bring versions created (and then promoted) by your colleagues into your workspace.

You invoke the **Update** command from the navigation pane's toolbar. The command works both in Directory View and in Searches View. A progress window appears, showing exactly what is being updated.



```
Update started
Searching entire workspace for files touched since last update - ok
Searching 7 files for unanchored modified files - ok
Calculating changes - ok
Making 3 changes
Content (1 K) of "doc\chap01.doc" - ok
Content (4 K) of "tools\frameutils.py" - ok
Content (1 K) of "doc\chap05.doc" - ok
Update complete.
```

The following sections describe the workspace update process in detail.

Kinds of Changes Involved in an Update

At its simplest, an update just copies versions of some files from your workspace's backing stream (its parent in the stream hierarchy) into the workspace. For example, your colleagues may have edited the contents of files **colors.java** and **main_menu.java**, created new versions of them in their workspaces, then promoted the versions to the common backing stream. When you invoke **Update**, the new versions of those two files are copied from the depot to your workspace, overwriting the older versions of the file.

In addition to incorporating such content changes into your workspace, **Update** incorporates namespace changes:

- renaming of a file
- moving of a file to another directory
- creation of new files and directories
- deletion (defuncting) of existing files and directories

AccuRev tracks namespace changes in the same way as content changes — by saving each change as a new version. If you **Rename** file **colors.java** to **colours.java**, the change is recorded as a new version of the file. Changing the name again, to **hues.java**, creates another new version.

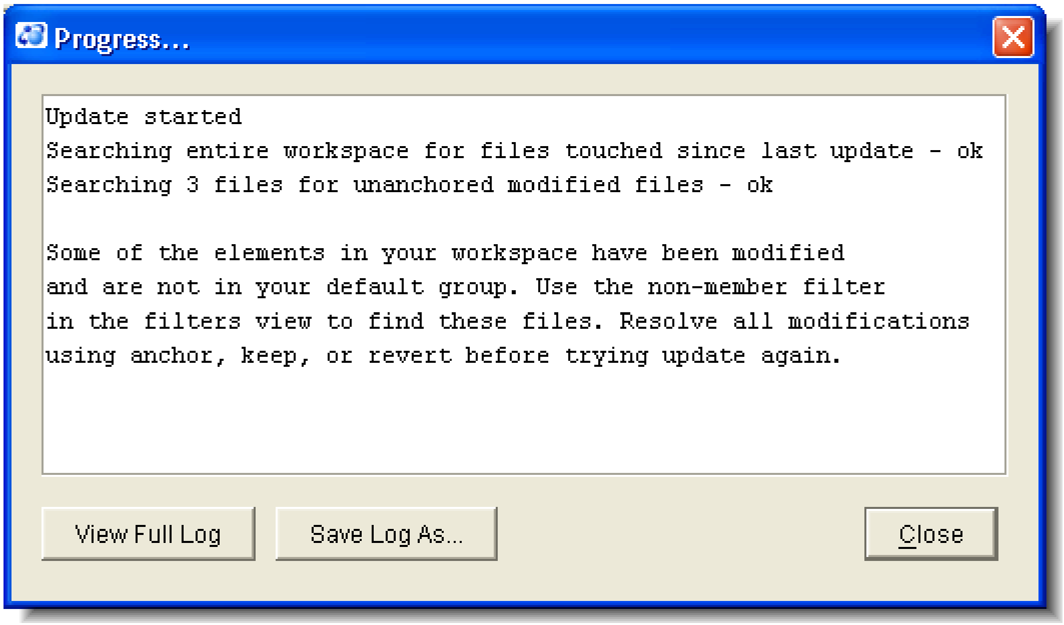
Deciding Which Files to Update

Roughly speaking, **Update** partitions the entire collection of files in your workspace into two categories: (1) files that you're currently working on, and (2) all the others. The basic strategy is for **Update** to leave the files you're working on undisturbed, and to copy any new versions of the other files into the workspace. This enables the workspace to provide the safety of isolation, while still "keeping in touch" with other users' progress.

AccuRev uses the concept of default group to keep track of the files you're working on. (We often use the more informal term "active files" or "active elements" to describe the members of the default group.) Files are placed in the default group when you process them with such commands as **Keep** and **Rename**. The **Update** command won't bring a new version of any file (or directory) element into your workspace if the element is in the workspace's default group.

Modified Files. You might think that all elements *not* in the workspace's default group would be candidates for updating. Well, almost ... one of AccuRev's most useful features introduces a complexity here. Other version-control systems force you to perform a "check out" operation on a file before editing it; with AccuRev, you can edit any file at any time. But this means that you might be actively working on a file that is not yet a member of the workspace's default group — because you haven't yet issued a **Keep** on that file. Such files have the status "modified, but not in default group". It would be wrong for **Update** to overwrite such files — you don't want this command to clobber changes you just made to a file!

Update handles this situation by refusing to proceed if the workspace contains any files whose status is "modified, but not in default group".



```
Update started
Searching entire workspace for files touched since last update - ok
Searching 3 files for unanchored modified files - ok

Some of the elements in your workspace have been modified
and are not in your default group. Use the non-member filter
in the filters view to find these files. Resolve all modifications
using anchor, keep, or revert before trying update again.
```

Note: you *can* use AccuRev is a way that resembles those other version-control systems — see *Serial Development through Exclusive File Locking* on page 25 of the *AccuRev Concepts Manual*. A side-effect of this feature is that files don't get into the "modified, but not in default group" state.

External Files. A second factor that makes "candidate for updating" not exactly equivalent to "not in the default group". A workspace can contain external files (and directories), which are not under version control. This typically includes text-editor backup files, results of software builds, mail messages, etc. External files won't ever be overwritten by an **Update**, since there are no versions of these files in the depot to be brought into the workspace. But there's a performance issue here: **Update** can spend a considerable amount of time analyzing the workspace's external files, before deciding that it doesn't need to worry about them.

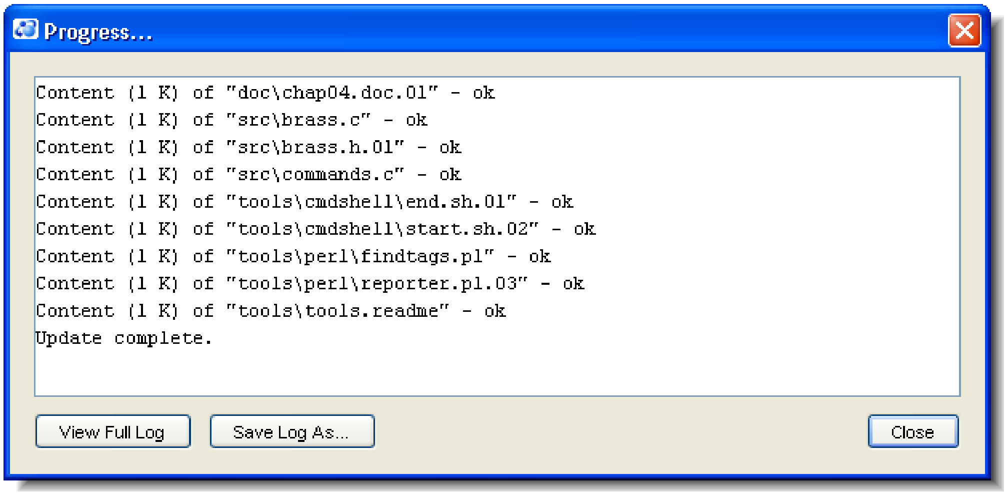
The possible existence of modified files and/or external files in the workspace means that **Update** must examine the entire contents of the workspace before proceeding to update it. This is exactly the same as the situation encountered by the File Browser in Searches View. Accordingly, **Update** performance is enhanced through use of the same optimizations described in section *Search Performance Optimizations* on page 59:

- A file is ignored if its pathname matches any of the patterns specified by environment variable ACCUREV_IGNORE_ELEMS.
- A file is ignored if its timestamp predates the time the workspace was last updated (the workspace's update time).

Performing the Update

After determining the set of elements that are candidates for updating, **Update** determines the subset that actually *need* updating: the elements for which a more recent version exists in the backing stream. This step is efficient and speedy — **Update** need only consider the elements that were involved in transactions recorded since the workspace's previous update. Only these transactions can contain changes that have not yet been incorporated into the workspace.

Update then replaces the workspace's versions with the newer backing-stream versions, logging its work in a separate window:



```
Content (1 K) of "doc\chap04.doc.01" - ok
Content (1 K) of "src\brass.c" - ok
Content (1 K) of "src\brass.h.01" - ok
Content (1 K) of "src\commands.c" - ok
Content (1 K) of "tools\cmdshell\end.sh.01" - ok
Content (1 K) of "tools\cmdshell\start.sh.02" - ok
Content (1 K) of "tools\perl\findtags.pl" - ok
Content (1 K) of "tools\perl\reporter.pl.03" - ok
Content (1 K) of "tools\tools.readme" - ok
Update complete.
```

If there are many elements to be updated, the logging information scrolls out of view. You can click the **View Full Log** button to open a text editor on the entire log, or save it to a text file with the **Save Log As** button.

Recording the Update

After it has completed the updating of versions in the workspace, **Update** updates two important workspace parameters:

- update time — the time at which the workspace was last updated. The more recent this value, the more effective the timestamp optimization invoked both by **Update** and by the File Browser in searches view. (See page 59.)

- update level (short for “update transaction level”) — the number of the depot’s most recent transaction. After an update, the workspace is “up to date as of transaction *N*”; *N* is the workspace’s update level. The higher this value, the fewer transactions your next invocation of **Update** will need to examine, in order to determine which elements need to be updated.

The StreamBrowser

One of AccuRev's most powerful tools is the StreamBrowser®. It enables graphical control over the entire configuration management environment, in a way that is simple, flexible, and powerful. The StreamBrowser enables you to view and manipulate all the streams in a depot. This includes the streams that are built into workspaces; it also includes snapshots, which are "frozen streams".

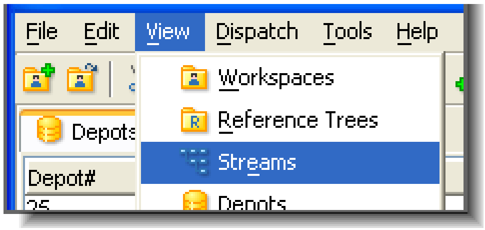
Prior to Version 3.5, the View menu included two commands:

- The **Streams** command opened a tab containing a table that listed the current depot's streams, snapshots, and workspaces.
- The **Stream Browser** command opened a tab containing a graphical display of the current depot's entire stream hierarchy.

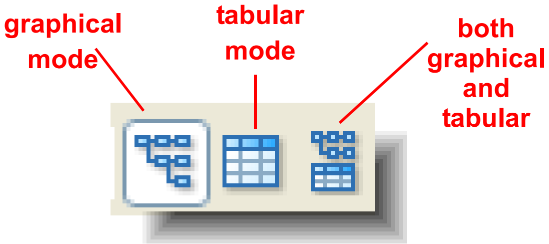
Starting in Version 3.5, the tab opened by the **View Streams** command offers both display formats: tabular and graphical. The **Stream Browser** command has been retired.

Opening a StreamBrowser Tab

To open a StreamBrowser tab, select **View > Streams** from the command menu. You can also click the **View Streams** toolbar button, or select **View > Streams** from the context menu of a depot on a Depots tab. The StreamBrowser tab displays the streams of the current depot (the one whose name is displayed at the bottom of the GUI window). If there is no current depot, AccuRev prompts you to select one.



The StreamBrowser offers three displays modes, which you select using toolbar buttons:

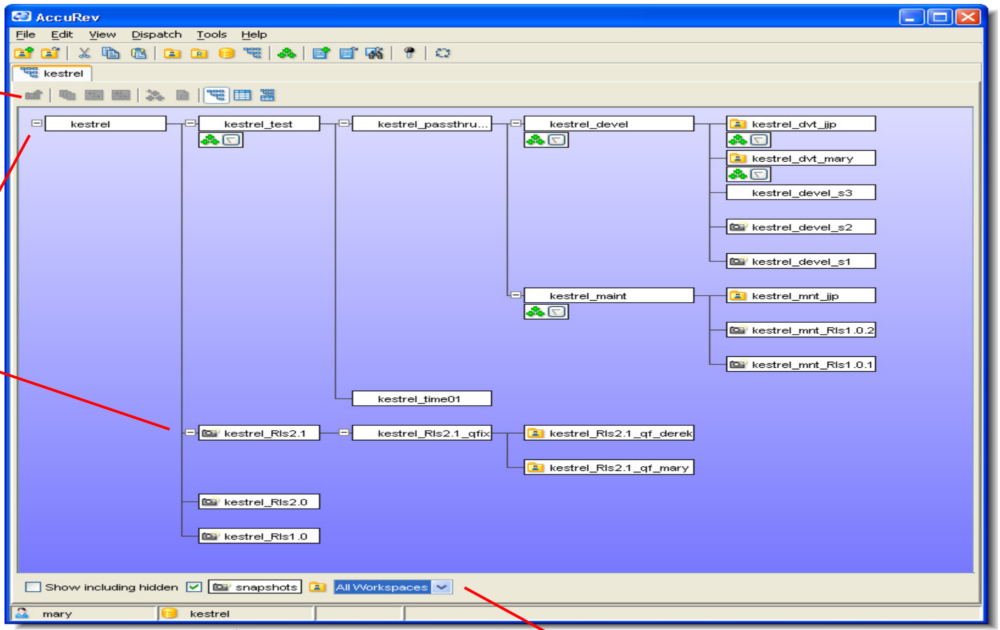


- Graphical display of the depot's stream hierarchy.
- Tabular display of the depot's streams
- Both of the above: the tab is divided into a graphical pane (above) and a tabular pane (below). A movable separator bar enables you to adjust the allocation of screen space to the panes.

You can have any number of StreamBrowser tabs open at the same time, each for a different depot. You can also open multiple StreamBrowser tabs on the *same* depot.(Be careful — this can be useful, but can also be a bit confusing if you forget which tab is which.)

Graphical StreamBrowser Display

The graphical display of a depot's stream hierarchy is organized as follows:



StreamBrowser toolbar

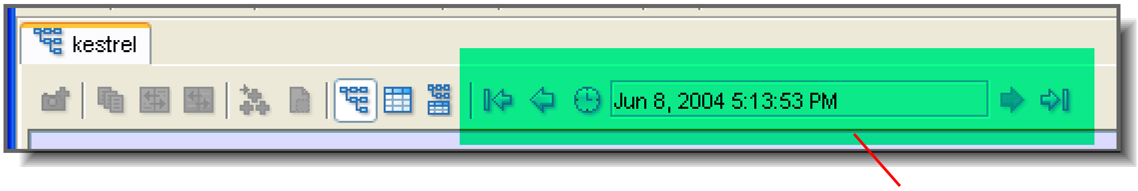
Expand/collapse controls display sub-hierarchies

depot whose stream hierarchy is displayed

Checkboxes and listbox control display of workspaces, snapshots, and inactive data structures

- The depot's root stream (top-level) is at the left edge.
- A given stream's children appear to its right; the children are arranged vertically, in this order:
  - Workspaces, in alphabetical order
  - Dynamic streams with no basis time, ordered by creation time (most recent first)
  - Dynamic Stream / Snapshot with basis times, ordered by basis time (most recent first)

If you run the command **File > Preferences > Enable Stream Browser History** before opening the StreamBrowser tab, a set of history controls are added to the toolbar:



Sream Browser history controls

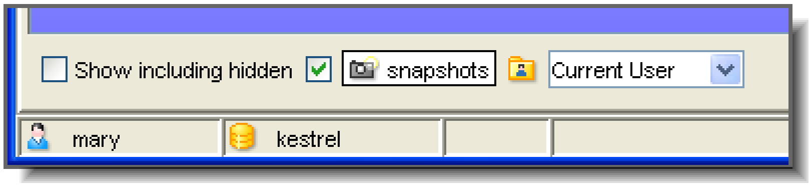
See *Stream History* on page 88.

Controlling the Display of Streams, Snapshots, and Workspaces

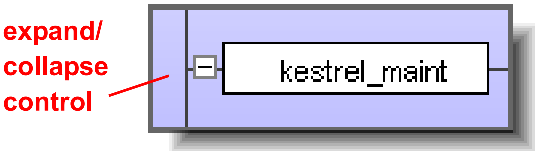
Initially, the StreamBrowser displays all of the depot's currently active dynamic streams, along with currently active snapshots. It doesn't display any workspace streams, nor any data structures that you have deactivated with the **Remove** command on its context menu.

Note: for simplicity, this chapter will usually refer to dynamic streams as “streams”, and to workspace streams as “workspaces”.

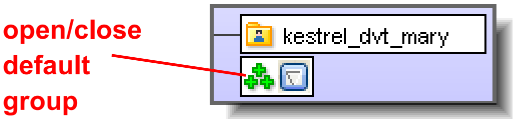
Using the checkboxes the listbox at the bottom of the StreamBrowser tab, you can:



- Reveal/hide the data structures that have been **Removed**. (The **Remove** command doesn’t actually delete anything from the depot; the data structure just becomes invisible and inactive.)
- Control the display of the depot’s snapshots (but perhaps not those that have been **Removed**).
- Control the display of the depot’s workspaces (but perhaps not those that have been **Removed**). The listbox contains the choices **Current User**, **All Workspaces**, **No Workspaces**, and the principal-name of each AccuRev user.



Each stream that has “children” (workspaces and/or snapshots) is displayed with an expand/collapse control. Collapsing causes the entire hierarchy below the stream to disappear from the screen. This affects the StreamBrowser display only. It does not affect the operation of the stream in any way.



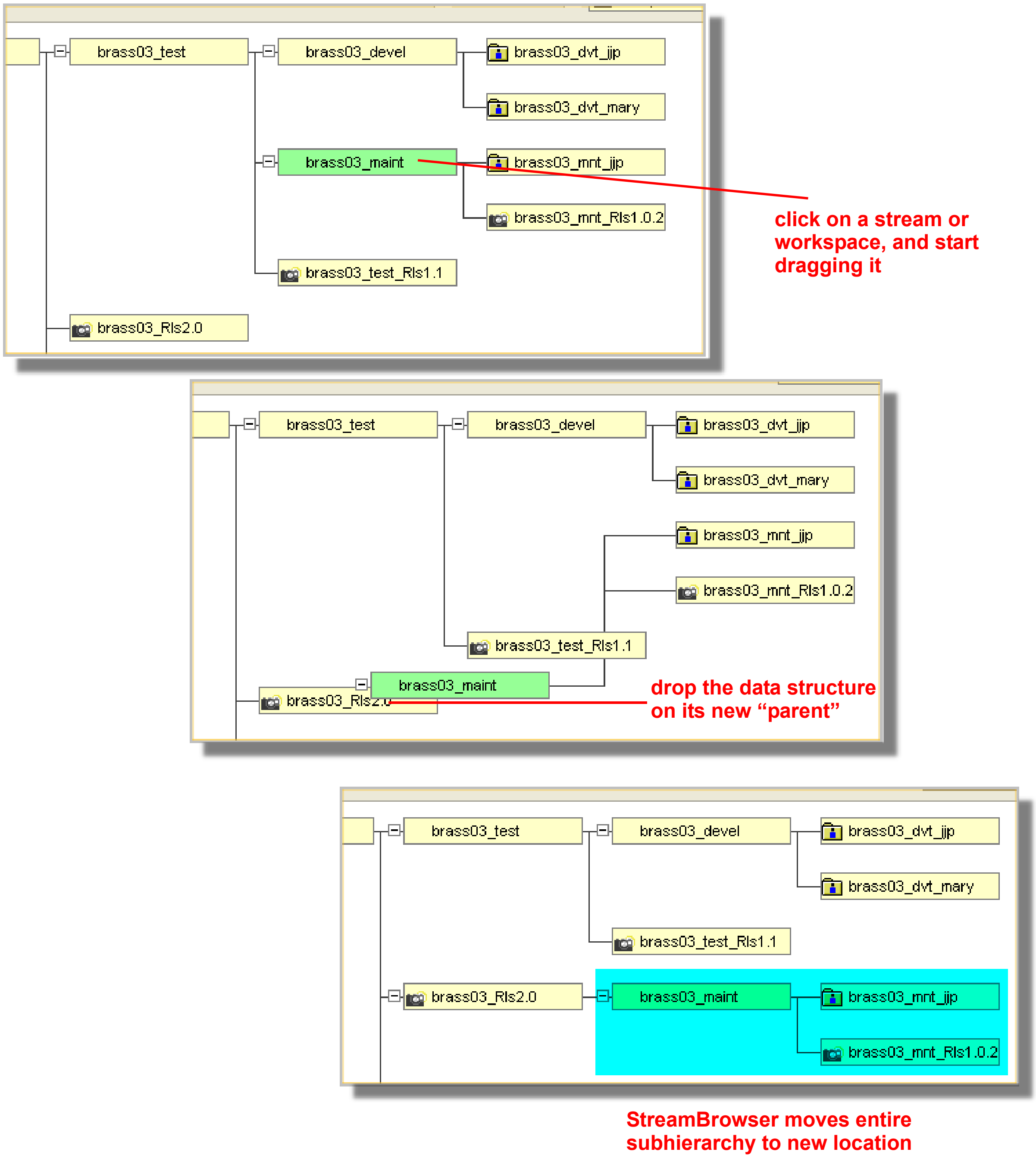
Each stream or workspace that has active elements — the default group of the stream or workspace contains one or more elements — is displayed with a special control that open and closes a window showing the default group. See *Displaying and Working with the Default Group* on page 78.

Manipulating the Stream Hierarchy

The StreamBrowser doesn’t just display a depot’s stream hierarchy — it’s also a tool for manipulating the hierarchy. You can add new streams, snapshots, and workspaces; you can rearrange the hierarchy, and you can remove (that is, hide) existing data structures.

Rearranging Streams and Workspaces

AccuRev lets you change the backing stream (parent stream) of any dynamic stream or workspace. The StreamBrowser makes it simple: you just drag-and-drop a stream or workspace from its current location in the hierarchy to its new parent. The entire subhierarchy moves to the new location:



click on a stream or workspace, and start dragging it

drop the data structure on its new “parent”

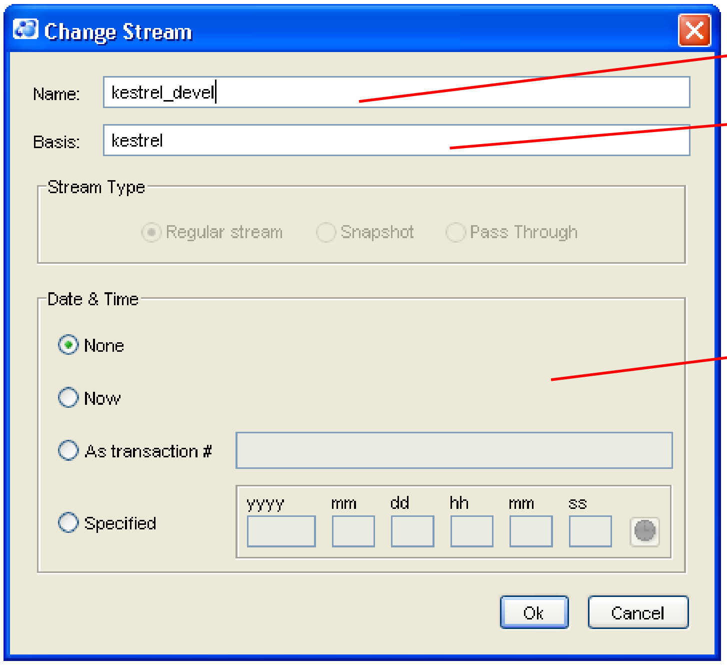
StreamBrowser moves entire subhierarchy to new location

Changing a data structure’s location in the stream hierarchy is called reparenting. You cannot reparent a snapshot; both the contents and the parentage of snapshot are fixed permanently.

> Note: after you change the location of a workspace, be sure to **Update** it. This ensures that the workspace contains the correct set of versions, many of which it will inherit from its new parent. Likewise, after changing the location of a stream, all workspaces in the subhierarchy below that stream should be **Update**d.

Reconfiguring a Stream

The drag-and-drop operation changes one property of a stream: its parent. You can also change a stream's properties using the **Change Stream** command on its context menu.



Change Stream

Name: kestrel_devel

Basis: kestrel

Stream Type: ● Regular stream ○ Snapshot ○ Pass Through

Date & Time: ● None ○ Now ○ As transaction # ○ Specified

rename the stream

change the stream's parent
(equivalent to drag-and-drop reparenting)

change the stream's basis time

Although you can give a new name to an existing stream, you cannot create then create a new stream with the old name. The old name remains associated with its stream. The only way to reuse a stream name is to completely remove the stream's depot from the AccuRev repository, using the AccuRev administration utility, **maintain**.

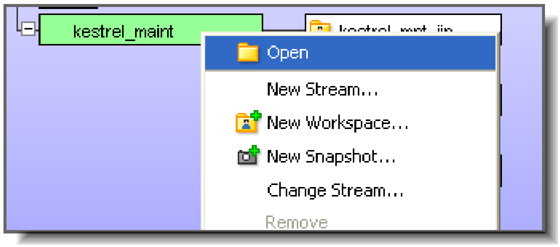
By default, a stream inherits all versions from its parent, no matter when those versions were created. If you assign a basis time to a stream, it inherits only those versions created before the specified point in time. The most common use of a basis time is to create a permanent, unchanging snapshot of a particular stream.

> Note: as of Version 3.5, there's another way to restrict which versions get inherited from the parent stream: the include/exclude facility. You must use the File Browser to access this facility.

Creating New Streams, Snapshots, and Workspaces

The context menu of any data structure in the StreamBrowser includes commands for creating new structures at that point in the hierarchy. **New Stream** and **New Snapshot** display dialog boxes similar to that of **Change Stream**, discussed above. The **New Workspace** command invokes a wizard that steps you through the process of defining a new workspace: you specify a name and a location on disk; you can also make some optional settings, such as controlling how line endings in text files are to be handled.

Exploring the Contents of Streams



You can view the entire contents of any stream, snapshot, or workspace by invoking a File Browser. Double-click the data structure, or right-click it and select **Open** from the context menu.

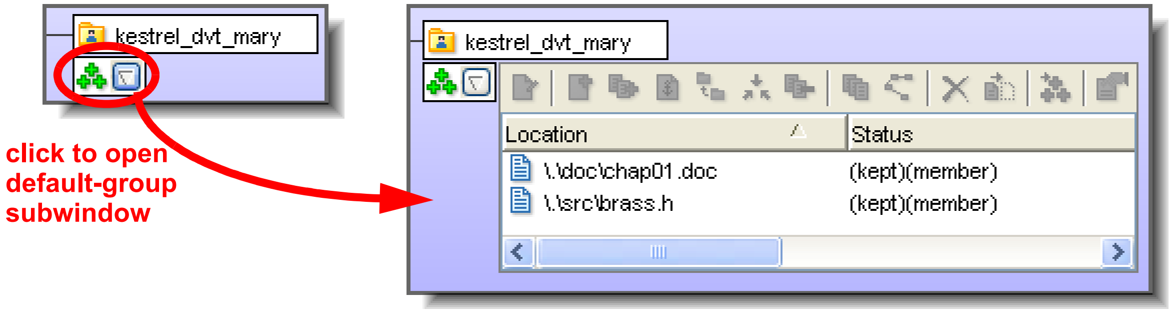
For user workspaces, the File Browser shows the status of the files in the user's disk storage (the workspace tree). If a particular workspace's storage is not available to you, the data display may be incomplete.

For dynamic streams and snapshots, the display is always complete: the File Browser displays configuration data that is stored in the depot in the AccuRev repository, not data from any individual user's workspace.

For more information, see *The File Browser* on page 29.

Displaying and Working with the Default Group

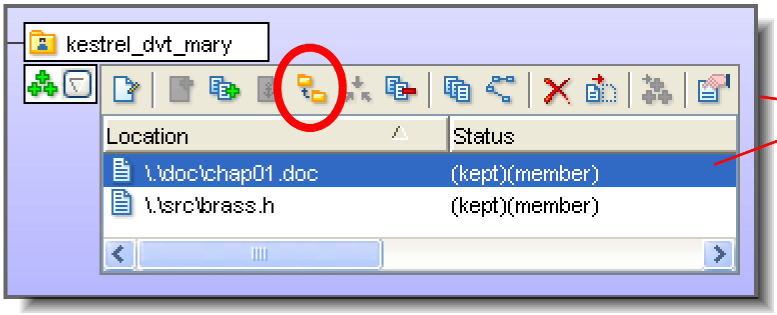
One of the most powerful features of the StreamBrowser is its ability to zero in on the "important" files — the ones that are under active development in a particular stream or workspace. These are the elements in the default group of that stream or workspace. Click the special control below a stream or workspace to open a subwindow that displays its default group. The control appears only if the default group is non-empty.



Location	Status
\.\doc\chap01.doc	(kept)(member)
\.\src\brass.h	(kept)(member)

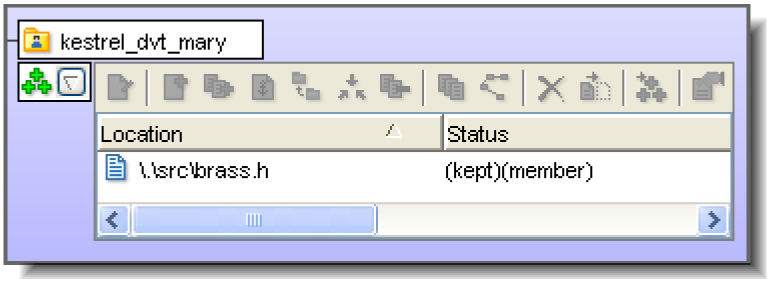
The default group subwindow closes when you click the control again — or when you open the default group of another data structure. (At most one default group subwindow can be open at any given time.)

The default group subwindow is small, but its scroll bars enable you to see any data that is not currently visible. It also has its own toolbar, similar to that of the File Browser's details pane. (Each element displayed has its own context menu, too.) This means you can perform many AccuRev operations in the default group subwindow, without having to invoke a File Browser at all. For example, you might promote file **chap01.doc** as follows:



Location	Status
\.\doc\chap01.doc	(kept)(member)
\.\src\brass.h	(kept)(member)

select chap01.doc and
click Promote toolbar button



Location	Status
\.\src\brass.h	(kept)(member)

after Promote command
completes, chap01.doc
is removed from the
default group

At this point, you could open a window on the default group of the parent stream, to verify that **chap01.doc** is now active in that stream.

For another file, you might decide that it shouldn't have been edited in that particular workspace. In that case, you could invoke the **Revert to Backed** command in the default group subwindow to deactivate the file and remove it from the default group.

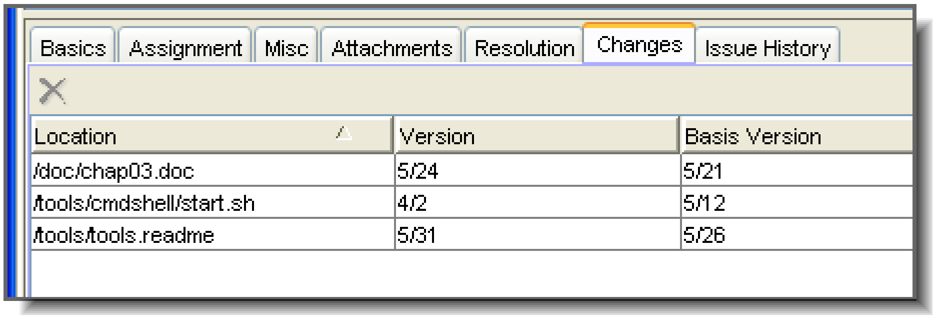
Viewing a Stream's Change Packages (Issues)

The preceding sections discuss how the StreamBrowser and File Browser enable you to see what individual versions are present — and perhaps active — in a stream. AccuRev also makes it easy to answer questions like these:

> "Are all the changes required to fix bug #4517 in this stream?"
>
> "Are any of the versions involved in the #4517 fix still active in this stream? (Or have all the changes already been promoted to a higher-level stream?)"
>
> "What new features are still under active development in this stream?"
>
> "The QA Group says they don't have all the Color Mixer changes for the upcoming release. Is that true?"

The key is to go beyond thinking of individual versions to considering collections of versions, called change packages. With AccuRev, change packages are implemented by Dispatch issue records. An issue record records the details of a bug or feature: its description, how important it is, who originated it, who's working on it, and so on. As of Version 3.5 (AccuRev Enterprise only), an issue record can also keep track of the changes that have been made to elements, in order to implement that particular bugfix or new feature.

For example, issue record #8 might contain a bug report, "Circles are not round". The bugfix involves changes to three elements:



Location	Version	Basis Version
/doc/chap03.doc	5/24	5/21
/tools/cmdshell/start.sh	4/2	5/12
/tools/tools.readme	5/31	5/26

change package
for issue record #8
contains three changes

When viewing the issue record through its edit form, go to the Changes tab to view the change package. In this example, the change package contains three changes:

- The changes that were made between version 5/21 and version 5/24 of element **chap03.doc**
- The changes that were made between version 5/12 and version 4/2 of element **start.sh**
- The changes that were made between 5/26 and version 5/31 of element **tools.readme**

For a discussion of how the Version and Basis Version specifications define a change as a series of versions, see *Change Packages and Integrations between Configuration Management and Issue Management* on page 39 of the *Dispatch Issue Management Manual*.

> Note: the Version and Basis Version columns always report real version-IDs — the IDs of versions as they were originally created in user's workspaces — not the virtual version-IDs acquired as versions are promoted up the stream hierarchy.

Change Packages "In" Streams

As the versions in a change package are promoted up the stream hierarchy, the change package itself implicitly moves up the hierarchy, also. Roughly speaking, a change package has risen to a certain level if all its constituent versions have risen to that level. More precisely, a change package is said to be "in" a particular stream if, for every element version listed in the *Version* column of the change package:

- the stream contains some version of the same element, and
- the stream's version is the same as — or is a descendant of — the change package's *Version*

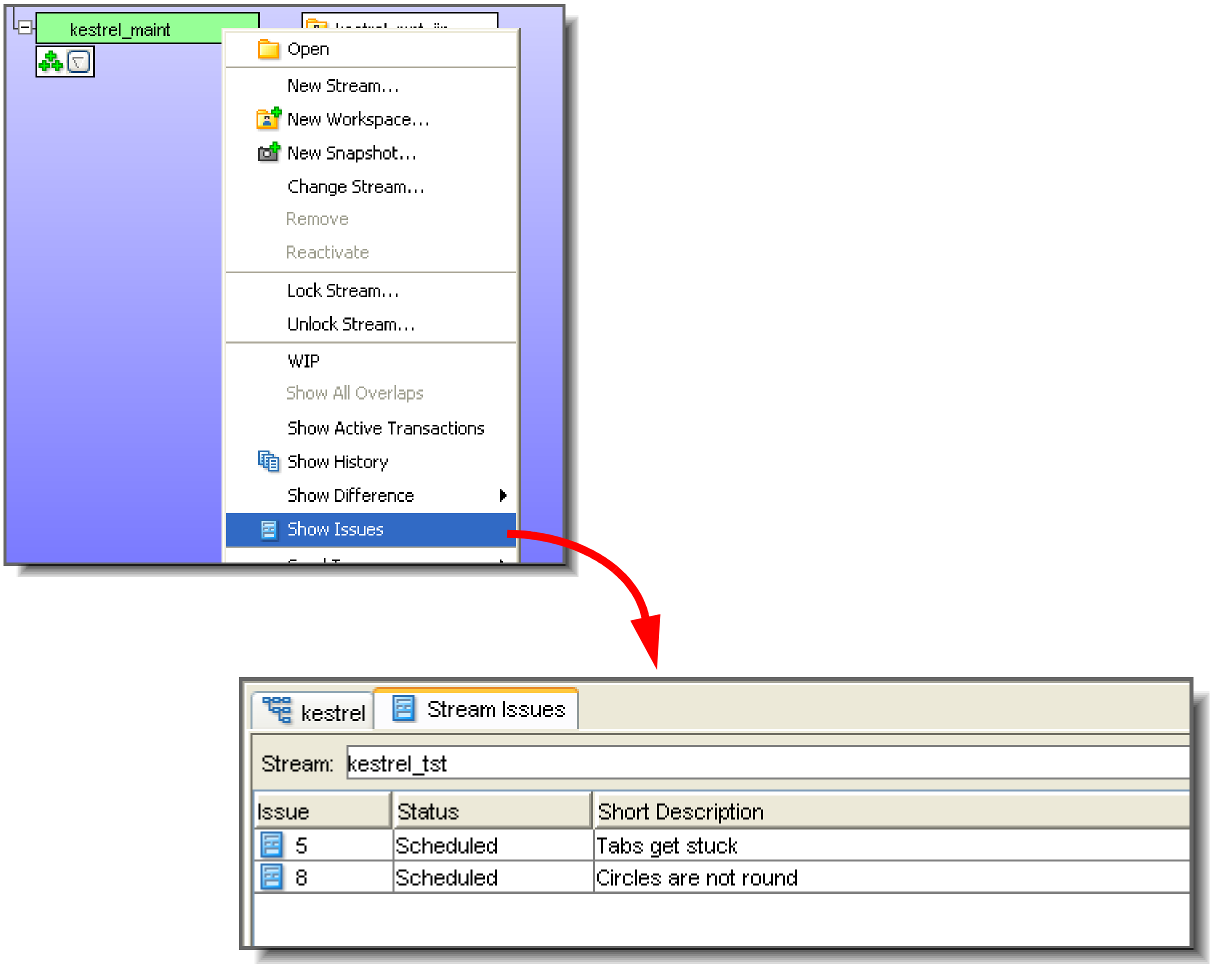
Two points need clarification here: why the "or is a descendant of" clause? And why don't we need to pay attention to the *Basis Version* column, too?

The change package shows that versions 5/9 through 5/13 of **tools.readme** were created to fix the "Circles are not round" problem. Suppose that after that, some user brought version 5/13 of **tools.readme** into her workspace with an update, then created a new version in her workspace to fix another problem — say, version 7/3. Is the "Circles are not round" fix to **tools.readme** in version 7/3? Yes — because the newer version is a descendant of version 5/13, it's safe to conclude that the changes in version 5/13 are still in version 7/3.

A similar ancestry analysis shows why we don't need to worry about the *Base Version* in a change-package entry. In this example, the change to **tools.readme** consists of the versions 5/9, 5/10, 5/11, 5/12, and 5/13. Version 5/13, the one listed in the *Version* column, still contains all the changes in versions 5/9, 5/10, 5/11, and 5/12 — because it’s a direct descendant of those versions. So if 5/13’s changes are “in” a particular stream, it’s safe to conclude that the changes in 5/9 through 5/12 are in that stream, too.

The ‘Show Issues’ Command

You can invoke the **Show Issues** command on any workspace, stream, or snapshot, using its StreamBrowser context menu. (We’ll use “stream” to cover all three data structures in the remainder of this section.) This command opens a Stream Issues tab, which displays the change packages — that is, the issue records — that are “in” the stream, according to the definition above.



Stream: kestrel_tst

Issue	Status	Short Description
5	Scheduled	Tabs get stuck
8	Scheduled	Circles are not round

The **Show Issues** command determines which fields of the issue records to display in the Stream Issues tab by consulting the schema for the issues database. To select the fields to be displayed:

1. Open the Schema Editor (**Dispatch > Schema Editor**).
2. Go to the Change Packages tab.
3. In the Change Package Results area, click the **Setup Column** button.

Two checkboxes on the Stream Issues tab enable you to refine the change-package analysis:

Show Incomplete

If **Show Incomplete** is cleared, the listing includes only change packages that are completely "in" the stream.

If the **Show Incomplete** is set, the listing includes only change packages that are partially "in" the stream — the definition above is satisfied for some, but not all, of the entries in the change package.

Note that the **Show Incomplete** checkbox is a toggle rather than a filter: setting and clearing the checkbox displays two non-overlapping collections of change packages.

Show Active

If **Show Active** is set, it filters the set of change packages displayed: a change package is included only if at least one of its versions is currently active in the stream is (is in the stream's default group). This filter helps you to concentrate on current programming efforts, rather than those that were completed long ago.

This filter has no effect when you invoke **Show Issues** on a snapshot, since no versions can be active in a snapshot.

Comparing the Contents of Streams

Another powerful StreamBrowser feature is its ability to quickly compare the contents of two streams. The **Show Difference > By Files** command does *not* directly compare the contents of files. Streams contain versions of elements. So a comparison of streams might determine that one stream has version 5 of file **foo.c**, while the other stream has version 13. (It's likely that the two versions have different contents, but the stream-comparison command doesn't "look inside" the versions to find out.)

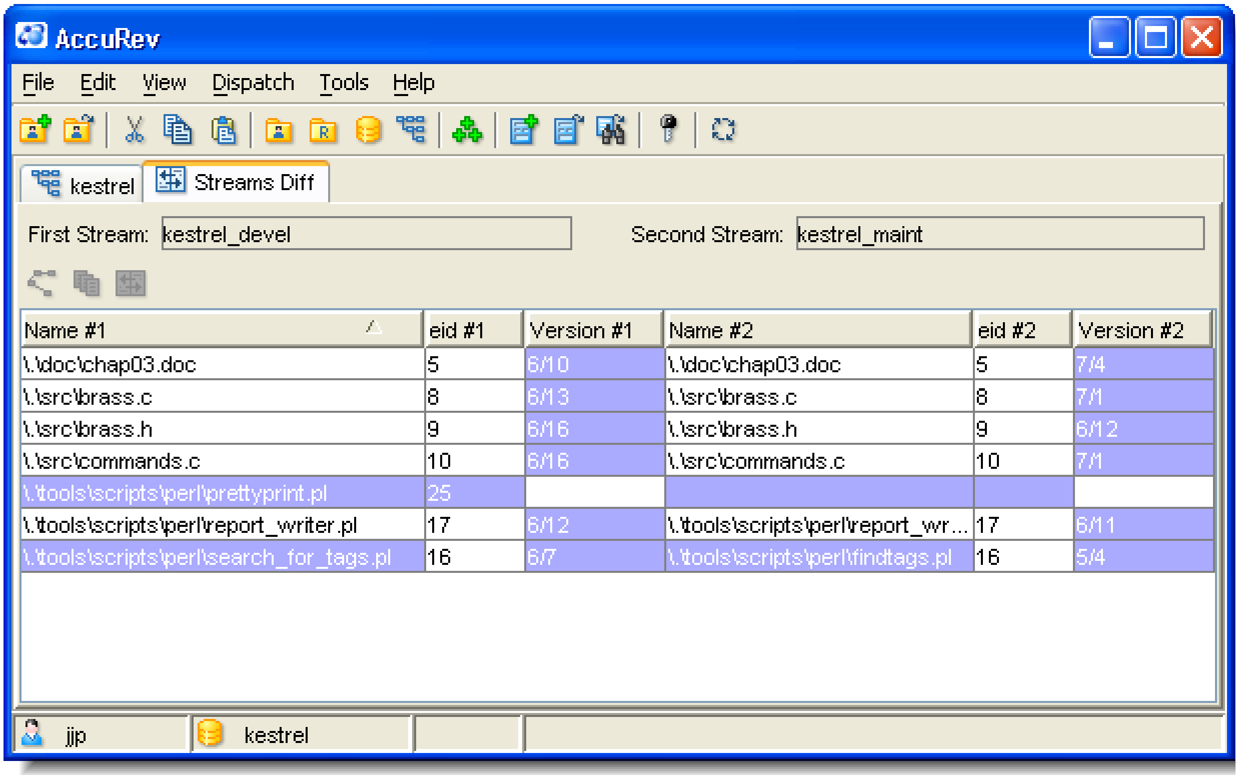
In addition, a stream comparison can report these differences:

- An element has been renamed in one or both streams.
- A new element has been added to one of the streams (**Add to Depot** command)
- An element has been removed from one of the streams (**Defunct** command).

To compare two streams:

1. Select one of the streams and click the **Show Files Difference** button on the StreamBrowser's toolbar. (Or select **Show Difference > By Files** from the stream's context menu.) The selected stream is highlighted in green, and the "show differences" icon is added to the mouse pointer.
2. Click the other stream.

This opens a **Streams Diff** tab:



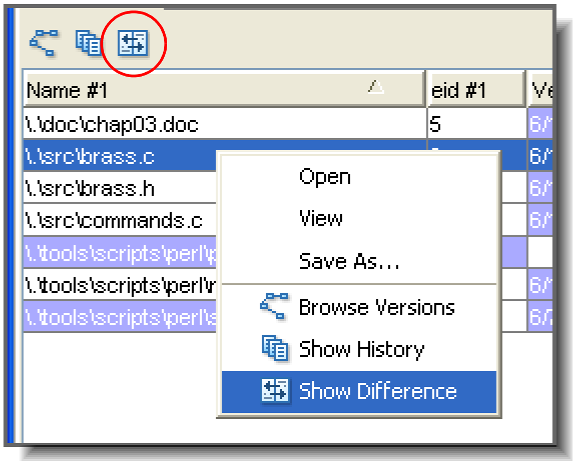
First Stream: kestrel_devel

Second Stream: kestrel_maint

Name #1	eid #1	Version #1	Name #2	eid #2	Version #2
\.\doc\chap03.doc	5	6/10	\.\doc\chap03.doc	5	7/4
\.\src\brass.c	8	6/13	\.\src\brass.c	8	7/1
\.\src\brass.h	9	6/16	\.\src\brass.h	9	6/12
\.\src\commands.c	10	6/16	\.\src\commands.c	10	7/1
\.\tools\scripts\perl\prettyprint.pl	25				
\.\tools\scripts\perl\report_writer.pl	17	6/12	\.\tools\scripts\perl\report_wr...	17	6/11
\.\tools\scripts\perl\search_for_tags.pl	16	6/7	\.\tools\scripts\perl\findtags.pl	16	5/4

The example above shows a total of seven differences between the two streams:

- The two streams have different versions of files **chap03.doc**, **report_writer.pl**, and three files in the **src** directory. To "drill down" and see what the changes are to any of these files, select the file and click the **Show Difference** toolbar button (or right-click the file and select **Show Difference**):
- A new element, **prettyprint.pl**, was added in stream **kestrel_devel**.
- The element that is named **findtags.pl** in stream **kestrel_maint** is named **search_for_tags.pl** in stream **kestrel_devel**.



The **Streams Diff** tab may not make it immediately clear which stream made the change that produced the difference. (Maybe a change was made in *both* streams.) To investigate, you can use the **View** command in the context menu to open a text editor on each version. (AccuRev notices whether you right-click on Version #1 or Version #2.) You can also use the context menu to open the Version Browser or History Browser on the element in question.

Comparing Streams Using Change Packages

Section *Viewing a Stream's Change Packages (Issues)* on page 79 describes how you can determine what change packages are "in" a workspace, stream, or snapshot. Similarly, you can use the **Show Difference > By Issues** command to cast the comparison of two workspaces/streams/snapshots in terms of change packages. (We'll use "stream" to cover all three data structures in the remainder of this section.)

This enables you to easily determine the answer to such questions as:

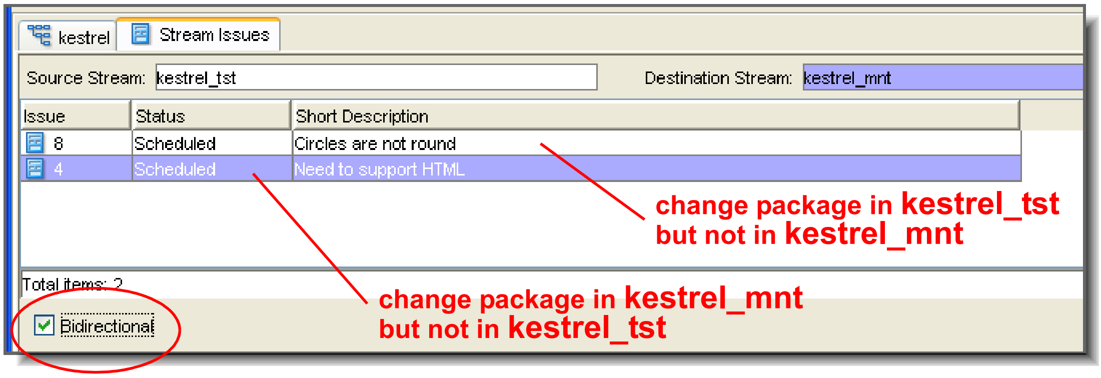
> "What features in the Integration stream have not yet been sent to the QA stream?"

To compare two streams by their change packages:

1. Select one of the streams and click the **Show Issues Difference** button on the StreamBrowser's toolbar. (Or select **Show Difference > By Issues** from the stream's context menu.) The selected stream is highlighted in green, and the "show differences" icon is added to the mouse pointer.
2. Click the other stream.

This opens a **Stream Issues** tab. Initially, this tab lists the change packages — that is, the issue records — that are in the first stream ("source"), but not in the second stream ("destination"). The names of the two streams appear at the top of the tab.

If you check **Bidirectional**, the listing is expanded to show the change packages that are in the second stream, but not in the first stream. Shading is added to help you to distinguish the two sets of change packages.



Source Stream: kestrel_tst

Destination Stream: kestrel_mnt

Issue	Status	Short Description
8	Scheduled	Circles are not round
4	Scheduled	Need to support HTML

Total items: 2

☑ Bidirectional

change package in **kestrel_tst** but not in **kestrel_mnt**

change package in **kestrel_mnt** but not in **kestrel_tst**

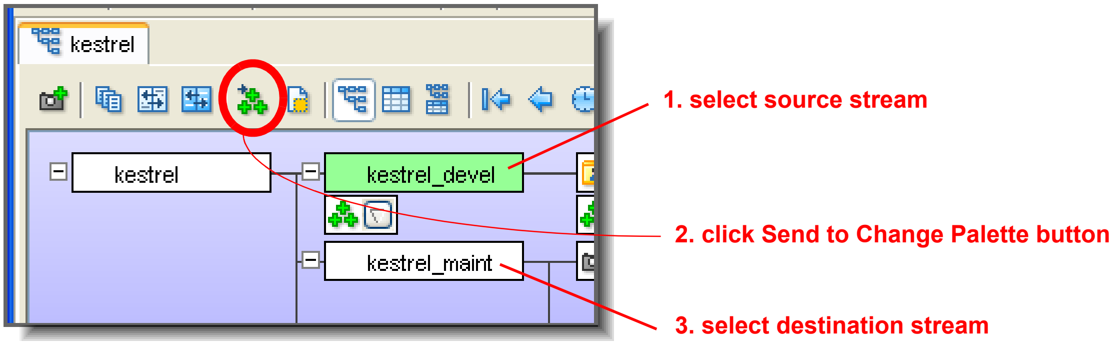
The **Show Difference > By Issues** command considers only change packages that are completely "in" the specified streams. It does not consider change packages that would be displayed by the **Show Issues** command with **Show Incomplete** selected.

Propagating Versions through the Stream Hierarchy

In *Displaying and Working with the Default Group* on page 78, we showed a simple example of how to use the StreamBrowser to promote files from a workspace or stream to its parent stream. More generally, AccuRev's Change Palette enables you to promote versions directly between any two dynamic streams. The StreamBrowser makes it particularly easy to “load” versions into the Change Palette, so that you can then promote them:

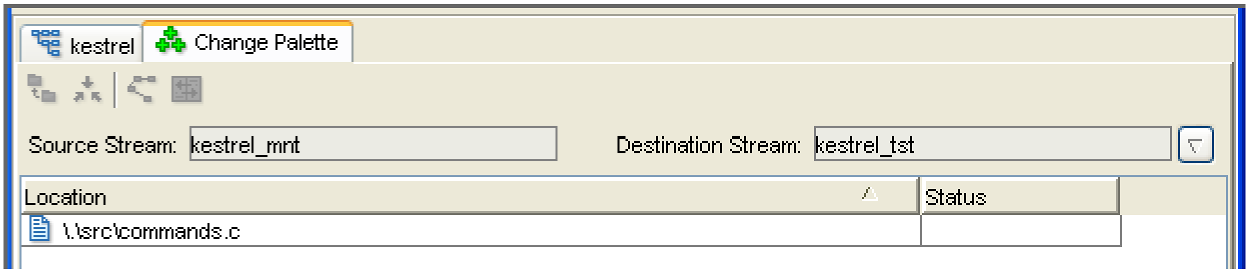
Propagating All of a Stream’s Changes

To propagate all of a stream’s changes to another stream:


kestrel
kestrel
kestrel_devel
kestrel_maint
1. select source stream
2. click Send to Change Palette button
3. select destination stream

1. Click on the source stream to select it.
2. Click the **Send to Change Palette** toolbar button.
3. Click on the destination stream.

This loads the Change Palette with all versions in the source stream containing changes that have not yet been sent to the destination stream:



Source Stream: kestrel_mnt

Destination Stream: kestrel_tst

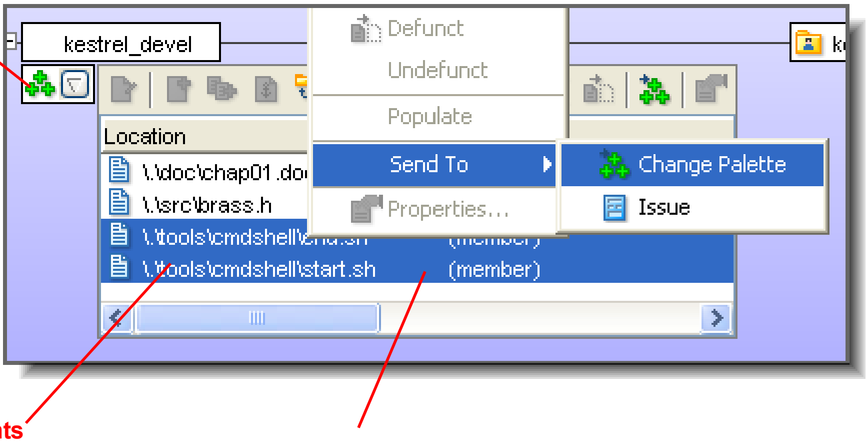
Location	Status
\.\src\commands.c	

Loading Selected Versions into the Change Palette

To load one or more versions in a stream’s default group into the Change Palette:

1. Open the stream’s default group subwindow.
2. Select one or more elements in the default group subwindow.
3. Select the **Send to Change Palette** command from the elements’ context menu.

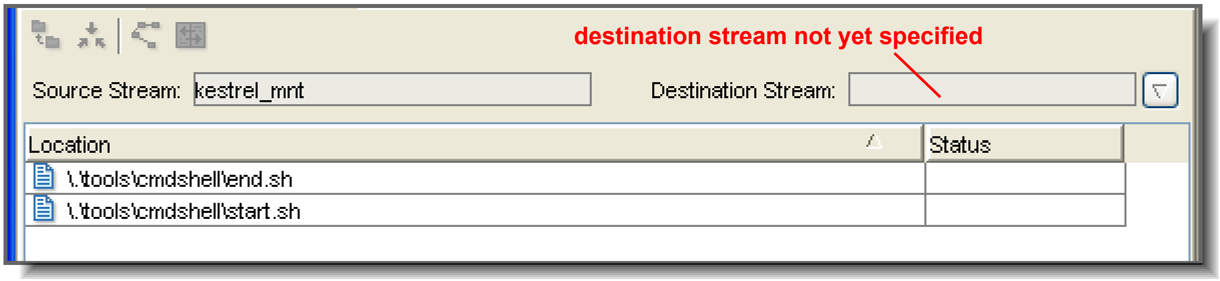
1. open the stream's default group subwindow



2. select elements

3. right-click and select Send To > Change Palette

This places the selected version(s) in the Change Palette. Note that this procedure specifies the source stream, but you still need to specify the destination stream.


destination stream not yet specified

Source Stream: kestrel_mnt

Destination Stream:

Location	Status
\.\tools\cmdshell\end.sh	
\.\tools\cmdshell\start.sh	

For details on using the Change Palette, see *Using the Change Palette* on page 129.

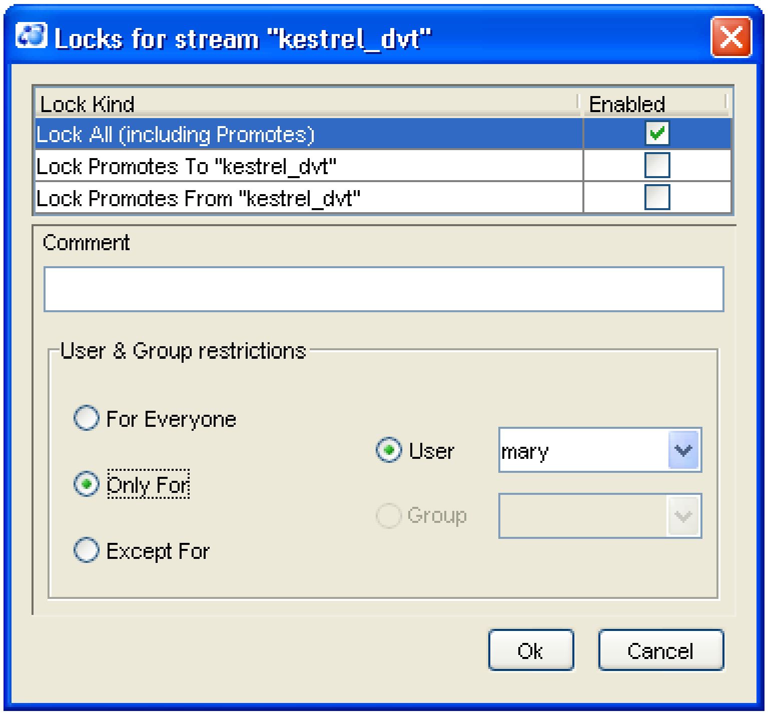
Additional Operations on Streams, Snapshots, and Workspaces

The context menu that appears when you right-click a stream, snapshot, or workspace also includes the following commands.

Locks ... (Lock/Unlock Stream)

(Dynamic stream only) Locking a stream disables various operations that modify the stream. The **Lock All** choice disables promotion of versions to/from the stream, disables include/exclude mode changes to the stream, and disables the **Change Stream** command for the stream. With **Lock Promotes To**, you can disable promotion to the stream; With **Lock Promotes From**, you can disable promotion from the stream.

You can restrict the effect of the lock to particular users or groups; alternatively, you can exempt particular users or groups from being affected by the lock.



Locks for stream "kestrel_dvt"

Lock Kind	Enabled
Lock All (including Promotes)	☑
Lock Promotes To "kestrel_dvt"	□
Lock Promotes From "kestrel_dvt"	□

Comment

User & Group restrictions

○ For Everyone
● Only For
○ Except For

● User: mary
○ Group

WIP (Work in Progress)

Displays the contents of the default group of every workspace backed by (directly below) the specified stream or snapshot.

Show All Overlaps

(Workspace only) For elements in the workspace's default group, display all overlaps with versions in all higher-level streams — not just the backing stream.

Show Active Transactions

(Dynamic stream only) Invoke a History Browser, containing the transactions that created the versions currently in the stream's default group.

Note: in certain cases, this command does not report transactions involving certain members of the stream's default group. If a version does not need to be promoted to its parent (because it, or a successor, already has been promoted), then the version is ignored by this command. This, in turn, can cause certain transactions not to be reported by this command.

Show Issues

Analyzes the change packages stored in the depot's Dispatch issues database. Reports on change packages that have been (partially or completely) incorporated into the stream.

Show Difference > By Issues

Prompts you to indicate another stream. Reports on differences between your stream and the other stream, in terms of change packages instead of individual elements.

Show History

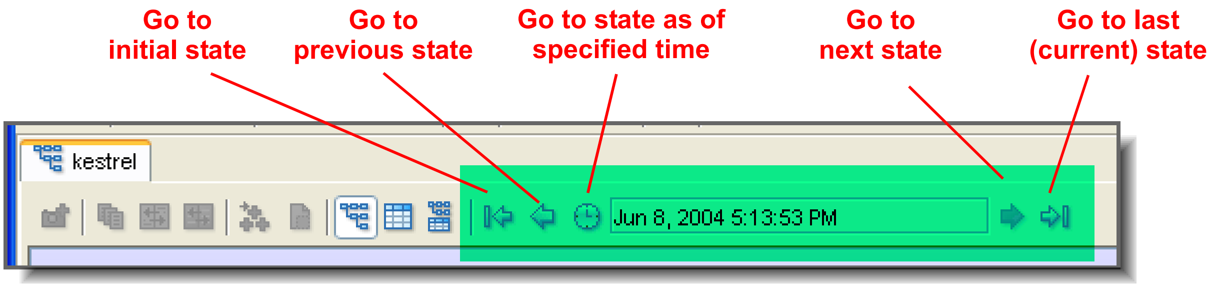
Invoke a History Browser, containing every transaction that created a version in the stream.

Show Patch List

Prompts you to indicate another stream. Lists all the individual versions that need to be patched to that other stream, in order to have the other stream include all the changes in your stream.

Stream History

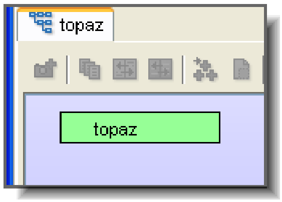
Like the individual files under version control, a depot's stream hierarchy undergoes change, too. Initially, the depot contains a single "base" stream. Thereafter, additional dynamic streams are created, along with snapshots and user workspaces. Each such change in the stream hierarchy creates a new "state" of the hierarchy. You can browse through all these states, like browsing through a slide show, using the History controls in the StreamBrowser's toolbar:



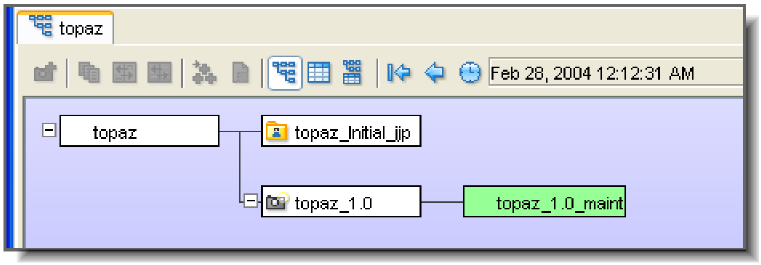
Note: at the time you open the StreamBrowser tab, you must already have enabled this feature for the current GUI session, using the command **File > Preferences > Enable Stream Browser History**. (This setting is not preserved from session to session.)

Here’s a simple example:

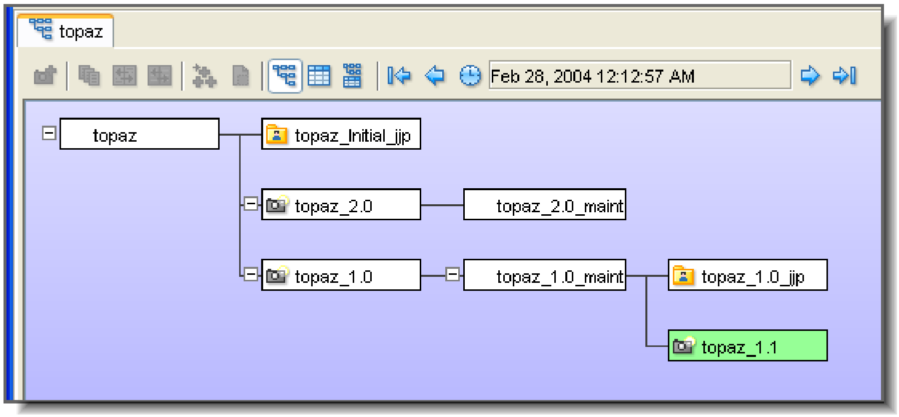
Click ⤒ to return to first state



Click ➡ a few times

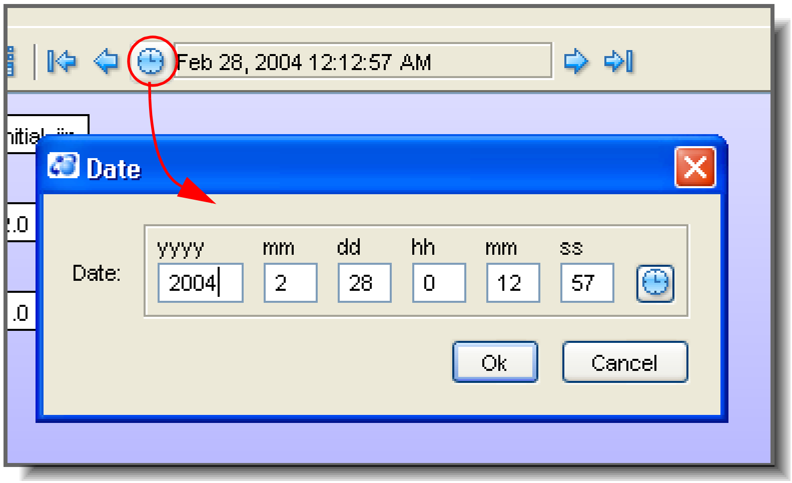


Click ➡ a few times more



This “slide show” is read-only. You cannot select or operate on any of the displayed streams until you return to the current state of the stream hierarchy.

When you click the 🕒 clock toolbar button, a dialog box appears in which you can specify a particular time. This causes the "slide show" to jump to the state that existed at that time.



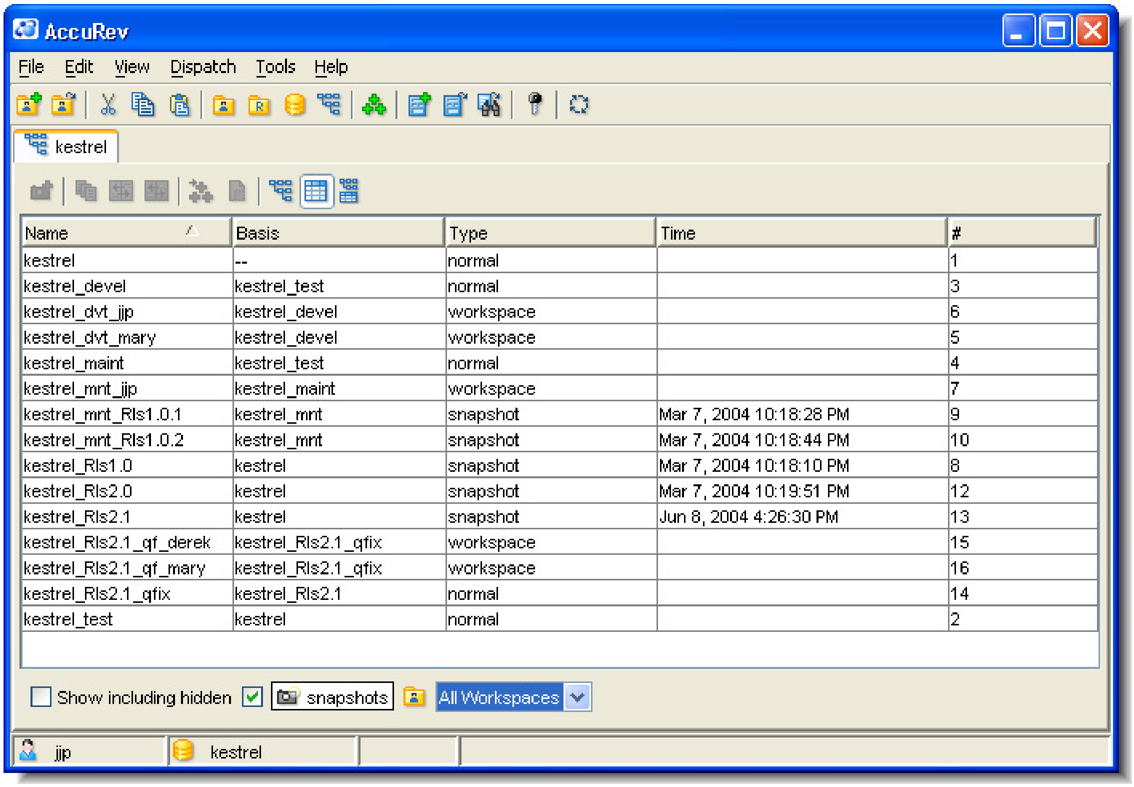
Feb 28, 2004 12:12:57 AM

Date

Date: yyyy 2004 mm 2 dd 28 hh 0 mm 12 ss 57

Tabular StreamBrowser Display

In tabular mode, the StreamBrowser tab displays a table listing some or all of the current depot's streams.



Name	Basis	Type	Time	#
kestrel	--	normal		1
kestrel_devel	kestrel_test	normal		3
kestrel_dvt_jjp	kestrel_devel	workspace		6
kestrel_dvt_mary	kestrel_devel	workspace		5
kestrel_maint	kestrel_test	normal		4
kestrel_mnt_jjp	kestrel_maint	workspace		7
kestrel_mnt_Rls1.0.1	kestrel_mnt	snapshot	Mar 7, 2004 10:18:28 PM	9
kestrel_mnt_Rls1.0.2	kestrel_mnt	snapshot	Mar 7, 2004 10:18:44 PM	10
kestrel_Rls1.0	kestrel	snapshot	Mar 7, 2004 10:18:10 PM	8
kestrel_Rls2.0	kestrel	snapshot	Mar 7, 2004 10:19:51 PM	12
kestrel_Rls2.1	kestrel	snapshot	Jun 8, 2004 4:26:30 PM	13
kestrel_Rls2.1_qf_derek	kestrel_Rls2.1_qfix	workspace		15
kestrel_Rls2.1_qf_mary	kestrel_Rls2.1_qfix	workspace		16
kestrel_Rls2.1_qfix	kestrel_Rls2.1	normal		14
kestrel_test	kestrel	normal		2

☐ Show including hidden ☑ snapshots All Workspaces

jjp kestrel

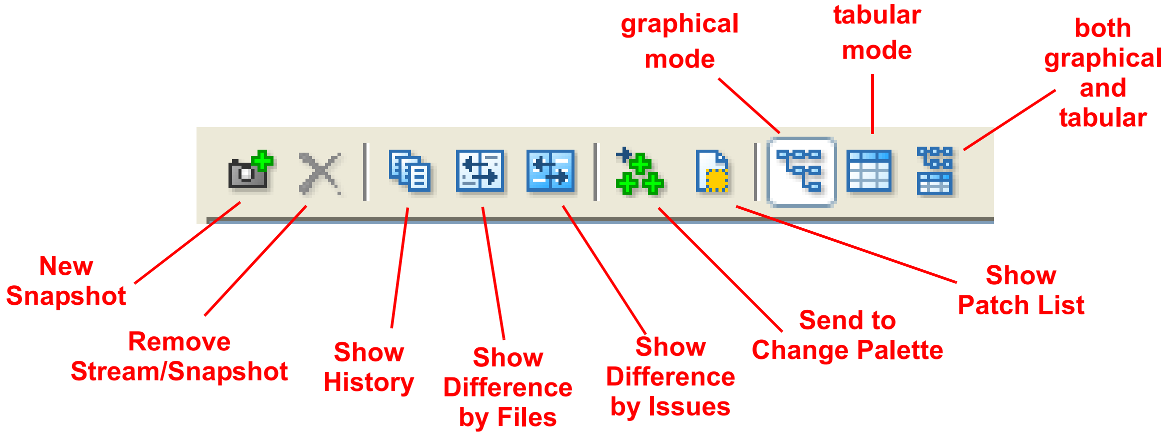
The checkboxes and listbox at the bottom of the tab work the same way as in graphical mode to control which streams are included in the table. As with most AccuRev tables, you can define single-column or multiple-column sorts to reorder the table's rows. You can drag column separators and drag-and-drop columns to rearrange the table's columns.

The StreamBrowser's tabular mode offers the same command toolbar and context menus as the graphical mode.

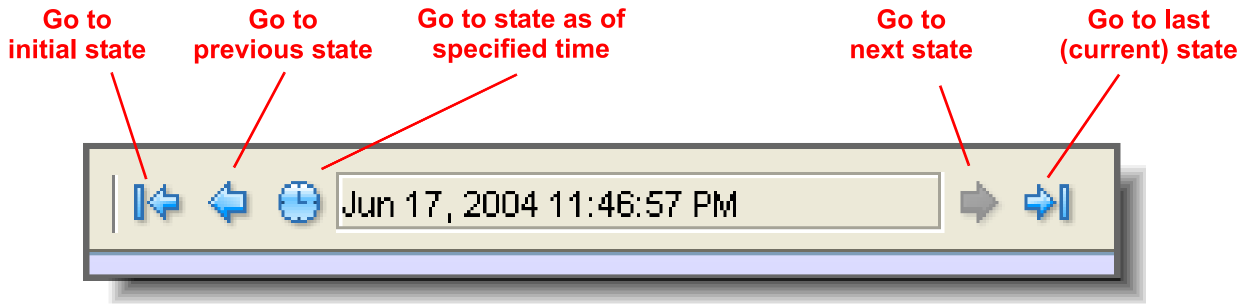
StreamBrowser Quick Reference

The following sections explain the commands on the StreamBrowser's toolbar, and on the context menu that appears when you right-click a stream, snapshot, or workspace.

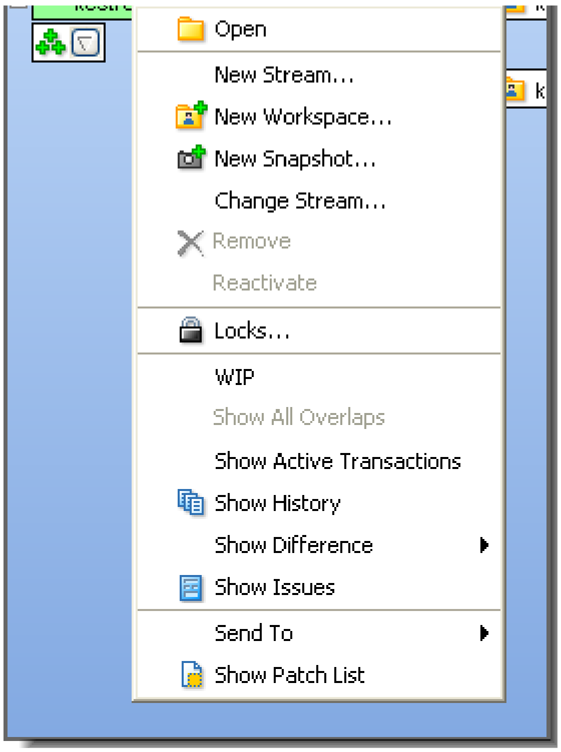
StreamBrowser: Standard Toolbar



StreamBrowser: Stream History Toolbar



StreamBrowser: Context Menu



Command	Description
Open	Open a File Browser tab for the selected stream or snapshot CLI: **files**
New Stream	Create a new dynamic stream, passthrough stream, or snapshot as a child of the selected stream. CLI: **mkstream**
New Workspace	Create a new workspace based on the selected stream or snapshot. CLI: **mkws**
New Snapshot	Create a new snapshot based on the selected stream or snapshot. CLI: **mksnap**
Change Stream	Change the specifications of a dynamic or passthrough stream. CLI: **chstream**
Remove	Deactivate the selected dynamic stream, passthrough stream, snapshot, or workspace. CLI: **remove**
Reactivate	Reactivate a stream that was previously **Removed** (but is still displayed, via the **Show hidden** checkbox).
Locks	Lock or unlock the selected stream. CLI: **lock**, **unlock**
WIP	Display work-in-progress for all workspaces backed by the selected stream. CLI: **wip**
Show All Overlaps	(workspace streams only) For files that are active in the selected stream, display all overlaps with versions in higher-level streams. CLI: **merge –B**
Show Active Transactions	Open a History Browser, displaying the transactions containing versions that need to be promoted out of the selected stream. CLI: **translist**

Show History	Open a History Browser, displaying the entire transaction history of the selected stream. CLI: **hist –s**
Show Difference By Files	Compare the selected stream with another stream that you click to specify. Differences in version-IDs and pathnames are displayed. CLI: **diff –a –i –v –V**
Show Difference By Issues	Compare the selected stream with another stream that you click to specify. Differences expressed in terms of change packages.
Show Issues	List the change packages that have been (partially or completely) incorporated into the stream, workspace, or snapshot.
Send to Change Palette	Load the Change Palette with the versions in the selected stream's default group. You must click to specify another stream as the destination for these versions. CLI: **mergelist**
Show Patch List	Lists all the individual versions that need to be patched to another stream, in order to have the other stream include all the changes in this stream. CLI: **patchlist**

The AccuRev Diff, Merge, and Patch Tools

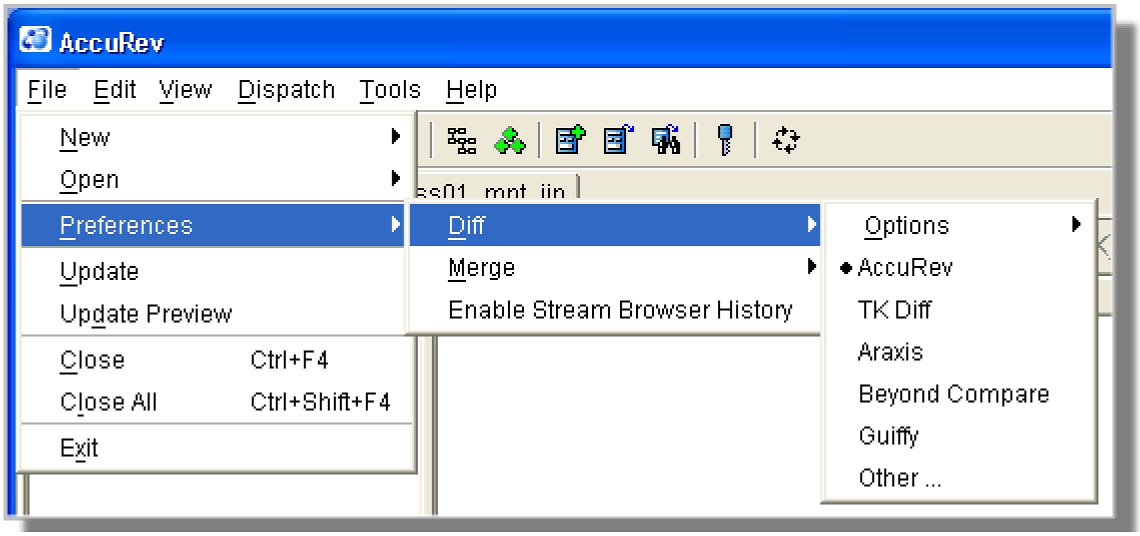
The AccuRev GUI includes tools that process two versions of the same file element:

- The **Diff** tool compares any two versions of a file element.
- The **Merge** tool combines the changes in two versions of a file element: your workspace's version and the version in the stream to which you intend to promote the workspace version.
- As of Version 3.5.5, the Merge tool can also be used to perform a **Patch** command (patch operation instead of merge operation). This enables you to incorporate just the recent changes made in the non-workspace version.

Since the merging of two versions of a file is a logical extension of finding their differences, the Diff and Merge tools are similar in many respects.

Enabling the Diff and Merge Tools

By default, the AccuRev GUI uses the "AccuRev" Diff and Merge tools. But you can configure it to use the tools that were included in previous AccuRev releases, or to use third-party or home-grown tools. The **File > Preferences** menu sets these configuration items.



These configuration settings are persistent; they are automatically reestablished the next time you start the AccuRev GUI.

If you select **Other** as the Diff or Merge tool, you are prompted to enter a command line to be executed. This command line must include substitution patterns, as indicated in the following table:

	Pattern	Meaning in Command String
Custom **Diff** Tool	`%1%`	first version to be compared
	`%2%`	second version to be compared

	Pattern	**Meaning in Command String**
Custom **Merge** Tool	`%a%`	common ancestor file
	`%1%`	version in backing stream
	`%2%`	file in workspace
	`%o%`	merge results (output) file

Make sure the alternative diff or merge program is located in a directory on your search path. The exit status of the merge program should be:

- Zero if there are no conflicting changes. The **merge** command's default "do next" action after running this program will be **keep**.
- Non-zero if there are conflicting changes. The **merge** command's default "do next" action after running this program will be **edit**.

Invoking the Diff Tool

There are several ways to specify the two versions of a file element to be compared:

- In a File Browser, each file's context menu includes a **Diff Against...** item. There are two choices, each of which compares your version (i.e. the file in your workspace) with a version stored in the depot:
  - **Most Recent Version**: Compares your file with the version currently in your workspace stream. Use this choice if you've modified the file since the last time you performed a **keep** on it (or if you've never performed a **keep** on it since the last update).
  - **Backed Version**: Compares your file with the version currently in the backing stream. For example, you might use this choice to see *all* the changes you've made to this file since you updated your workspace and starting working on the file. (And assuming no one else has promoted a new version to the backing stream in the meantime.) This might include the changes stored in several intermediate versions that you've created with **keep**.
- In a StreamBrowser display, you can view the elements in any stream's default group: right-click a stream, then select **View Default Group** from the context menu. Each element has a context menu that includes **Diff Against > Backed Version**. This invokes the Diff tool to compare the version in the selected stream with the version in its parent stream.
- In a Version Browser display (**History > Browse Versions** in the context menu of a file), you can compare any two versions: right-click any version and select **Show Difference** from its context menu; then left-click on any other version.

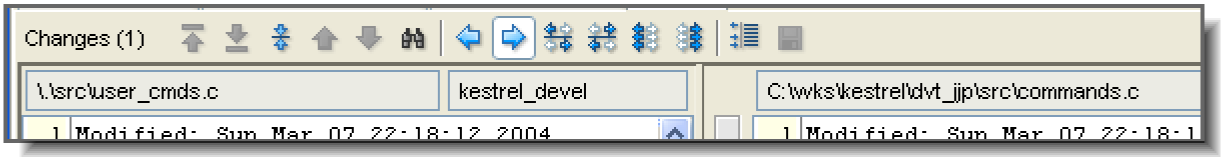
Using the Diff Tool

Each time you invoke it, the Diff tool opens a new tab of the overall AccuRev GUI window. When you're finished comparing the versions, you can close the Diff tab like any other tab: right-click on the tab's title and select **Close**.

Diff displays the two specified versions side-by-side, as described in the following sections.

Difference in Pathname

The two versions being compared can have different pathnames: the element may have been renamed or moved to another directory within the depot. (This may have occurred in one of the versions, or in both of them.) The Diff tool does not highlight or announce such a namespace-level difference. It shows the pathname of each version at the top of the difference display, enabling you to quickly determine whether they differ.



\.\src\user_cmds.c

kestrel_devel

C:\wks\kestrel\dvt_jjp\src\commands.c

```
1 Modified: Sun Mar 07 22:18:12 2004
1 Modified: Sun Mar 07 22:18:1
```

Comparison of Binary Files

If the versions to be compared are in a binary image format that AccuRev can render, the Diff tool simply displays the versions, so that you can determine their differences by inspection. As of Version 3.1, AccuRev can render the following image formats:

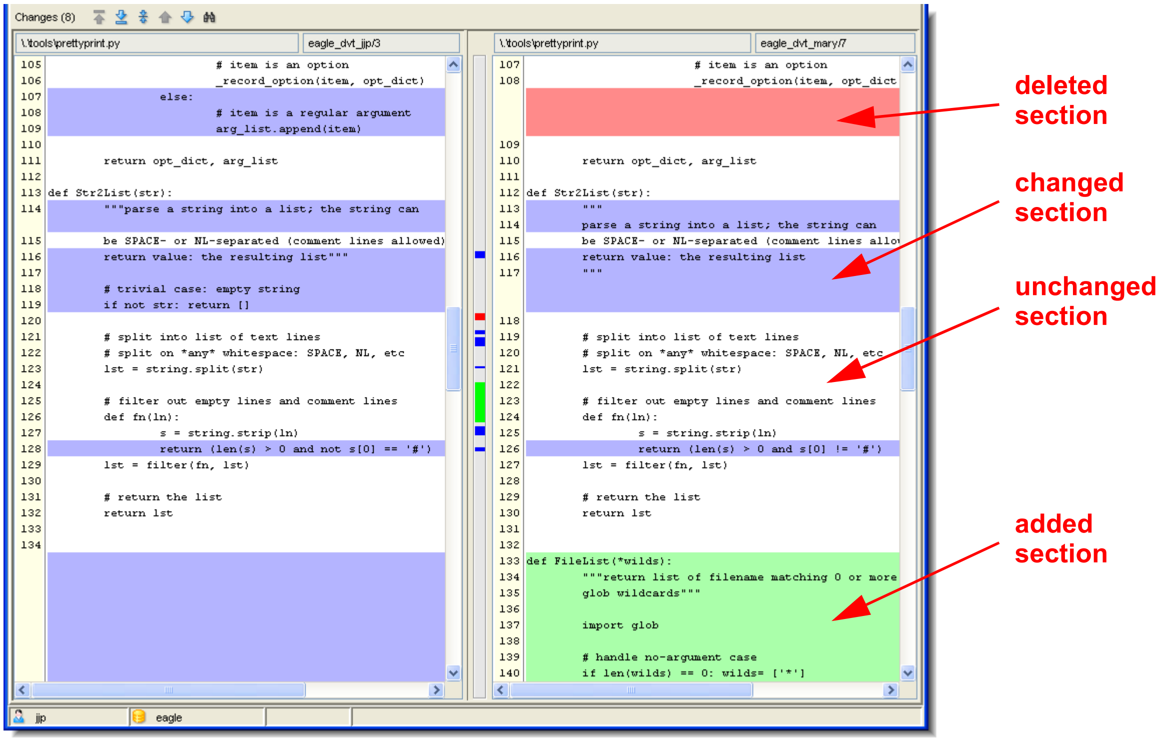
JPEG (**.jpg** or **.JPG** filename suffix)
PNG (**.png** or **.PNG**)
GIF (**.png** or **.GIF**)

Comparison of Text Files

If the versions to be compared are text files, the Diff tool displays them side-by-side in separate panes, so that corresponding text lines in the versions line up visually. We'll call them the "before version" (displayed on the left) and the "after version". Initially, the two panes are the same width, but you can drag the vertical separator to change the relative widths. To make both panes wider, just increase the size of the overall AccuRev GUI window.

> Note: depending on how you launch the Diff tool from the Version Browser, an older version might be displayed on the right, not the left. We recommend keeping the older version on the left, so that "before" and "after" in the descriptions below correspond to reality.

Through color-coding, the Diff tool partitions the text into the following kinds of sections:



```
            # item is an option
            _record_option(item, opt_dict)
        else:
            # item is a regular argument
            arg_list.append(item)

    return opt_dict, arg_list

def Str2List(str):
    """parse a string into a list; the string can

    be SPACE- or NL-separated (comment lines allowed)
    return value: the resulting list"""

    # trivial case: empty string
    if not str: return []

    # split into list of text lines
    # split on *any* whitespace: SPACE, NL, etc
    lst = string.split(str)

    # filter out empty lines and comment lines
    def fn(ln):
        s = string.strip(ln)
        return (len(s) > 0 and not s[0] == '#')
    lst = filter(fn, lst)

    # return the list
    return lst


```

```
            # item is an option
            _record_option(item, opt_dict



    return opt_dict, arg_list

def Str2List(str):
    """
    parse a string into a list; the string can
    be SPACE- or NL-separated (comment lines allo
    return value: the resulting list
    """



    # split into list of text lines
    # split on *any* whitespace: SPACE, NL, etc
    lst = string.split(str)

    # filter out empty lines and comment lines
    def fn(ln):
        s = string.strip(ln)
        return (len(s) > 0 and s[0] != '#')
    lst = filter(fn, lst)

    # return the list
    return lst


def FileList(*wilds):
    """return list of filename matching 0 or more
    glob wildcards"""

    import glob

    # handle no-argument case
    if len(wilds) == 0: wilds= ['*']
```

deleted section

changed section

unchanged section

added section

- **Unchanged Sections**: Text sections in which the versions are identical are displayed with a **white** background.
- **Added Sections**: If a text section occurs only in the “after” version, it is displayed there with a **green** background. Empty space, colored blue but without any line numbers, appears at that point in the “before” version.
- **Deleted Sections**: If a text section occurs only in the “before” version, empty space with a **red** background appears at that point in the “after” version. The deleted section is colored blue in the “before” version.
- **Changed Sections**: Sometimes, the Diff tool decides that a text section in the “before” version has been revised, producing the corresponding section in the “after” version. The before and after sections are not necessarily the same length, in which case some empty space is displayed in one of the versions. Both the before and after sections appear with a **blue** background.

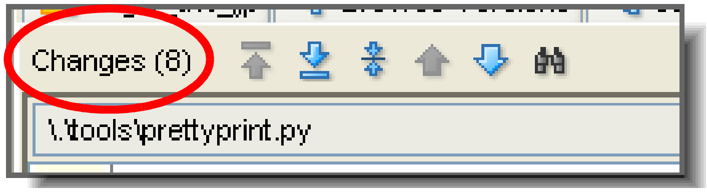
If a text block has been moved from one location in the file to another, the Diff tool indicates this as two separate changes: a section deleted at the original location, and another section added at the new location.

Navigating the Differences

Most text files are too long to fit on the display screen, and some are too wide. Accordingly, the panes in which the versions appear have both vertical and horizontal scroll bars. When you scroll one of the panes in any direction, the other one scrolls automatically. Line numbers at the edge of each pane help you to keep oriented as you scroll through long files.

For a text file that contains hundreds or thousands of lines, there may be only a few difference sections (added, deleted, or revised), separated by large unchanged sections. The Diff tool offers two ways to navigate among the difference sections:

- **Sequential Access**: There's a status indicator and a toolbar at the top of the Diff tab. (This is distinct from the toolbar for the overall AccuRev GUI window.) The status indicator shows of the total number of difference sections found:



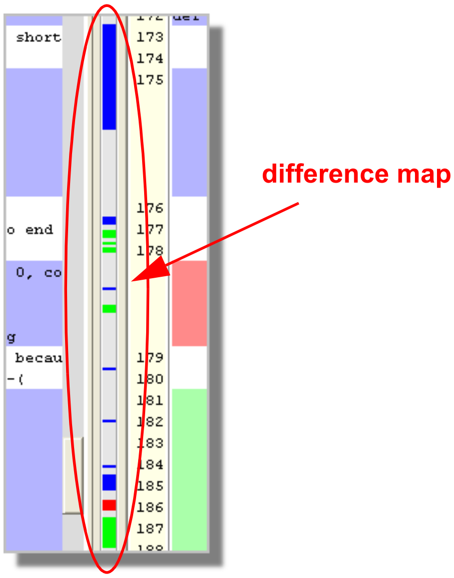
\.\tools\prettyprint.py

  The toolbar buttons **Next Diff** and **Previous Diff** provide sequential access to all the difference sections. The **First Diff** and **Last Diff** buttons are useful for restarting a scan of all the difference sections.

  Whenever you use one of these buttons to jump to a particular difference section, the Diff tool remembers it as the current difference and highlights the lines numbers in both panes.

- **Random Access**: Between the two panes, there's a difference map that shows the relative locations and sizes of all the difference sections. The map uses the same color-coding as the difference sections themselves: added sections in green, deleted sections in red, revised sections in blue. Click on any colored area of the map to scroll both panes directly to the corresponding difference section. (Actually, you can click *anywhere* in the difference map; the panes will scroll to that location, even if the files are identical there.)

  Using the difference map merely scrolls the panes; it does not change the current difference. But after you jump to a particular location, you can then click the difference section at that location to make it the current difference.



difference map

Using the scroll bars or the difference map, you can make the current difference scroll off screen. To bring it back onscreen, click the **Center** button on the Diff tab's toolbar.

Searching for Text

In addition to navigating among the difference sections, you can search for any text string in either pane, using the **Search** toolbar button.

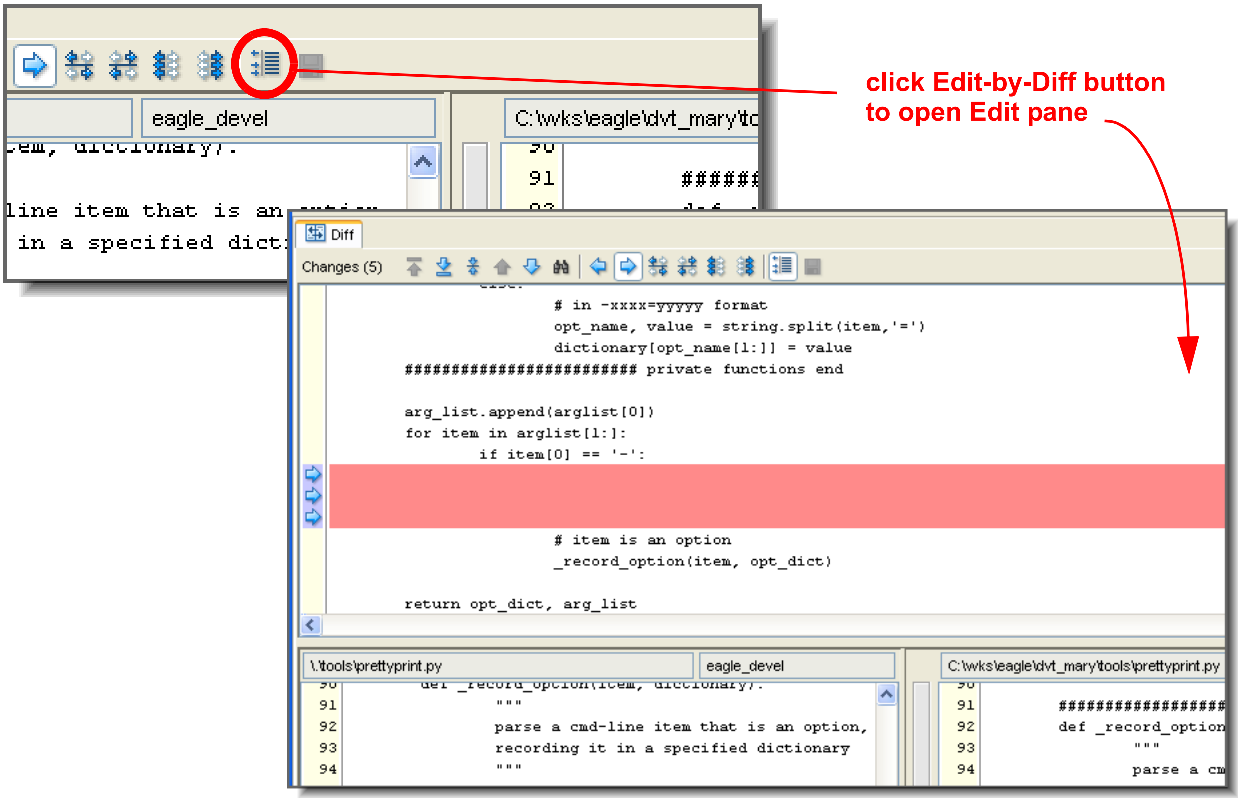
Editing a File Using the Diff Tool

In addition to comparing two versions of a file, the Diff tool can help you to edit the contents of the version in your workspace. This Edit-by-Diff capability is available only when you're comparing your workspace version with another version; it's not available when you invoke the Diff tool from the Version Browser.

Here's a procedure for using this capability:

1. Invoke the Diff tool in a File Browser, to compare your workspace version with some other ("before") version of the same file. (You cannot invoke Edit-by-Diff from the Version Browser.)

2. Click the **Edit-by-Diff** button in the Diff toolbar. This opens a third pane (Edit pane), which initially contains the same text as your workspace version.



click Edit-by-Diff button to open Edit pane

```
line item that is an option
 in a specified dict
```

```
                # in -xxxx=yyyyy format
                opt_name, value = string.split(item,'=')
                dictionary[opt_name[1:]] = value
        ######################### private functions end

        arg_list.append(arglist[0])
        for item in arglist[1:]:
            if item[0] == '-':

                # item is an option
                _record_option(item, opt_dict)

        return opt_dict, arg_list
```

```
        """
        parse a cmd-line item that is an option,
        recording it in a specified dictionary
        """
```

```
    #################
    def _record_option
        """
            parse a cm
```

3. As you browse through the difference sections, using the techniques described above, you can click the **Revert My Change** toolbar button to swap in the "before" version's text at that point (a change, addition, or deletion). You can also add the "before" text to your text, using one of the "take both" buttons, or .

4. At any time, you can edit text manually in the Edit pane.

5. If you change your mind about a difference section where you've swapped in text from the "before" version, select that section and click the **Restore My Change** toolbar button. Any manual edits you've made in that difference section will be lost.

6. Clicking the **Revert All of My Changes** button is a shortcut for visiting every difference section and clicking **Revert My Change**. Similarly, the **Restore All of My Changes** button is a shortcut for visiting every difference section and clicking **Restore My Change**.

7. When you're done, click the **Save Edits & Close** toolbar button. This simply replaces the file in your workspace. (It doesn't perform an AccuRev **keep** or **promote** command.) To cancel the edit session without saving any changes to the file, just close the Diff tab.

Note: using the Edit-by-Diff capability is similar to using the Merge tool (described below). An important difference is that an Edit-by-Diff operation involves just two versions, and a Merge operation takes into account a third version: the closest common ancestor of the two versions being merged.

When to Use the Merge Tool

Perhaps the most common usage pattern in AccuRev is:

- Using **keep** to create one or more versions of a text file in your workspace.
- Using **promote** to propagate the most recently kept version to the backing stream.

Occasionally, someone else "gets there first". That is, both you and a colleague are working on the same file concurrently, each of you using a copy of the file in your own workspace. And the colleague promotes his changes to the common backing stream before you do.

Before you can promote your own changes to the backing stream, you must first merge the changes from your colleague's version with your changes. Merging ensures that no one's work is inadvertently lost or overwritten in a concurrent development environment.

The AccuRev GUI makes it easy to tell which files require a merge prior to promotion. If a file needs to be merged, the details pane in your workspace shows the file as having an **overlap** status — that is, the work of one or more of your colleagues has overlapped your own. If you select **Overlap** in the GUI's Searches View, the details pane displays all your **overlap**-status files: all files in the depot that you must merge before you can promote them from your workspace to the backing stream.

Although the above scenario is by far the most common one, AccuRev's flexibility allows for other merge scenarios, too:

- You can modify the file in your workspace by "pulling in" the changes from *any* stream's version of the file, not just the backing stream's version.
- You can merge any two dynamic streams' versions of the file.

These scenarios involve using the GUI's **Change Palette**. In all cases, however, you use the Merge tool in the same way, to combine the contents of two versions of the same file element.

The Merge Algorithm

Before describing the Merge tool's interface, we briefly describe how the tool works.

The Merge tool analyzes two contributor versions of a file, at both the content level and the namespace level:

- It combines the two versions' contents to produce a merged version of the file. Sometimes, it can produce the merged version completely automatically; other times, it needs help from you to resolve conflicts between the versions.
- It compares the pathnames of the two versions. If the pathnames differ — because one or both of the contributors was renamed, or was moved to a different directory within the depot — it applies the change to the merged version. As above, the Merge tool sometimes makes the name change automatically, but sometimes needs help from you to resolve a naming conflict.

The analysis that the Merge tool performs on the contributor versions at the content level is similar to that performed by the Diff tool. Both tools identify difference sections, where the contributors differ from each other. That's all the Diff tool needs to do, but the Merge tool goes on to decide how each difference represents a change from the past. In this context, "the past" means the closest common ancestor version of the two contributor versions.

To emphasize the Merge tool's perspective, we use the term change section to describe a location where the contributors differ from each other. (In the context of the Diff tool, we use the term difference section.)

A Non-Conflicting Content Change

For example, suppose a change section consists of 13 lines that occur in contributor #2 but not in contributor #1. To determine what kind of change this represents, the Merge tool looks at the corresponding location in the closest common ancestor version:

- If those 13 lines exist in the ancestor version, the Merge tool concludes that a change was made in contributor #1 (the lines were deleted) but no change was made in contributor #2 (the lines are still there).
- If the 13 lines do not exist in the ancestor version, the Merge tool concludes that a change was made in contributor #2 (the lines were added) but no change was made in contributor #1 (nothing was added).

In both these cases, there was a change from the common ancestor in exactly one of the contributors. The Merge tool deems this a non-conflicting change. It automatically incorporates the change (be it an addition, a deletion, or a revision of existing text) into the merged version.

A Conflicting Content Change

Let's take another example. A one-line error message has a slightly different wording in the two contributors:

- Contributor #1:

```
#define E_COLOR498    "No color with that name was found."
```

- Contributor #2:

```
#define E_COLOR498     "Color name unknown."
```

The following line occurs at the corresponding location in the closest common ancestor version:

```
#define E_COLOR498     "Huh?"
```

In this situation, the Merge tool finds a change from the common ancestor in *both* contributors, not just one of them. This is a conflicting change (or more simply, a conflict). The Merge tool doesn't try to decide which contributor's change is better. It just makes it easy for *you* to make this decision when you perform the merge.

> Note: there's another kind of change, where both contributors have made *the same* change from the common ancestor version. For example, both contributors might have replaced the error message **Huh?** with the message **No such color**. In such cases, the Merge tool silently incorporates the agreed-upon change into the merged version.

Changes at the Namespace Level

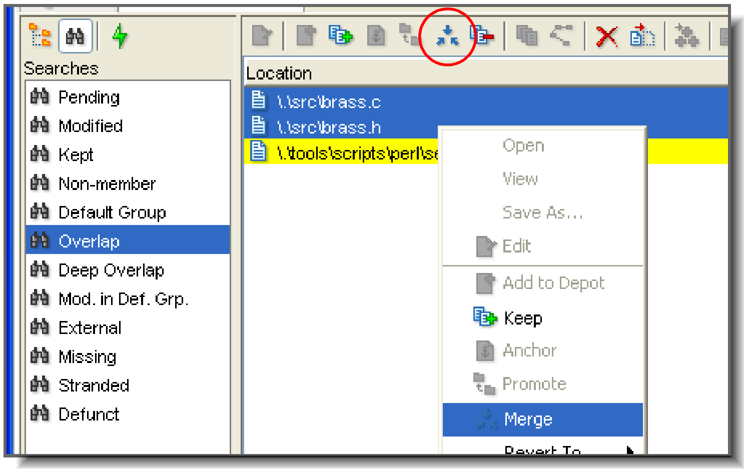
Performing a merge at the namespace level is simpler than at the content level:

- If exactly one of the contributors had its pathname changed, the merged version automatically gets the new pathname.
- If both contributors had their pathnames changed, the Merge tool prompts you to select one of the new pathnames. (You can also choose to go back to the original pathname of the common ancestor version.) The merged version gets the pathname you choose.
- If both contributors had their pathnames changed in exactly the same way — for example, both versions were renamed from **chap01.doc** to **chapter01.doc** — the merged version automatically gets that new pathname.

Invoking the Merge Tool

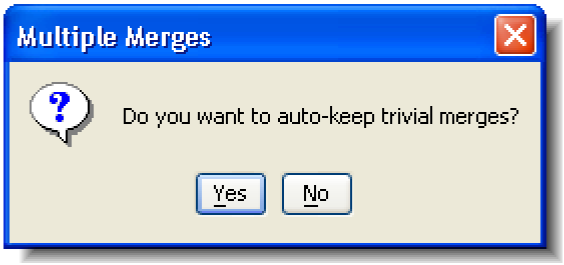
To invoke the Merge tool, select one or more files with **(overlap)** status, then click the **Merge** toolbar button. (Alternatively, invoke **Merge** from the file's context menu.)

You can do this in the File Browser's details pane, in either the standard directory view or in searches view. You can also invoke the Merge tool from the Change Palette, to merge versions in higher-level streams.



The Merge tool processes the files one-by-one. For each one, it opens a new Merge tab in the overall AccuRev GUI window. On this tab, you interactively merge these two contributor versions: (1) the file in your workspace; (2) the version in the backing stream, containing changes made by one or more of your colleagues.

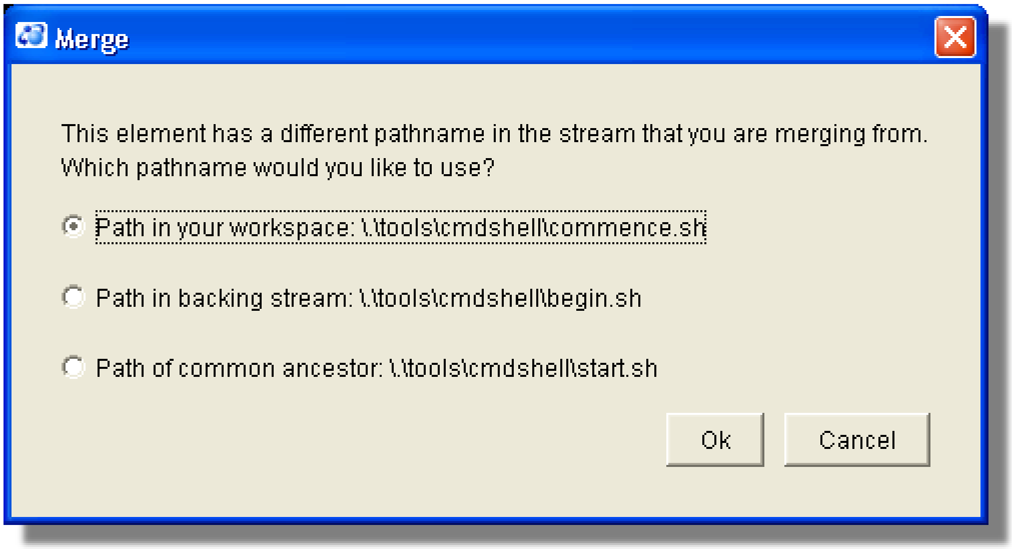
If you select multiple files, you can choose to have the Merge tool merge as many of them automatically as possible. An automatic merge is possible if there are no conflicting changes between the two versions being merged (see above). For files in which there are conflicting changes, the Merge tool always opens an interactive Merge tab.



Do you want to auto-keep trivial merges?

Resolving a Namespace-Level Conflict

As outlined above, a namespace-level conflict exists if *both* contributors had their pathnames changed, and the new pathnames differ from each other. The Merge tools prompts you to resolve the conflict by choosing one of the names. You can also choose to restore the name of the common ancestor version.



This element has a different pathname in the stream that you are merging from.
Which pathname would you like to use?

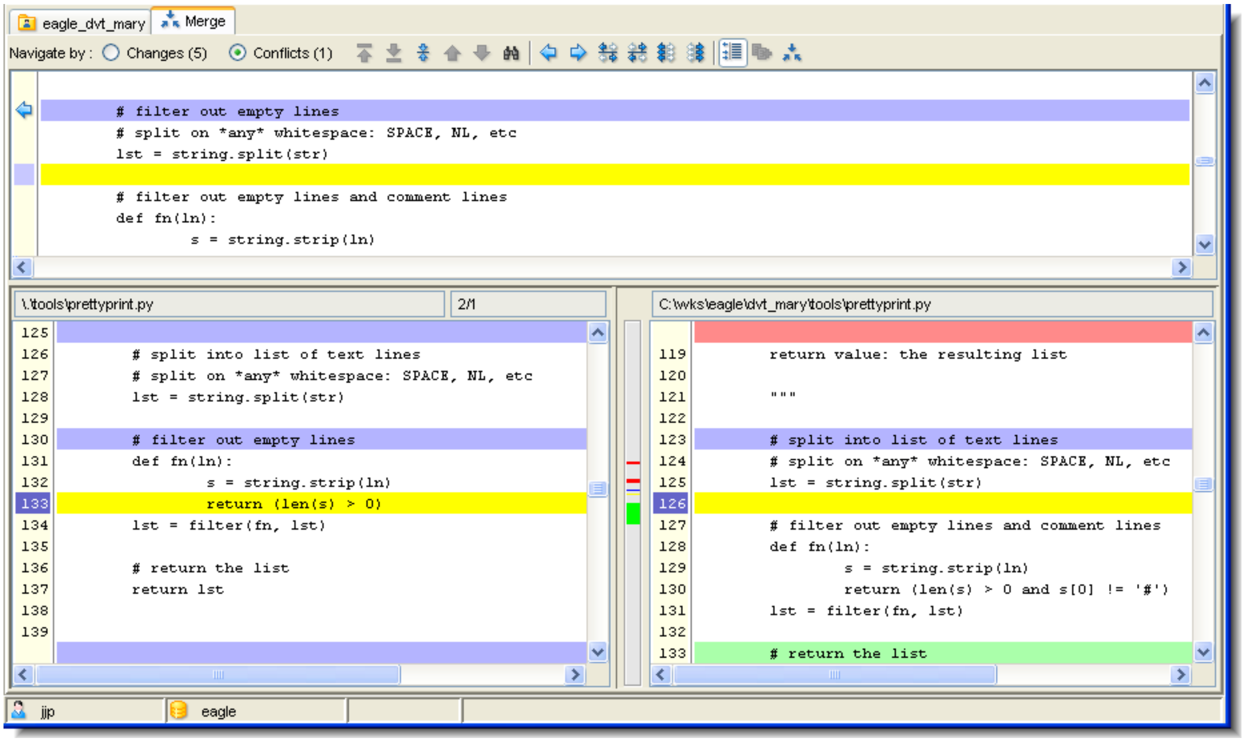
● Path in your workspace: \.\tools\cmdshell\commence.sh

○ Path in backing stream: \.\tools\cmdshell\begin.sh

○ Path of common ancestor: \.\tools\cmdshell\start.sh

Viewing and Resolving Content-Level Conflicts

The Merge tool displays the two contributors’ contents side-by-side, with the file in your workspace on the right. Above them, it displays the merged version. The bottom panes are synchronized: scrolling either one of them causes the other to scroll, too.



Navigate by : ○ Changes (5) ● Conflicts (1)

```
        # filter out empty lines
        # split on *any* whitespace: SPACE, NL, etc
        lst = string.split(str)

        # filter out empty lines and comment lines
        def fn(ln):
                s = string.strip(ln)
```

\.\tools\prettyprint.py   2/1

```

        # split into list of text lines
        # split on *any* whitespace: SPACE, NL, etc
        lst = string.split(str)

        # filter out empty lines
        def fn(ln):
                s = string.strip(ln)
                return (len(s) > 0)
        lst = filter(fn, lst)

        # return the list
        return lst


```

C:\wks\eagle\dvt_mary\tools\prettyprint.py

```
        return value: the resulting list

        """

        # split into list of text lines
        # split on *any* whitespace: SPACE, NL, etc
        lst = string.split(str)

        # filter out empty lines and comment lines
        def fn(ln):
                s = string.strip(ln)
                return (len(s) > 0 and s[0] != '#')
        lst = filter(fn, lst)

        # return the list
```

Like the Diff tool, the Merge tool partitions the contributors' contents, displaying unchanged sections with a white background and difference sections with colored backgrounds. The color-coding scheme is similar to the Diff tool's, but not identical. That's because the Merge tool has the more complex job of distinguishing conflicting changes from non-conflicting ones.

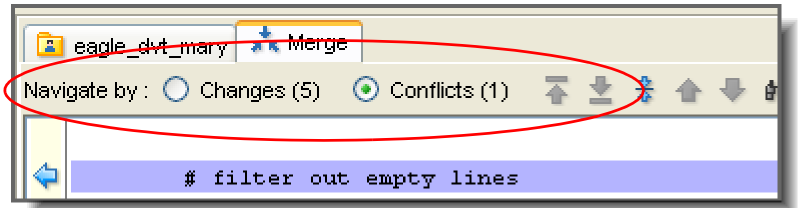
- For each non-conflicting change (where just one contributor differs from the ancestor version), the Diff color-coding scheme is used: the block in the backing-stream version (left side) is colored **blue**; the corresponding block in your version (right side) is colored **green** (text you've added), **red** (text you've deleted), or **blue** (text you've revised).
- For each conflict (where both contributors differ from the ancestor version), the blocks in both contributors are colored **yellow**.

The merged version is displayed with color-coding, too. Each colored block is a location where a change has been incorporated from one of the contributors into the merged version. When it starts up, the Merge tool performs as much merging as it can automatically, applying all of the non-conflicting changes to the merged version. Color indicates the origin of the change: blue if it comes from the left-hand contributor; green, red, or blue if it comes from the right-hand contributor. In addition, the change is marked with left-arrows or right-arrows, indicating which contributor provided the change.

The Merge tool cannot resolve conflicts automatically, so the location of each conflict is initially indicated in the merged version by a blank yellow block.

Navigating the Display

The Merge tool is similar to the Diff tool in its navigation facilities. The panes have scroll bars, which are synchronized with each other. A status indicator at the top of the Merge tab reports the total number of change sections (**Changes**) found in the contributors. It also tracks the current number of conflicting changes (**Conflicts**) that you have not yet resolved.



Navigate by : ○ Changes (5) ● Conflicts (1)

```
# filter out empty lines
```

For example, the status indicators may read as shown here when the Merge tool starts up.

This means that there are a total of 5 change sections; 4 of them are non-conflicting changes, which are automatically applied to the merged version; 1 of them is a conflict, which you must resolve during the merge session.

You can browse sequentially through the change sections using the Merge toolbar buttons **Next**, **Previous**, **First**, and **Last**. Click the **Changes** radio button to have these buttons browse all the change sections; click the **Conflicts** radio button to have these buttons browse just the unresolved conflicts.

Between the two contributor panes, there’s a change map that shows the relative locations and sizes of all the change sections. The map uses color-coding to indicate the kind of change. Click on any colored area of the map to scroll all the panes directly to the corresponding change section.

Whenever you use a toolbar button or the change map to jump to a particular change section, the Merge tool remembers it as the current change. Using the scroll bar, you can make the current change scroll off screen. To make it visible again, click the **Center** button on the Merge tab’s toolbar.

Resolving Conflicts

Sometimes, the Merge tool can construct a merged version without any help from you. This occurs if all the change sections are non-conflicting. The Merge tool just applies all the changes to the merged version and announces that it’s done. This section describes the more interesting case: the contributors have one or more conflicts, which you must resolve.

The Merge tool automatically jumps to the first conflict. That is, it scrolls to the first yellow-highlighted change section in the contributors, and the corresponding blank yellow block in the merged version. Now, you must decide which of the changes is to be incorporated into the merged version:

- Click the **Take their change** toolbar button to incorporate the change in the left-hand contributor.

- Click the **Take my change** toolbar button to incorporate the change in the right-hand contributor.

- The other two buttons provide a way to incorporate *both* contributors' changes — in either order — into the merged version. Typically, you'll need to do some manual editing to "smooth out" the combined changes. See the *Manual Editing* section below.

When you click the button, the selected change appears in the merged version and the highlight changes from yellow (unresolved) to blue (resolved). Also, the Conflicts counter in the Merge toolbar is decremented.

You've resolved one conflict; all you need to do is resolve the rest of them in the same way. Use the ⇩ **Next** button to scroll down to the next conflict. It may take more than one click if sequential browsing is in Changes mode rather than in Conflicts mode. Again, use one of the "take changes" buttons to select the text from the left contributor or the right contributor (or both of them).

You don't have to resolve the conflicts in order. You can jump around as much as you want among the change sections, or scroll through unchanged sections to look up information the affects your merge decisions. The Conflicts counter always shows how much more work you need to do to complete the merge.

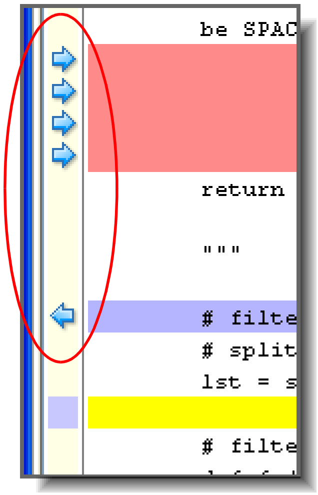
You can also change your merge decisions as much as you want. For example, you can revisit a particular change section where you had selected ⇦ **Take their change** and click the ⇨ **Take my change** button. This switches the text incorporated into the merged version.

The 'Revert All' and 'Use Only' Buttons

There are two additional toolbar buttons for resolving conflicts. The **Revert all of my changes** button makes the merged version the same as the "from" version (typically, the version in the backing stream). The **Use only my changes** button makes the merged version the same as the "to" version (the version in your workspace).

Here's a good way to think of this. The merged version has arrow annotations that indicate how each conflict has been resolved by you, and how each non-conflicting change has been resolved automatically by the Merge tool. "Revert all" makes all these arrows point to the left; "Use only" makes all these arrows point to the right.

It's important to note that these operations can undo the Merge tool's automatic resolution of one or more non-conflicting changes.



```
be SPAC
return
"""
# filte
# split
lst = s
# filte
```

"Revert all" makes all these arrows point left

"Use only" makes all these arrows point right

Manual Editing

At any time during a merge session, you can manually edit the contents of the merged version. Just click anywhere in the pane containing the merged version, and type. The **Delete** and **Backspace** keys work as expected. Using context (right-click) menus, you can **Copy** and **Paste** sections of text that you've highlighted with the mouse. You can also use the common keyboard shortcuts: **Ctrl-C** or **Ctrl-Ins** to copy, **Ctrl-V** or **Shift-Ins** to paste.

Notes:

If you make changes manually within a conflict section that you've resolved with a **Take ... change** button, the manual changes will be overwritten if you click any of the **Take ... change** buttons again.

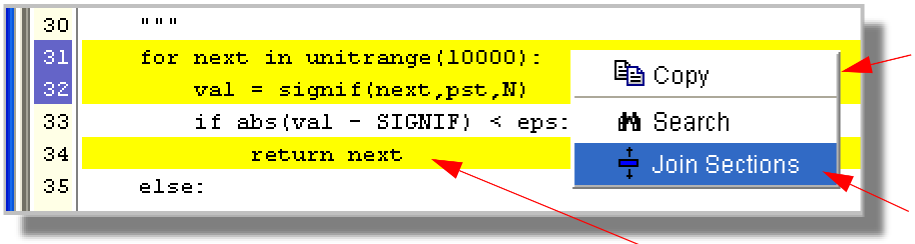
Selection of text sections using the keyboard is not currently supported.

Searching for Text

As with the Diff tool, you can search for any text string in either contributor pane, using the **Search** toolbar button. You can also invoke this command from the context menu that appears when you right-click anywhere in one of the contributor panes.

Joining Change Sections

Sometimes, there is a block of lines that you consider to be a single change section, but that the Merge tool decides are two discrete sections. You can tell the Merge tool to combine the two change sections:



```
    """
    for next in unitrange(10000):
        val = signif(next,pst,N)
        if abs(val - SIGNIF) < eps:
            return next
    else:
```

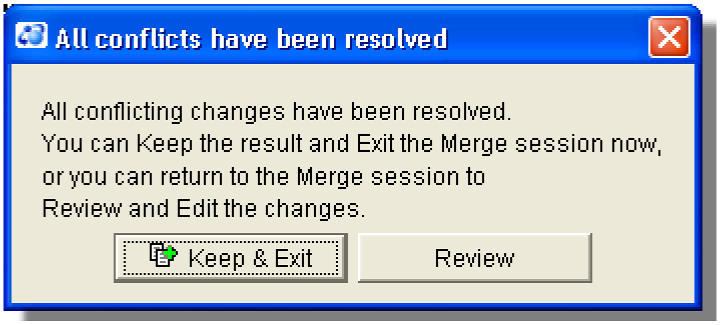
1. Right-click one of the change sections

2. Select **Join Sections** from the context menu

3. Click the other change section

Finishing a Merge Session

When you resolve the last remaining conflict by clicking one of the “take changes” buttons, the Conflicts counter goes to zero and the Merge tool displays this message window:



All conflicting changes have been resolved.
You can Keep the result and Exit the Merge session now,
or you can return to the Merge session to
Review and Edit the changes.

Clicking **Keep & Exit** ends the merge session immediately, closing the Merge tab. AccuRev overwrites the file in your workspace with the merged version. (Thus, you can think of the Merge tool as a fancy text-editor — it modifies the contents of a file in your workspace.) Then, it immediately preserves this modified file by **keep**ing a new version in your workspace stream. The **overlap** status of the file is removed, so that you can now **promote** the new version to the backing stream.

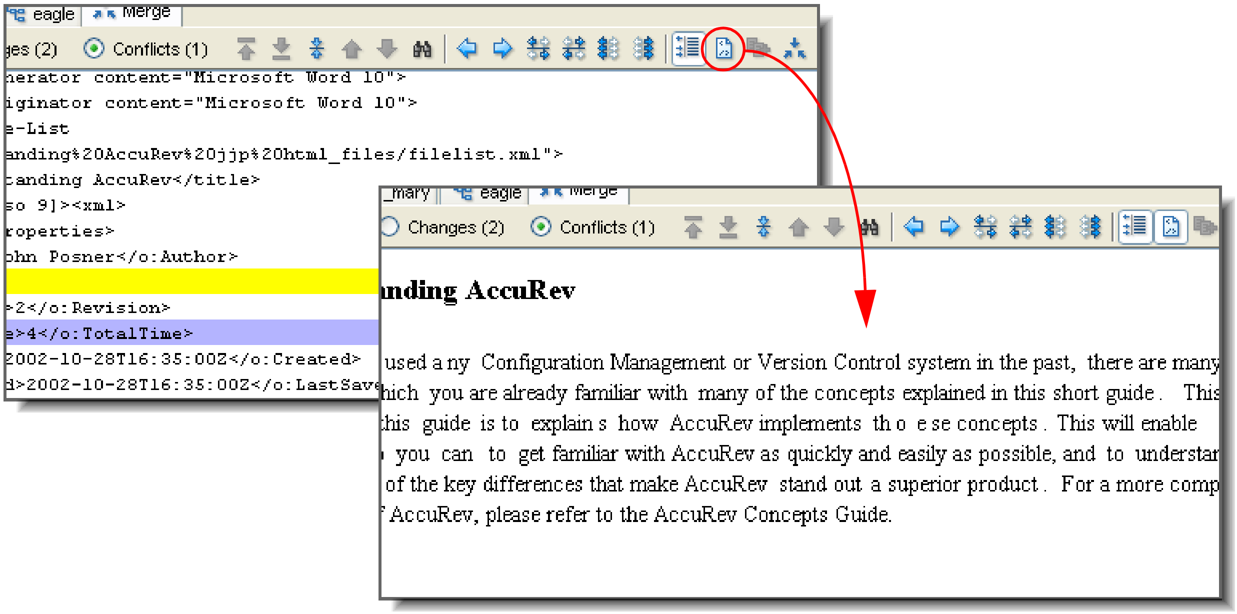
Clicking **Return** continues the merge session. You can review your work, change some of your merge decisions, and perform manual edits. At any time, you can click the **Keep** button in the Merge toolbar to complete the merge process.

You can also cancel the entire merge during a review pass by closing the Merge tab without keeping: right-click on the tab title, then confirm.

Merging HTML Files

AccuRev’s merge algorithm treats HTML-format files just like all other text files: merging takes place line-by-line; no attempt is made to parse the files’ HTML data structures in determining difference sections.

To help you determine whether merging is producing a valid (and desirable) result, though, an HTML viewer is built into the Merge tool. At any time, you can click the **Show HTML Result** toolbar button to render the current contents of the merged version. In the upper pane, the merged HTML-format text is replaced by a rendering of the text.



```
herator content="Microsoft Word 10">
iginator content="Microsoft Word 10">
e-List
anding%20AccuRev%20jjp%20html_files/filelist.xml">
tanding AccuRev</title>
so 9]><xml>
roperties>
ohn Posner</o:Author>

>2</o:Revision>
e>4</o:TotalTime>
2002-10-28T16:35:00Z</o:Created>
d>2002-10-28T16:35:00Z</o:LastSave
```

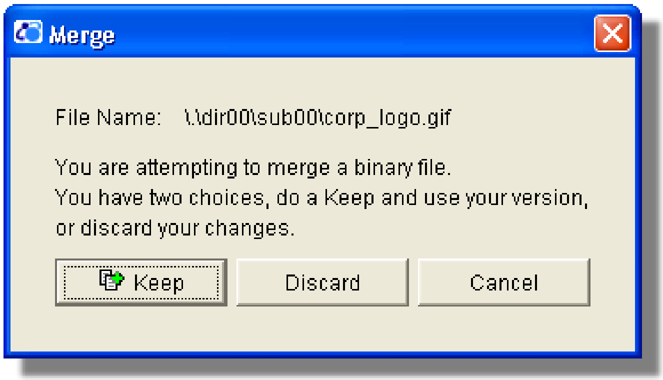
nding AccuRev

used a ny Configuration Management or Version Control system in the past, there are many
hich you are already familiar with many of the concepts explained in this short guide. This
this guide is to explain s how AccuRev implements th o e se concepts. This will enable
you can to get familiar with AccuRev as quickly and easily as possible, and to understar
of the key differences that make AccuRev stand out a superior product. For a more comp
f AccuRev, please refer to the AccuRev Concepts Guide.

This is a toggle button; click it again when you're finished viewing the rendered HTML code and you want to return to performing the merge.

Merging Binary Files

No generally accepted algorithm exists for merging the contents of binary-format files. But it is quite possible for a binary-format element to get into an overlap state in a concurrent development environment. For example: team members Justine and Derek each revise the corporate logo, **corp_logo.png**, using an image editor. Justine keeps and promotes the file to the backing stream, creating an overlap situation for that file in Derek's workspace.



File Name: \.\dir00\sub00\corp_logo.gif

You are attempting to merge a binary file.
You have two choices, do a Keep and use your version,
or discard your changes.

To resolve the overlap, Derek invokes the Merge command, just as he would for a text file. The Merge tool, seeing that **corp_logo.png** is in binary format, offers him the only two possible choices:

- If he selects **Keep**, Derek retains his changes to the file. A new version is created in Derek's workspace, recording a merge from Justine's version. This allows Derek to promote his version to the backing stream.
- If he selects **Discard**, Derek's changes are purged from his workspace. The workspace reverts to using the version of the file that was current at the previous update. The file's status becomes stale, reflecting the fact that Justine has created a new version in the backing stream since that update. Derek can bring Justine's version into his workspace with the **Update** command.

Performing a Patch Instead of a Merge

> Note: operationally, performing a patch is exactly the same as performing a merge. Using the Merge tool, you combine text from some other source with the contents of the file in your workspace. This section explains the difference between the patch and merge operations from an SCM perspective.

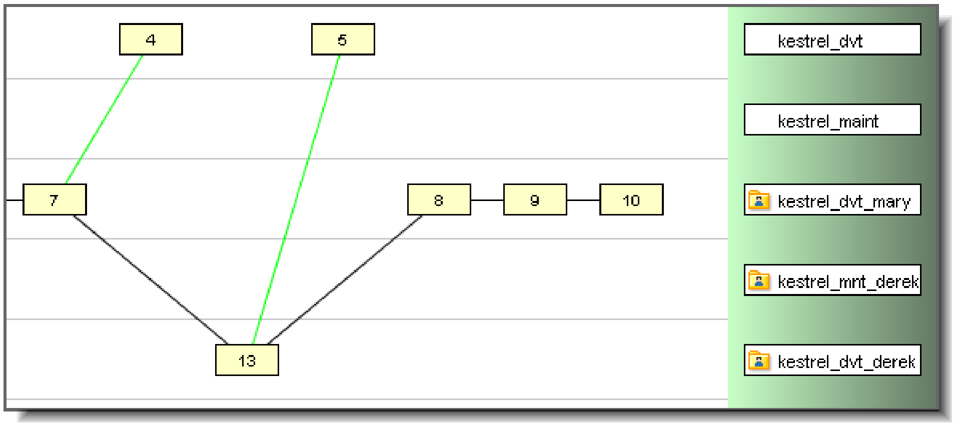
When you merge version V of a file into your workspace version, you are essentially saying, "combine my file with version V, taking into account all the changes in version V, back to the common ancestor". By contrast, when you patch version V into your workspace version, you are saying, "combine my file with version V, taking into account *only the recent changes* to version V".

In AccuRev 3.5.5, the meaning of "only the recent changes" has been modified to make the version-patching facility more effective and less labor-intensive. In previous releases, it meant "just the changes between version V and its immediate predecessor". For example, suppose user Mary recently created versions 4, 5, 6, and 7 of a file in her workspace, **dvt_mary**. When patching version **dvt_mary/7** into your workspace version, AccuRev previously only incorporated the changes between versions **dvt_mary/6** and **dvt_mary/7**. The drawback of this simple algorithm is clear: you probably wanted to incorporate *all* of Mary's recent changes — the modifications in versions 4, 5, 6, and 7. In previous releases, you would have had to perform four separate **Patch** commands to achieve this result.

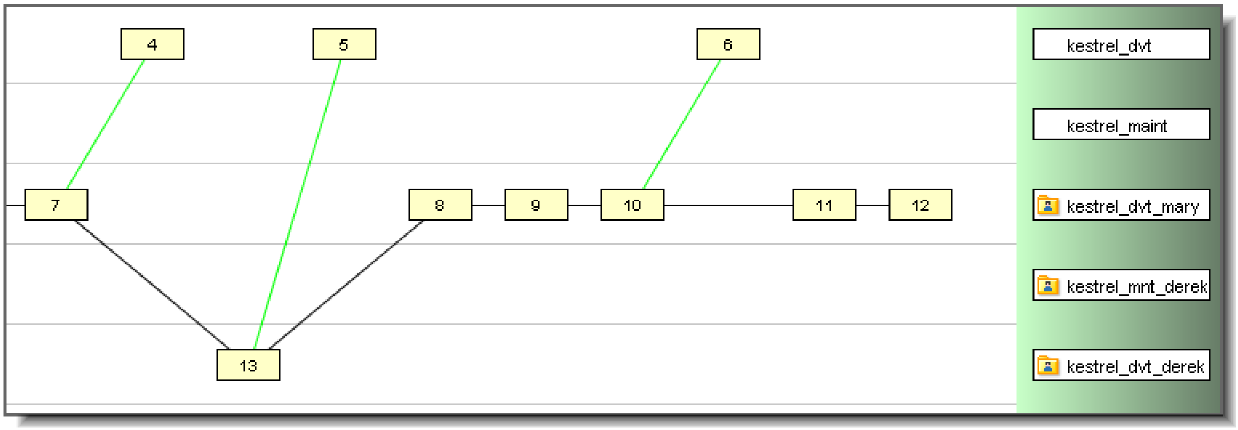
AccuRev 3.5.5 implements a more reasonable definition of "only the recent changes in Version V". It scans backward through the file's ancestry, starting at version V, stopping when it encounters a version that was originally created in another workspace, or a version that was promoted to another stream. This older version, termed the basis version, is judged to precede (and not belong to) the set of "recent changes" in version V.

Examples:

- Before Mary started her recent work, she updated her workspace. This brought in a version of the file originally created by another user — say, version **dvt_derek/13**. Then she proceeded to create versions **dvt_mary/8** through **dvt_mary/10**. When you patch version **dvt_mary/10** into your workspace, AccuRev searches backward through the element's ancestry, and includes the changes in all the versions recently created in the same workspace: **dvt_mary/8**, **dvt_mary/9,** and **dvt_mary/10**. It doesn't include the predecessor of **dvt_mary/8** — version **dvt_derek/13** — because it was created in a different workspace.



- Suppose Mary started working as described above, but proceeded a bit differently: she created versions **dvt_mary/8** through **dvt_mary/10**, promoted her work to the backing stream, then created versions **dvt_mary/11** and **dvt_mary/12**. In this case, when you patch version **dvt_mary/12** into your workspace, AccuRev decides that only the versions since the promotion — versions 11 and 12 — contain “recent changes”. The idea is that a promotion typically marks the end of a programming task, not an intermediate checkpoint.

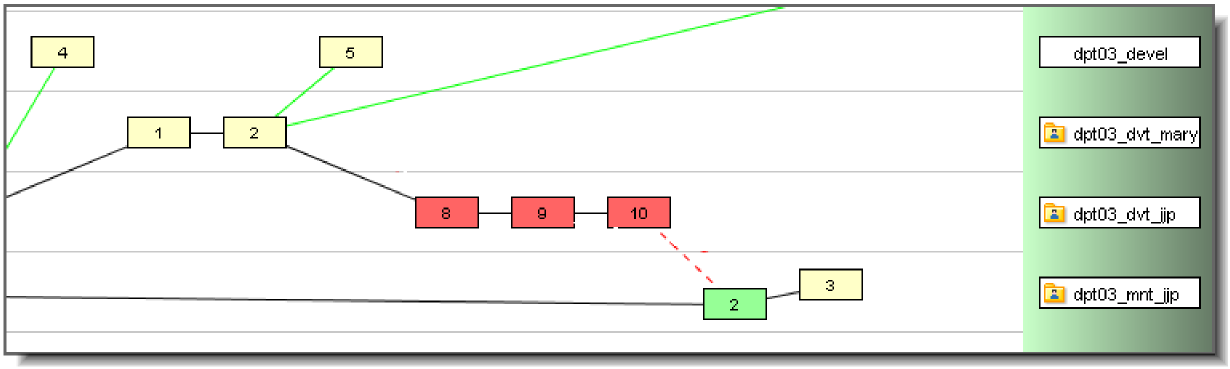


The basis version is indicated in the **keep** transaction that records a **Patch** command in the repository:

```
transaction 106; keep; 2004/10/13 13:38:39 ; user: jjp
  # patched
  version 7/3 (7/3)
  ancestor: (7/2) patched from: (5/2)-(6/7)
```

In this command, version 6/12 was patched into version 7/2, and the result was kept as version 7/3. The backward search through the file’s ancestry identified 6/10 as the basis version. So the patch encompasses the series of versions in workspace #6 that follow version 6/10, up to and including version 6/12.

As in previous releases, the Version Browser uses a dashed red line to indicate a version created by a **Patch** command. Selecting the version causes the entire series of versions included in the patch to be highlighted in red.



The predecessor of the earliest version in the highlighted series is the base version (**dpt03_dvt_mary/2** in the illustration above); the latest version in the highlighted series (**dpt03_dvt_jjp/10** in the illustration above) is the version specified as the source of the patch.

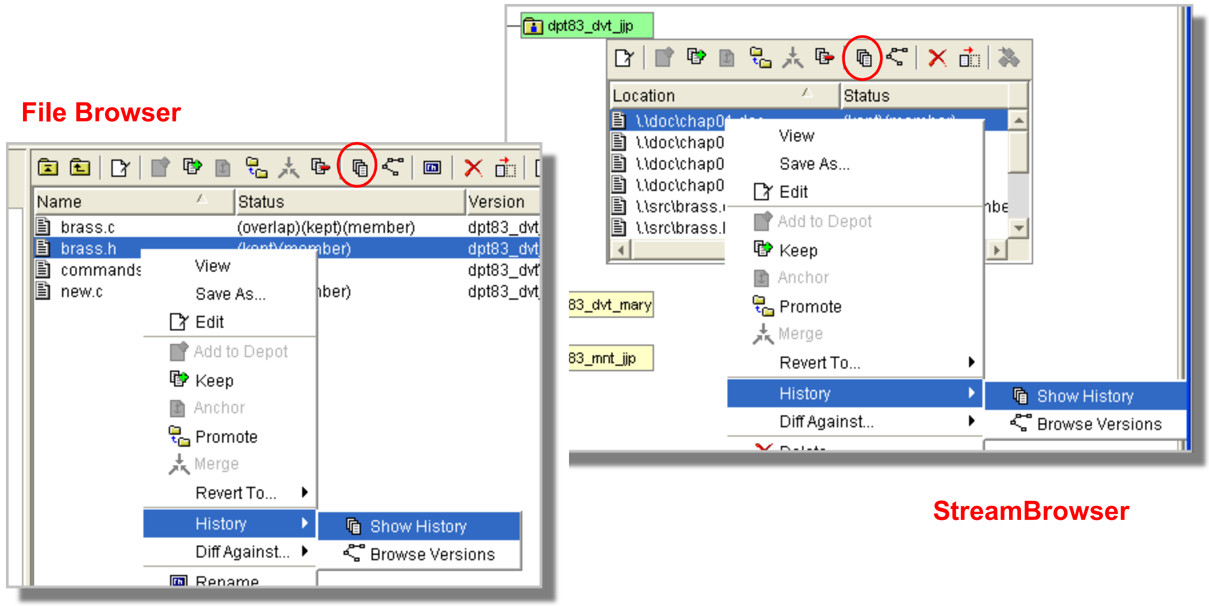
The History Browser

The AccuRev GUI includes a History Browser, with which you can view some or all of the transactions associated with a particular element, stream, or depot. This tool has been significantly enhanced for this release.

Invoking the History Browser

There are several ways to launch the History Browser:

- **Transactions Involving an Individual Element.**You can have the History Browser display the transactions in which one particular element was involved. (Other elements may have been involved in these transactions, too.) Element names are listed in the details pane of a File Browser tab; they are also listed in a stream's Default Group Contents display in the StreamBrowser.



  In either of these contexts, you can select an element and click the **Show History** toolbar button. Alternatively, right-click an element and select **History > Show History** from the context menu.

- **Transactions Involving a Particular Stream.** You can have the History Browser display transactions that affected a particular stream. For a workspace stream, this principally includes the **keep** transactions that create real versions in the stream. It can also include **co**, **move**, **purge**, **defunct**, and **undefunct** transactions. For a dynamic stream, this includes **promote** transactions *to* the stream, but not promotions *from* the stream.

  In the StreamBrowser, right-click a stream and select **Show History** from the context menu.

  (For a workspace stream, you may need to click the **Show Workspaces** button first.)

- **Active Transactions for a Particular Stream.** Every stream has a default group, consisting of the elements that are active in that stream. In a workspace stream, an element typically becomes active when a new version is created in a **keep** transaction. (Other commands, such as **mv** and **anchor**, also activate an element.) In a dynamic stream, elements become active when versions are sent to the stream in **promote** transactions.

  At any given moment, a certain set of versions are active in a given stream. In the StreamBrowser, you can select **View Default Group** from a stream's context menu to display these versions. Another context-menu item, **Show Active Transactions**, opens a History Browser tab and loads the set of transactions (**keep**, **promote**, etc.) in which those versions were created.

- **Transactions Involving a Particular Depot.** You can view the transactions for an entire depot — all the elements, in all the streams. Use the **View > Depots** command or **View Depots** toolbar button to list all of the repository's depots. Then, select a depot and choose **Show History** from the context menu.

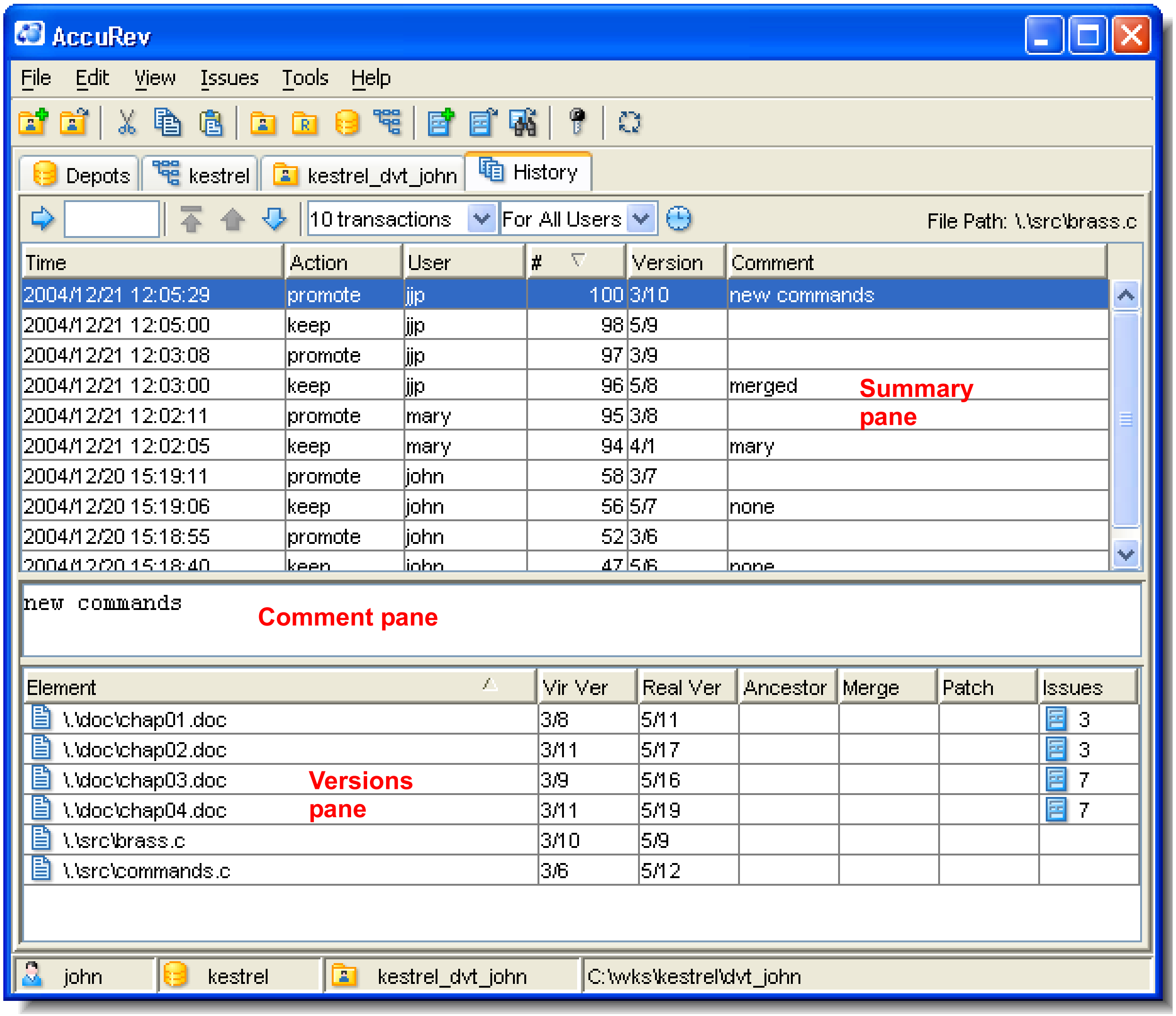
There are also contexts in which you can invoke the History Browser to view a single transaction, or a specific list of them, rather than the collections described above:

- In the Version Browser, selecting **Show History** from any version's context menu displays the transaction in which that particular version was created.
- If you've used the transaction-level integration between configuration management and issue management, an issue record's **affectedFiles** field contains a list of transaction numbers. Clicking the **Show History** button next to this field displays just those listed transactions.

The History Browser Display

The History Browser appears in a separate tab of the AccuRev GUI's multiple-tab display. The tab is divided into three panes:

- The **Summary** pane displays a group of transactions, one per line. This pane displays overall information: transaction number, timestamp, transaction type, etc.
- The **Comment** pane shows the comment string, if any, that was specified for the currently-selected transaction.
- The **Versions** pane shows all the versions that were involved in the currently-selected transaction. It also indicates which change packages, if any, those versions belong to.



File Path: \.\src\brass.c

Time	Action	User	#	Version	Comment
2004/12/21 12:05:29	promote	jjp	100	3/10	new commands
2004/12/21 12:05:00	keep	jjp	98	5/9	
2004/12/21 12:03:08	promote	jjp	97	3/9	
2004/12/21 12:03:00	keep	jjp	96	5/8	merged
2004/12/21 12:02:11	promote	mary	95	3/8	
2004/12/21 12:02:05	keep	mary	94	4/1	mary
2004/12/20 15:19:11	promote	john	58	3/7	
2004/12/20 15:19:06	keep	john	56	5/7	none
2004/12/20 15:18:55	promote	john	52	3/6	
2004/12/20 15:18:40	keep	john	47	5/6	none

Summary pane

new commands

Comment pane

Element	Vir Ver	Real Ver	Ancestor	Merge	Patch	Issues
\.\doc\chap01.doc	3/8	5/11				3
\.\doc\chap02.doc	3/11	5/17				3
\.\doc\chap03.doc	3/9	5/16				7
\.\doc\chap04.doc	3/11	5/19				7
\.\src\brass.c	3/10	5/9				
\.\src\commands.c	3/6	5/12				

Versions pane

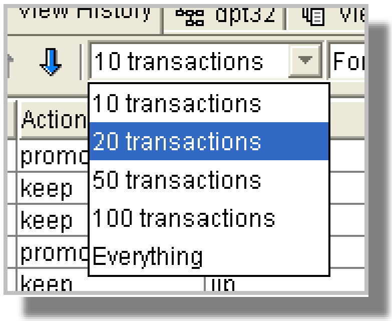
john | kestrel | kestrel_dvt_john | C:\wks\kestrel\dvt_john

The Summary Pane

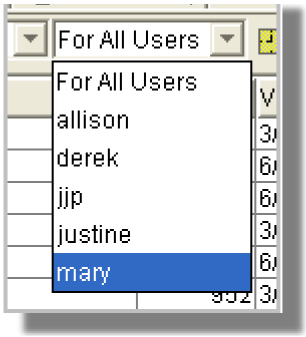
The Summary pane displays a table containing a set of transactions, one per row. You can manipulate the table in the standard ways: adjusting column widths, rearranging columns, sorting the rows on one or more columns.

There's potentially a large number of transactions to display in this table — e.g. when displaying the history of an entire depot. Retrieving all the transactions at once from the depot's database can be time-consuming. The History Browser handles this situation by initially retrieving a small number of transactions, then letting you control the retrieval of additional transactions with the toolbar located just above this table.

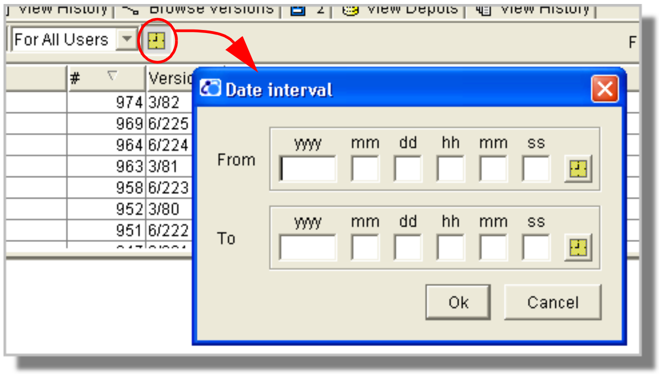
The table initially contains the 10 most recent relevant transactions. To change this count, use the Transaction Count listbox. (Each time you change the count, the browser returns to displaying the most recent transactions.) Selecting **Everything** loads all the relevant transactions into the Summary pane.



By default, the browser displays transactions created by any user. To restrict the display to one user's transactions, use the User Filter listbox.



Use the ⏲**Set Date Interval** toolbar button to restrict the display to transactions that occurred in a specified interval.



Once a certain set of transactions has been loaded into the Summary pane, you can browse through those transactions. Click any transaction to select it; the Comment and Versions panes are updated with the transaction's details. You can use the usual navigation keys to change the selected transaction: up-arrow and down-arrow, PgUp and PgDN, Ctrl-PgUp and Ctrl-PgDn.

Several controls on the toolbar enable you to retrieve additional transactions, which have not yet been loaded into the Summary pane:

- The ⇩ button unloads the current set of transactions, then loads the next older set of transactions.
- The ⇧ button unloads the current set of transactions, then loads the next newer set of transactions. This button is enabled only when you're viewing the history of an entire depot.
- The ⤒ button unloads the current set of transactions, then loads the first (newest) set of transactions.

1509

- Entering a transaction number in the Goto field unloads the current set of transactions, then loads the set beginning with the specified number. (If that particular transaction is not relevant — e.g. it did not involve the element whose history you're viewing — the set begins with the next older relevant transaction.)

For all of these controls, the Transaction Count, User Filter, and Date Interval settings determine exactly how many transactions (and which ones) are retrieved from the depot's database and loaded into the Summary pane.

Each row of the Summary pane's table displays the following information about an individual transaction:

Time

A timestamp, indicating when the transaction took place.

Action

The kind of transaction: **keep**, **promote**, etc.

User

The principal-name of the user who initiated the transaction.

# (transaction number)

The unique number (within this depot) of the transaction.

Version

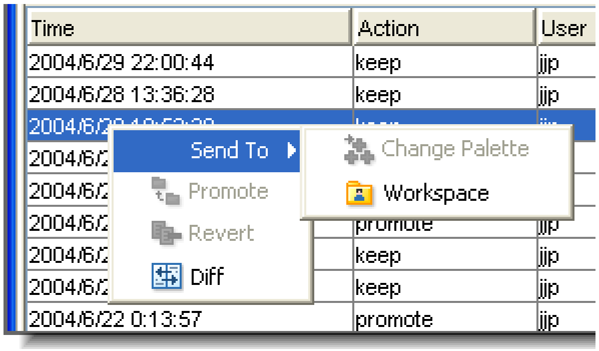
(only for transactions involving an individual element) The real version or virtual version of the element that was created in this transaction. This column doesn't appear in displays of a stream's history or an entire depot's history; the version(s) created by the transaction appear in the Versions pane.

Comment

The first line of the user-supplied comment for this transaction. If the comment extends to additional lines, an ellipsis ("dot dot dot") appears here. For the full text of the comment, look in the Comment pane.

Operations on Transactions in the Summary Pane

You can perform several operations on a selected transaction in the Summary pane, using its context menu.



Time	Action	User
2004/6/29 22:00:44	keep	jjp
2004/6/28 13:36:28	keep	jjp
2004/6/2		
2004/6/2		
2004/6/2		
2004/6/2	promote	jjp
2004/6/2	keep	jjp
2004/6/2	keep	jjp
2004/6/22 0:13:57	promote	jjp

Send to Change Palette

(from dynamic stream history only) Record each of the transaction's versions in the Change Palette, for promotion to another stream.

Send to Workspace

Activate each of the transaction's versions in a particular workspace. AccuRev prompts you to specify one of your workspaces.

Promote

Promote all of the transaction's versions to the parent stream of the stream/workspace from which you invoked the History Browser.

Revert

(from dynamic stream history only) For the selected **Promote** transaction, performs the equivalent of an "undo" in a particular child workspace of the dynamic stream. AccuRev prompts you to specify which workspace. Older versions of the transaction's elements will be activated in that workspace. For details, see the **revert** reference page in the *AccuRev User's Guide (CLI)*.

Diff

(from text-file element history only) Compare the version in the selected transaction with the version in another transaction. AccuRev prompts you to select another transaction from the Summary pane.

The Comment Pane

The Comment pane displays the full text of the user-supplied comment for this transaction.

The Versions Pane

The Versions pane displays a table of all the versions, if any, created by the currently-selected (highlighted) transaction. Both the real version and virtual version IDs are listed. (For transactions in a workspace, these IDs are the same; for transactions in a dynamic stream, they differ.)

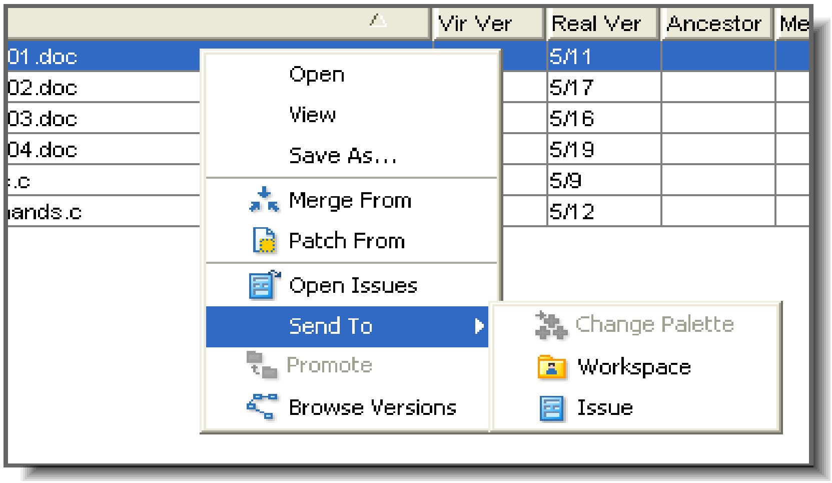
For transactions in a workspace, the Ancestor column indicates the version that the newly created version was derived from. For a **keep** transaction that was performed by a **Merge** command, the Merge column indicates the "from" version in the merge.Similarly, the Patch column indicates the "from" version for a **keep** transaction that was performed by a **Patch** command.

The Issues column, introduced in Version 3.5.5, indicates the change package(s) to which each version belongs.

You can manipulate this table in the standard ways: adjusting column widths, rearranging columns, sorting the rows on one or more columns.

Operations on Versions in the Versions Pane

You can perform several operations on a selected version in the Versions pane, using its context menu.



Open

Run the appropriate command on the file, according to its file type

View

Open a text editor on a temporary copy of the currently selected file (text files only).

Save As

Copy the currently selected file to another filename.

Merge From

(from workspace history only) Merge the selected version into the version in the workspace from which the History Browser was invoked.

Patch From

(from workspace history only) Patch the selected version into the version in the workspace from which the History Browser was invoked.

Open Issues

Displays the change package(s) — that is, the issue record(s) — to which the version belongs. For a version that belongs to a single change package, an edit form opens on that one issue record. For a version that belongs to multiple change packages, an Issues tab opens, listing all the issue records.

Send to Change Palette

(dynamic stream only): Load the selected versions into the Change Palette, so that they can be promoted to other streams.

Send to Workspace

(from workspace history only) Activate the selected element in the workspace from which the History Browser was invoked.

Send to Issue

Record the selected version in the change package section (Changes tab) of a particular issue record.

Promote

Promote the selected version to the parent of the stream/workspace from which the History Browser was invoked.

Browse Versions

Open a Version Browser on the element whose version you selected.

The Version Browser: Ancestry Tracking

AccuRev maintains complete ancestry information for each element, keeping track of how each version of the element was created. There are four possibilities:

ancestor

Modifying an existing version, then keeping the results to create a new real version.

alias

Promoting an existing version, to create a new virtual version.

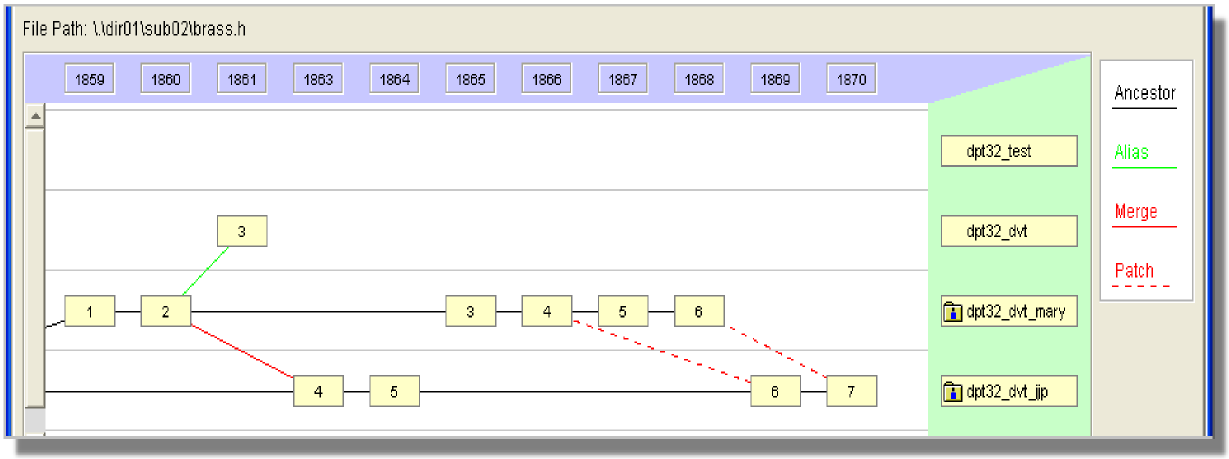
merge

Merging two versions, then keeping the results to create a new real version.

patch

Incorporating a subset of the changes made in one version into another version, then keeping the results to create a new real version.

The **Version Browser** can display some or all of an element's versions, using color-coded lines to indicate the way in which each version was created.



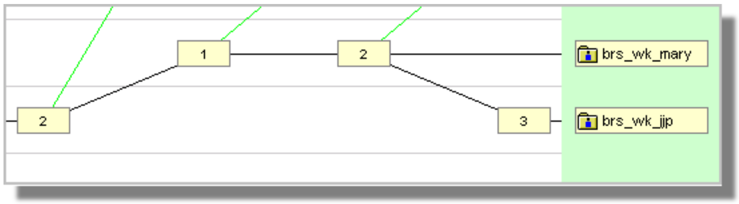
File Path: \.\dir01\sub02\brass.h

The following sections discuss the four kinds of ancestry in more detail, along with the important concept of closest common ancestor.

Ancestor — Modification of an Existing Version

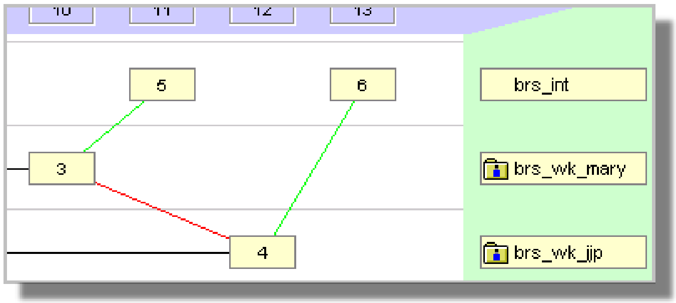
Probably the most common operation in AccuRev is starting with an existing version (created by you or by someone else), making changes, and then **keep**ing the changes. This creates a new real version, whose direct ancestor is the real version you started with. In the Version Browser, a black line connects the two real versions.

In the figure at right:



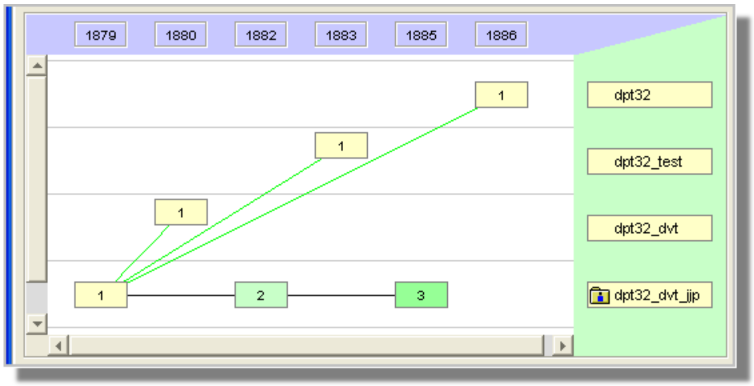
- Version 2 in the **brs_wk_jjp** stream was edited to create version 1 in the **brs_wk_mary** stream.
- Version 1 in the **brs_wk_mary** stream was edited to create version 2 in the same stream.
- Version 2 in the **brs_wk_mary** stream was edited to create version 3 in the **brs_wk_jjp** stream.

Alias — Virtual Version Ancestry



Virtual versions are created principally with the **promote** command. (A few other commands, such as **co** and **mv**, also create virtual versions.) Each virtual version is an alias for — that is, another name for — some real version. In the Version Browser, a green line connects a virtual version to the corresponding real version.

In the figure above, version 5 in the **brs_int** stream is an alias for (was promoted from) version 3 in the **brs_wk_mary** stream. Similarly, version 6 in the **brs_int** stream is an alias for version 4 in the **brs_wk_jjp** stream.



In a depot with a deep stream hierarchy, it's common to successively promote a particular version to the parent stream, then to the grandparent stream, then to the great-grandparent stream, etc. All the versions created by this series of **promote**s are aliases for the same real version. The Version Browser shows how all the virtual versions relate back to the original real version. In the figure at right, the versions in streams **dpt32_dvt**, **dpt32_test**, and **dpt32** are all aliases for the real version in workspace stream **dpt32_dvt_jjp**. (The display does not indicate the fact that the version was promoted from **dpt32_dvt** to **dpt32_test**, and from **dpt32_test** to **dpt32**.)

Merge — Merging of Two Versions

A standard **merge** operation combines the contents of these two versions of a file:

- The most recently kept version in your workspace stream. This version contains the changes that you have made to the file in your workspace.
- The most recent version in the backing stream.

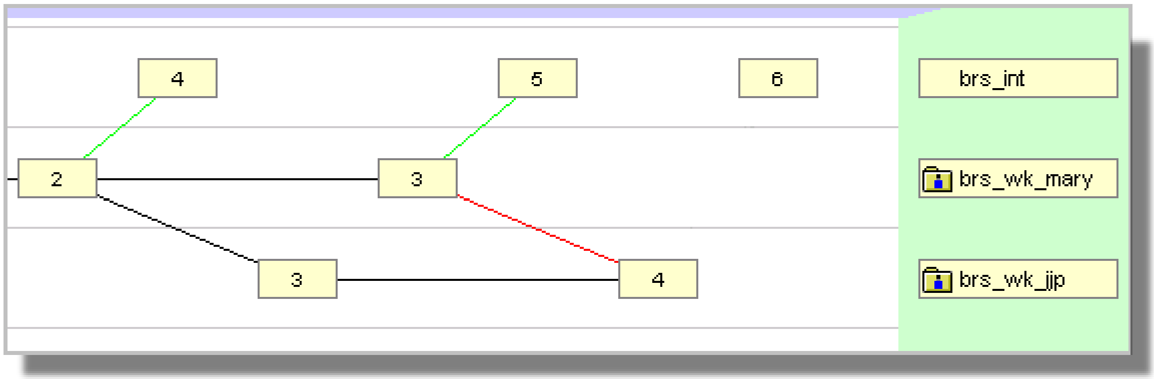
The result file of the merge operation is kept as a new version in the workspace stream. (You can think of merging as a fancy text-editing operation; as with any edit to a file, you preserve the results with **keep**.) This new, merged version has two ancestors: the two versions listed above.

This is all simple enough. There's a twist, though, which shows up in the Version Browser display: AccuRev always records real versions, not virtual versions, as the two ancestors of a new, merged version. Thus, the ancestors in the standard merge scenario described above are:

- The most recently kept version in your workspace stream.
- The version in some other workspace stream that was promoted to the backing stream, causing the overlap that necessitated the merge.

The screen shot below shows a merge from the backing stream **brs_int** to the workspace stream **brs_wk_jjp**. The new, merged version is **brs_wk_jjp/4**. Its ancestors are:

- Real version **brs_wk_jjp/3**.
- Real version **brs_wk_mary/3**, which was promoted to become virtual version **brs_int/5** in the backing stream.



A solid red line shows the merging of data from one stream, **brs_wk_mary** to a different stream, **brs_wk_jjp**. The black line ("direct ancestor") between versions 3 and 4 in the **brs_wk_jjp** stream reflects the viewpoint that merging is just a fancy text-editing operation, automating the creation of the next version of a file.

Closest Common Ancestor

It's instructive to follow all the black and solid-red lines in an element's Version Browser display. This traces the entire ancestry of real versions of an element. In particular, you can use the real-version ancestry to determine the closest common ancestor of any two versions. This is the most recent version upon which the two versions are both based, by some combination of ancestor and merge connections.

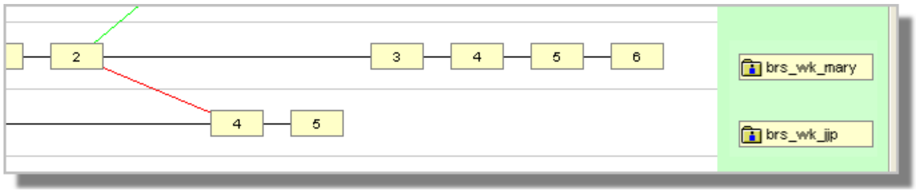
(When considering a virtual version in a closest-common-ancestor analysis, first follow the green line back to the corresponding real version.)

The **merge** command determines the closest common ancestor of the two versions to be merged, and uses this version to perform a 3-way merge. This merge algorithm evaluates each difference between the two versions as a *change* — in one or both versions — from the closest common ancestor.

Patch — Selective Merging of Two Versions

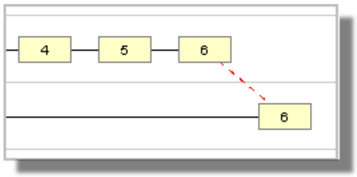
A patch operation is similar to a merge operation. In both, text from another version (the "from" version) is incorporated into your workspace's version. Here's the difference:

- A merge operation considers the entire contents of the "from" version.
- A patch operation considers only the parts of the "from" version that are changes from its immediate ancestor version

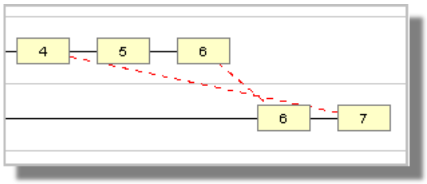


An example will clarify the distinction. Keep in mind that the merge algorithm considers differences to be changes from the closest common ancestor. Suppose user **jjp** wants to create a new version of a particular file in his workspace stream, **brs_wk_jjp**, by incorporating text from version 6 in stream **brs_wk_mary**.

Performing a merge would consider the entire contents of version **brs_wk_mary/6**. That is, it would consider the entire series of changes **mary** has made since version **brs_wk_mary/2**: the changes in all the versions 3, 4, 5, and 6.



By contrast, performing a patch from version **brs_wk_mary/6** incorporates only the changes made in that single version. It ignores the changes made in versions 3, 4, and 5. The Version Browser uses a dashed red line to indicate a patch. (Recall that a solid red line indicates a merge.)



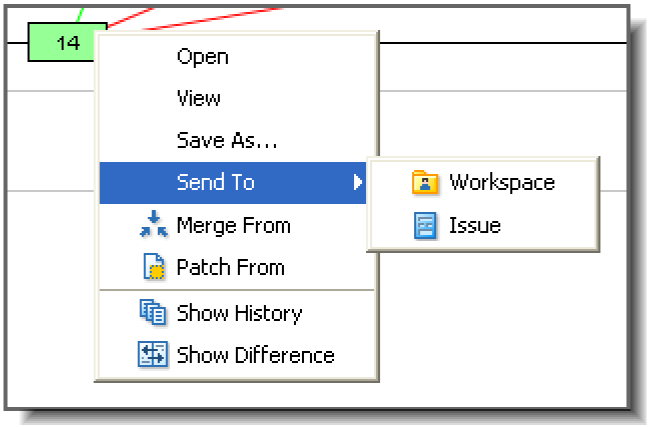
How can **jjp** incorporate the rest of **mary**'s changes? He can simply perform a merge, part of which harmlessly duplicates the patch operation. Or he can perform additional patch operations, incorporating **mary**'s change in any order. Here's what the Version Browser would display after patching from version **brs_wk_mary/4**.

> Note: AccuRev tracks patch ancestry separately from merge ancestry. In determining the closest common ancestor of two versions for a merge operation, AccuRev takes into account previous merge operations (solid red), but not previous patch operations (dashed red).

The **patchlist** command compares two versions of a text element; it expresses the difference as a list of versions that need to be patched into the second version, in order to incorporate all the changes that appear in the first version.

Operations on Versions

You can perform several operations on a selected version in the Versions pane, using its context menu.



Open

Run the appropriate command on the file, according to its file type

View

Open a text editor on a temporary copy of the currently selected file (text files only).

Save As

Copy the currently selected file to another filename.

Send to Workspace

Activate the selected element in the workspace from which the Version Browser was invoked.

Send to Issue

Record the selected version in the change package section (Changes tab) of a particular issue record.

Merge From

Merge the selected version into the version in the workspace from which the Version Browser was invoked.

Patch From

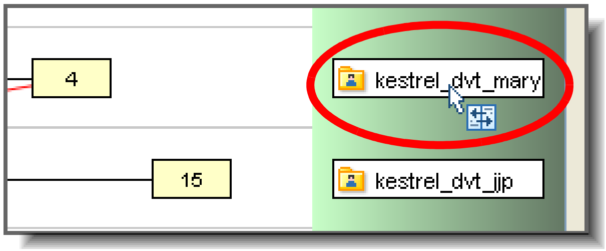
Patch the selected version into the version in the workspace from which the Version Browser was invoked.

Show History

Open a History Browser tab, showing the transaction that created the selected version.

Show Difference

Compare the selected version with another version of the element. AccuRev prompts you to select the other version. You can click on a stream or workspace label to indicate the version currently in that stream or workspace.



Using the Change Palette

Usually, the structure of a depot’s stream hierarchy determines how versions of an element will be propagated from stream to stream:

- A new version is originally created in some user’s workspace stream.
- Later, the version is promoted to the parent stream.
- Still later, the version is promoted from the parent stream to *its* parent.
- And so on, typically all the way up to the depot’s root stream.

At times, however, you may want to propagate changes directly to some stream other than the parent stream. When you want to go “outside the lines” of the depot’s stream hierarchy, the Change Palette is the tool to use.

Using the Change Palette, you can promote active versions from a dynamic stream to another dynamic stream. The Change Palette also detects the need for a merge prior to a desired promotion, and manages the carrying out of such merges. As of Version 3.5.5, you can also use the Change Palette to send an active version from a dynamic stream to a workspace. This is actually a **Send to Workspace** operation (CLI command: **co**), not a **Promote** operation.

You cannot use the Change Palette to promote versions from your workspace. Instead, you must use the **Promote** command to send such versions to the workspace’s backing stream. Also note that a lock can be placed on a dynamic stream, preventing promotion either *from* that stream or *to* that stream.

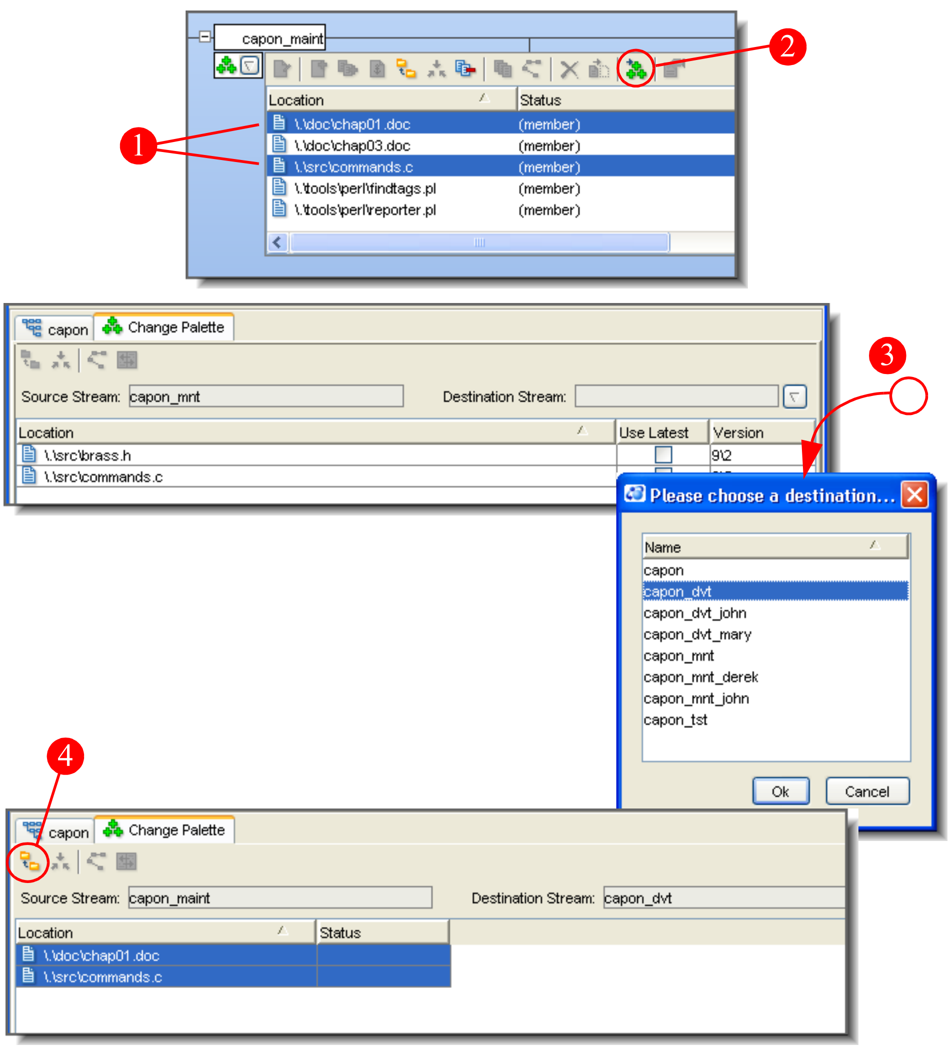
In the most common case, using the Change Palette is almost as simple as executing a **Promote** command:

1. Using the StreamBrowser, select the version(s) to be promoted from the Default Group of some dynamic stream.

2. Invoke the **Send to Change Palette** command.

3. In the Change Palette, specify the destination stream or workspace for the version(s).

4. Invoke the **Promote** command on the version(s). (If the destination is a workspace, the command is **Send to Workspace** instead of **Promote**.)



Location	Status
\.\doc\chap01.doc	(member)
\.\doc\chap03.doc	(member)
\.\src\commands.c	(member)
\.\tools\perl\findtags.pl	(member)
\.\tools\perl\reporter.pl	(member)

Source Stream: capon_mnt Destination Stream:

Location	Use Latest	Version
\.\src\brass.h	□	9\2
\.\src\commands.c	□	

Name
capon
capon_dvt
capon_dvt_john
capon_dvt_mary
capon_mnt
capon_mnt_derek
capon_mnt_john
capon_tst

Source Stream: capon_maint Destination Stream: capon_dvt

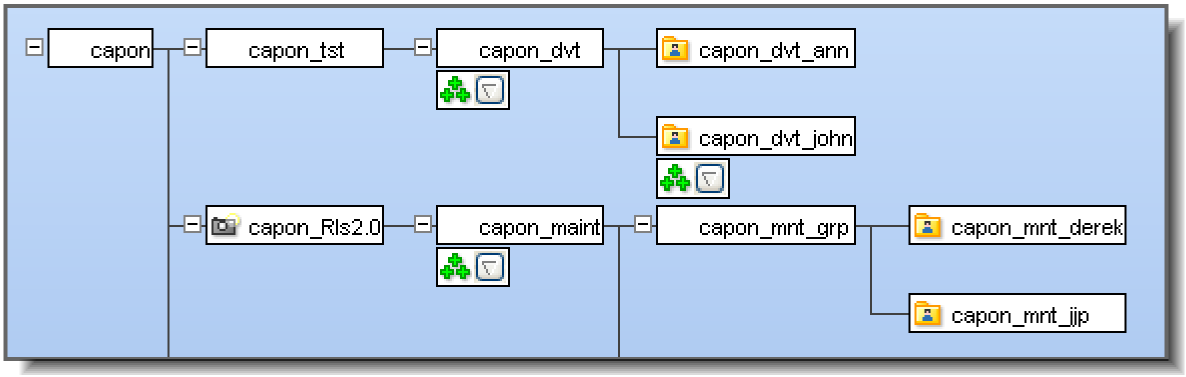
Location	Status
\.\doc\chap01.doc	
\.\src\commands.c	

If a version must be merged with the version in the destination stream prior to promotion, the Change Palette displays an **overlap** status flag. You can invoke the **Merge** command from the Change Palette, and also control which workspace the merge will take place in.

An **overlap** can also occur when you choose a workspace as the destination. In this case, instead of merging, you can choose to simply replace the version in the workspace with the version in the source stream.

If a workspace is the destination, you can choose to **Patch** from the selected version, instead of invoking **Send to Workspace** or **Merge**. The **Patch** command takes just the recent changes from the selected version, whereas **Send to Workspace** and **Merge** take all the changes.

The sections below describe all of the Change Palette's capabilities, covering all the use cases. The examples all come from the depot pictured here:



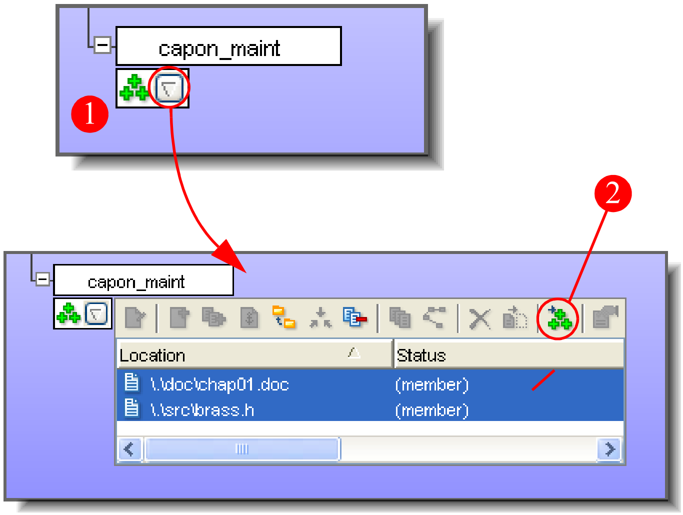
We'll show how to use the Change Palette to propagate changes directly ...

- ... from the **capon_maint** stream to the **capon_dvt** stream
- ... from the **capon_maint** stream to the **capon_dvt_ann** workspace
- ... from the **capon_dvt** stream to the root stream, **capon** (bypassing the **capon_tst** stream)

Sending Selected Versions to the Change Palette

Before using the Change Palette itself, you must specify the version(s) that are to be promoted "outside the lines" of the depot's stream hierarchy. One way is to select the versions in the StreamBrowser, File Browser, or History Browser, then invoke the **Send to Change Palette** command.

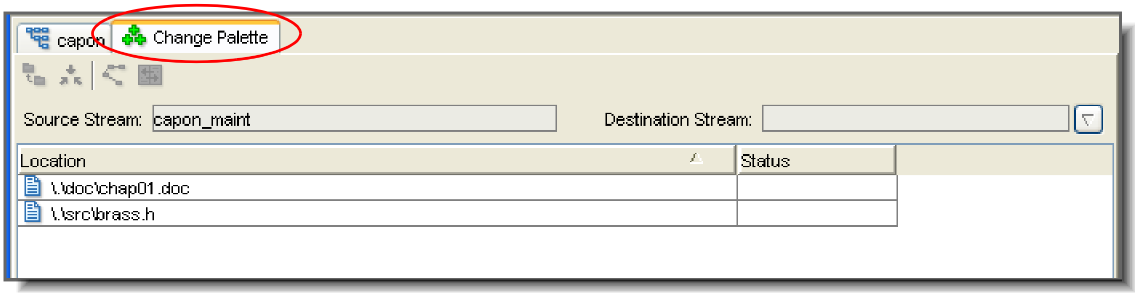
Only active versions in dynamic streams can be sent to the Change Palette. That is, a version must be in the default group of a dynamic stream to be eligible for promotion to another dynamic stream (or sending to a workspace) using the Change Palette. The easiest way to select such versions is with the StreamBrowser:



Location	Status
\.\doc\chap01.doc	(member)
\.\src\brass.h	(member)

1. To see the contents of any stream's default group, click the arrow that appears below the stream.
2. In the default-group listing, select the versions you wish to promote, then invoke the **Send to Change Palette** command, using the toolbar button or the context menu.

A Change Palette tab automatically appears in the AccuRev GUI window, with a line-item for each version you selected.



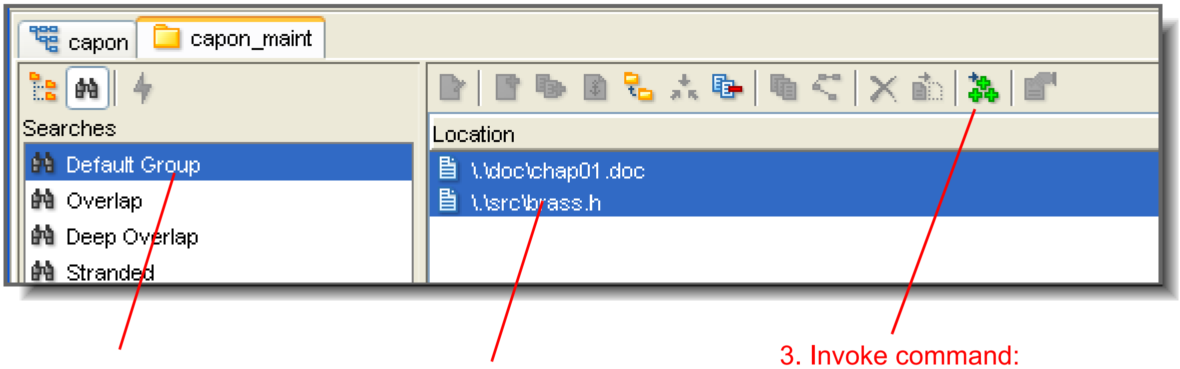
Source Stream: capon_maint

Destination Stream:

Location	Status
\.\doc\chap01.doc	
\.\src\brass.h	

You can also select the versions to send to the Change Palette using the File Browser:

1. To open a File Browser on a dynamic stream, double-click the stream in the StreamBrowser.
2. In the navigation pane of the File Browser, click the **Searches View** button, and select the Default Group search. Then, select versions and invoke the **Send to Change Palette** command as described above.



- Default Group
- Overlap
- Deep Overlap
- Stranded

Location
\.\doc\chap01.doc
\.\src\brass.h

1. Set Default Group search

2. Select files

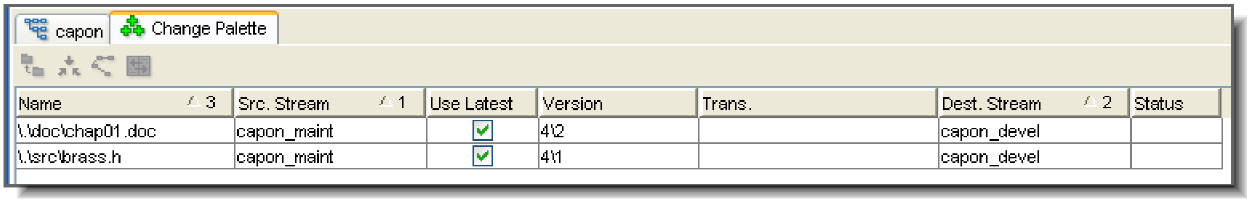
3. Invoke command: Send to Change Palette

Letting AccuRev Select the Versions to be Sent

There's another, particularly powerful way to populate the Change Palette. Instead of selecting individual versions from the Default Group of a stream, you can let AccuRev determine the *complete set* of versions in some source stream that have not yet been propagated to some destination stream or workspace:

1. In the StreamBrowser, right-click the source stream and select **Send to > Change Palette** from the context menu. (Or use the **Send to Change Palette** button in the StreamBrowser's toolbar.)
2. The mouse cursor changes to include the Change Palette icon. Click on another dynamic stream or a workspace to be the destination.

This populates the Change Palette with a line-item for each version that needs to be propagated to the destination stream or workspace.



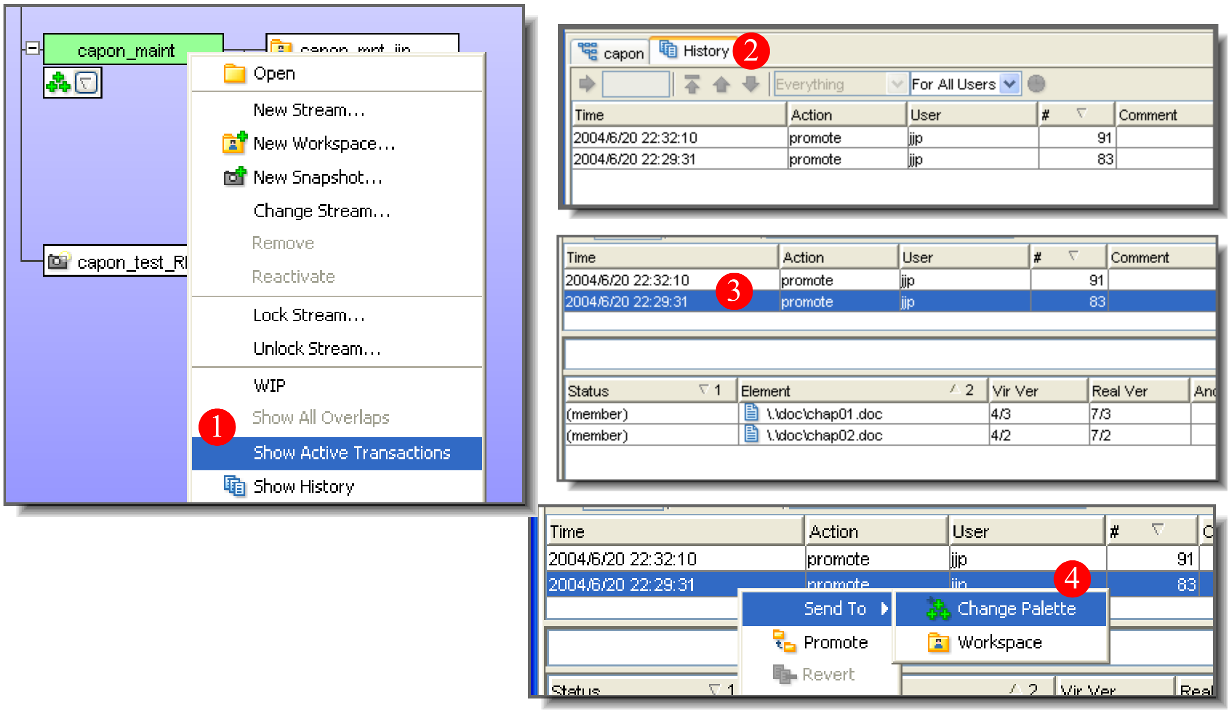
Name 3	Src. Stream 1	Use Latest	Version	Trans.	Dest. Stream 2	Status
\.\doc\chap01.doc	capon_maint	☑	4\2		capon_devel	
\.\src\brass.h	capon_maint	☑	4\1		capon_devel	

When you populate the Change Palette in this way, AccuRev automatically selects **Use Latest** for each version. A check in this column indicates that when you proceed to promote the element to the destination stream, AccuRev should use the version that is *currently* in the source stream. Given the indefinite time lag, this might be different from the version, listed in the Version column, that **Send to Change Palette** initially selected. Deselecting **Use Latest** for an element guarantees that the version listed in the Version column will be the one propagated to the destination stream.

> Note: in some cases, **Use Latest** propagates a version that is "in the source stream", but not "active in the source stream", to the destination stream. This occurs when the selected version gets promoted out of the source stream between the time you invoke **Send to Change Palette** and the time you invoke **Promote**. The version propagated to the destination stream is one that the source stream is inheriting from a higher-level stream at the time of the **Promote**.

Populating the Change Palette from the History Browser

Yet another way to send a stream's active versions to the Change Palette involves the History Browser. The command **Show Active Transactions** finds all the **Promote** transactions that "activated" a stream's currently-active versions; it opens a History Browser tab to display these transactions. For example:

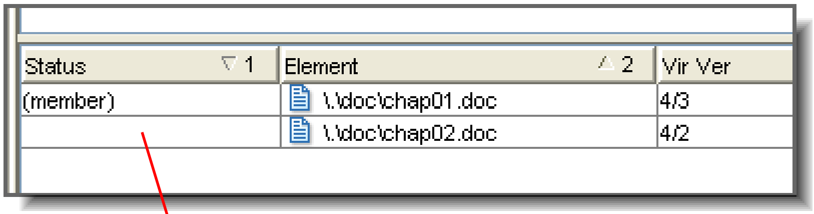


Time	Action	User	#	Comment
2004/6/20 22:32:10	promote	jjp	91	
2004/6/20 22:29:31	promote	jjp	83	

Status 1	Element 2	Vir Ver	Real Ver	Anc
(member)	\.\doc\chap01.doc	4/3	7/3	
(member)	\.\doc\chap02.doc	4/2	7/2	

1. Invoke the **Show Active Transactions** command on stream **capon_maint**.
2. A History Browser tab opens, showing the two transactions, #83 and #91, that promoted versions to **capon_maint** that are still active in the stream.
3. Select one of the transactions to see which versions it “activated” in the stream — in this example, version 4/3 of file **chap01.doc** and version 4/2 of file **chap02.doc**.
4. Use the context menu of one of the transactions to send all its active versions to the Change Palette. Alternatively, select one or more of the transaction’s versions in the lower pane, and use the context menu to send all the selected versions to the Change Palette.

Note: one or more of the versions in an “active transaction” may no longer be active in the stream. This indicates that the version has already been promoted to the parent stream. You cannot send such versions to the Change Palette. When all the versions in an active transaction have been promoted, the transaction will no longer appear in the Show Active Transactions listing. Note also that a version can become inactive through a purge (**Revert to Backed Version**) instead of a promotion.



Status ▽ 1	Element △ 2	Vir Ver
(member)	\.\doc\chap01.doc	4/3
	\.\doc\chap02.doc	4/2

no **(member)** status flag: this version of chap02.doc is no longer active in this stream

Working in the Change Palette

After you’ve sent a set of versions to the Change Palette, you can proceed to propagate them to the destination stream or workspace. Here’s a summary of the procedure:

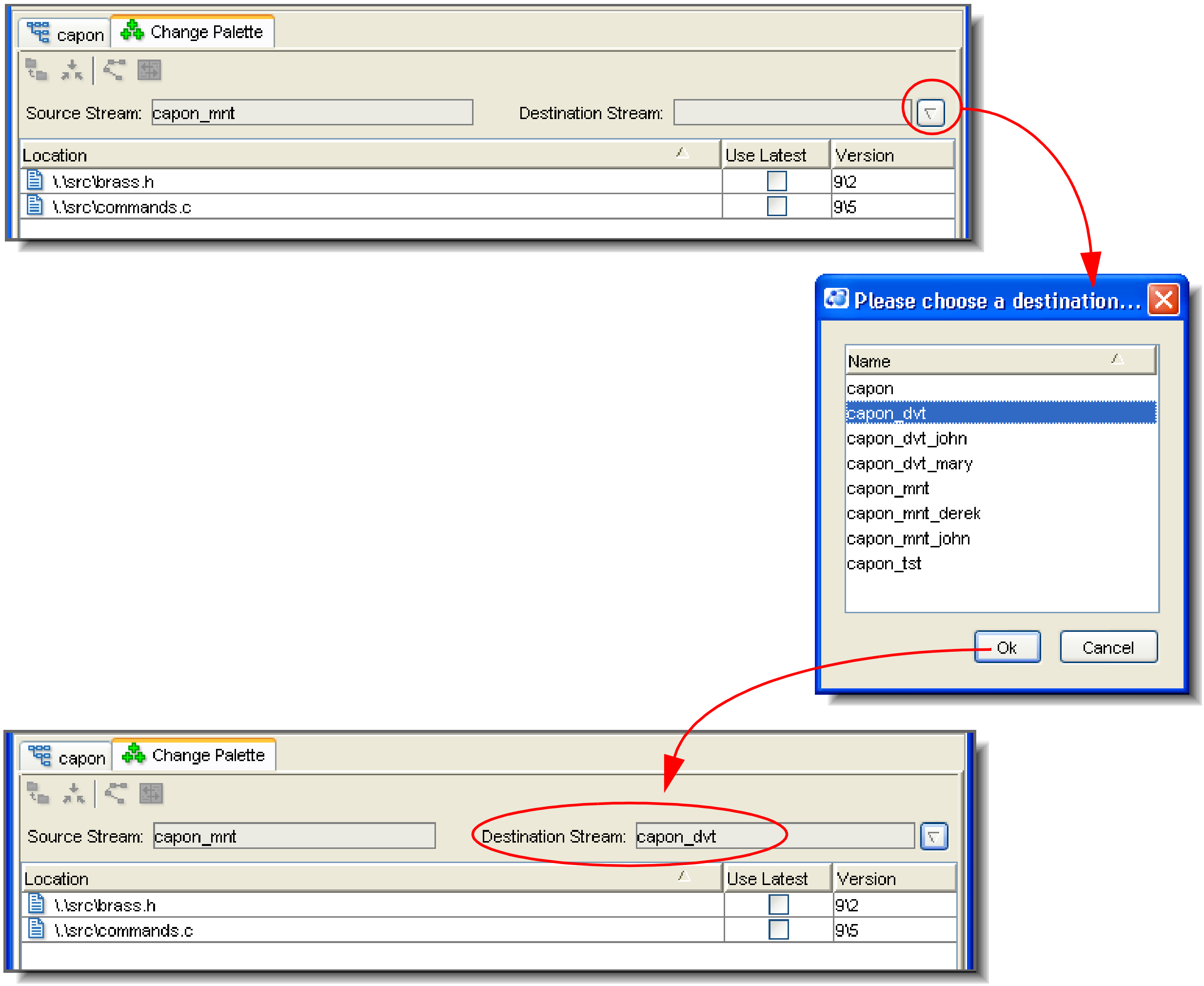
1. If necessary, specify the destination stream to which the version(s) are to be propagated.
2. If necessary, merge the source and destination versions of one or more of the elements.
3. Perform the **Promote** or **Send to Workspace** command. If the destination is a workspace, the **Patch** command is an alternative.

The following sections explore the details of this procedure.

Specifying the Destination Stream

> Note: you can skip this step if you used the procedure described in section *Letting AccuRev Select the Versions to be Sent* above, because you’ve already specified both the source and destination.

At this point, the **Source Stream** field shows the stream in which you invoked the **Send to Change Palette** command. Now, you must indicate a destination stream or workspace. Click the arrow control to the right of the **Destination Stream** field. A dialog box appears, containing a list of dynamic streams and workspaces; choose one of them to be the destination.



Source Stream: capon_mnt

Destination Stream:

Location	Use Latest	Version
\.\src\brass.h	□	9\2
\.\src\commands.c	□	9\5

Name
capon
capon_dvt
capon_dvt_john
capon_dvt_mary
capon_mnt
capon_mnt_derek
capon_mnt_john
capon_tst

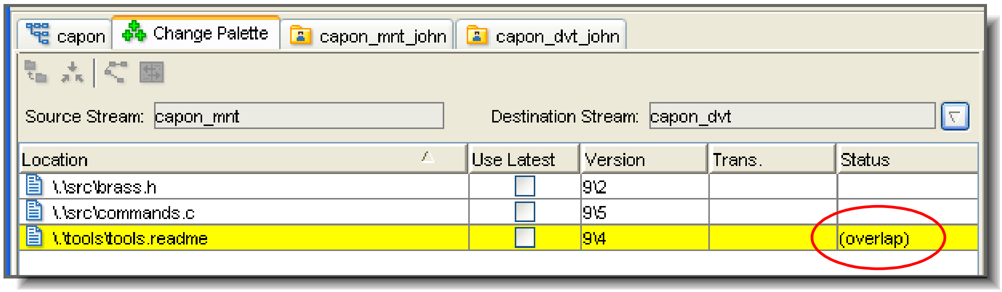
Source Stream: capon_mnt

Destination Stream: capon_dvt

Location	Use Latest	Version
\.\src\brass.h	□	9\2
\.\src\commands.c	□	9\5

Dynamic Stream Destination: Merging (If Necessary)

When you specify a dynamic stream as the destination, AccuRev determines for each element whether a merge is required between the source-stream version and the destination-stream version. If so, it places an **overlap** indicator in the line-item's **Status** column.



Source Stream: capon_mnt

Destination Stream: capon_dvt

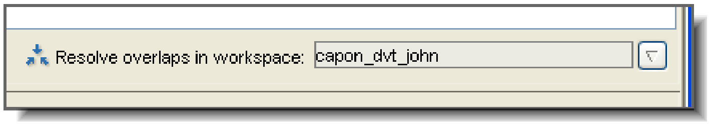
Location	Use Latest	Version	Trans.	Status
\.\src\brass.h	□	9\2		
\.\src\commands.c	□	9\5		
\.\tools\tools.readme	□	9\4		(overlap)

Before you can promote the source-stream version to the destination stream, you must resolve the overlap status by performing a merge. The **Merge** command combines the contents of the source-stream version with the contents of the destination-stream version, and creates a new version of the element with the combined ("merged") contents.

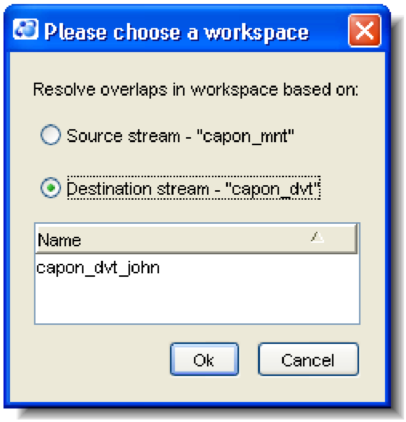
Selecting a Workspace for Performing Merges

Where does the merged version get created? It can't be either be in either the source or destination stream, because these are *dynamic* streams — all new versions of AccuRev elements must be originally created in a *workspace* stream. For the **Merge** command to proceed, a workspace belonging to you must be attached to either the source stream or the destination stream.

The first time you invoke **Merge** in the Change Palette, AccuRev prompts you to establish a merge workspace. Thereafter, it displays the workspace setting, which you can change at any time.



Resolve overlaps in workspace: capon_dvt_john



Please choose a workspace

Resolve overlaps in workspace based on:

○ Source stream - "capon_mnt"

● Destination stream - "capon_dvt"

Name
capon_dvt_john

What if *no* workspace exists that belongs to you and is attached to either the source or destination stream? In this case, you must click **Cancel** in the dialog box, and do either of the following:

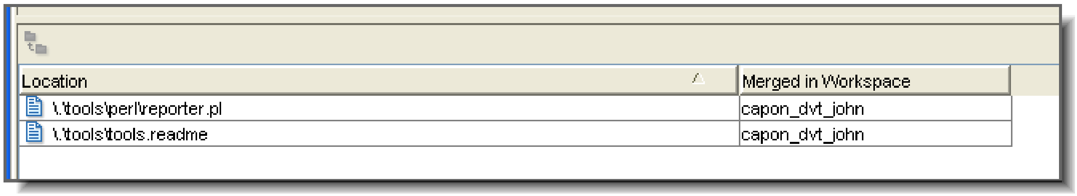
- Create a workspace attached to one of the streams. (In the StreamBrowser, right-click the stream and select **Create New Workspace** from the context menu.)
- Re-parent one of your existing workspaces: in the StreamBrowser, drag-and-drop the workspace from its current backing stream to the source or destination stream. Note that you should **Update** a reparented workspace before using it to perform Change Palette merge work.

Performing the Merges

To proceed with merging, select one or more of the overlap-status files in the Change Palette and click the **Merge** toolbar button. Alternatively, right-click the line-item(s) and select the **Merge** command from the context menu. You can merge the overlap-status files all at once or one at a time. Perform the merge(s) in the standard way. (See *Viewing and Resolving Content-Level Conflicts* on page 104.)

When you've finished merging the file, several things occur:

- The Merge tab disappears, so that the Change Palette tab appears again
- A new version of each merged file is created in the selected workspace; the new version contains the merged contents.
- A new pane opens at the bottom of the Change Palette tab, displaying these new workspace version(s).



Location	Merged in Workspace
\.\tools\perl\reporter.pl	capon_dvt_john
\.\tools\tools.readme	capon_dvt_john

At any time, you can promote one or more of these merged versions, as described in the next section.

Performing the Promotions

You can **Promote** one or more files from the Change Palette's upper pane or lower pane (but not both panes at once), using the **Promote** tool button or the files' context menu. Files in the upper pane can be promoted if they do not have overlap status. All files in the lower pane can be promoted.

> Note: In Version 3.5.5, files promoted from the upper pane go to the designated destination stream. Files promoted from the lower pane go to the backing stream of the workspace in which the merge was performed. Depending on the situation, this may be either the source stream or the destination stream.
>
> This means that when you merge using a workspace based on the source stream, you need to promote the merged version again — from the source stream to the destination stream — to achieve your original goal of propagating changes to the destination stream.

CLI Commands Related to the Change Palette

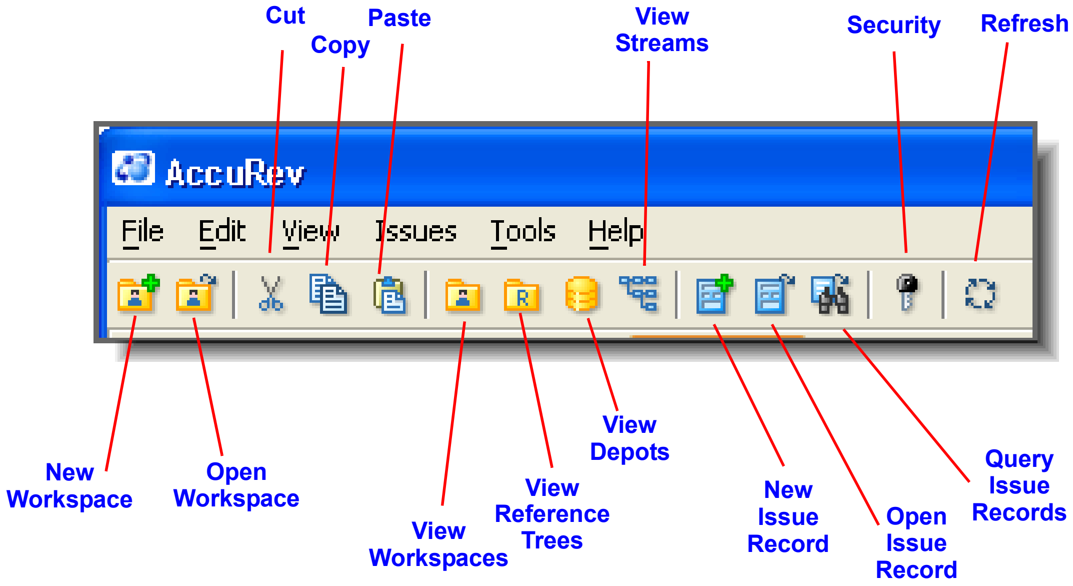
The following AccuRev CLI commands implement the various Change Palette capabilities described in this chapter:

- The **mergelist** command implements a stream's **Send to Change Palette** command in the StreamBrowser. (This is the command that determines *all* the changes that need to be propagated from one specified stream to another.)
- The **merge** command implements the **Merge** GUI command — both in the Change Palette and in the File Browser. The version in the "Resolve overlaps in ..." workspace is one of the merge contributors; the **–v** option specifies the other merge contributor.
- The **promote** command implements the **Promote** GUI command — both in the Change Palette and in the File Browser. The **–s** and **–S** options specify the source and destination dynamic streams.

AccuRev Graphical User Interface: Quick Reference

This chapter provides quick-reference summaries of the command menus and toolbars offered by the various tools that make up the AccuRev GUI.

GUI Window: Top-Level Toolbar

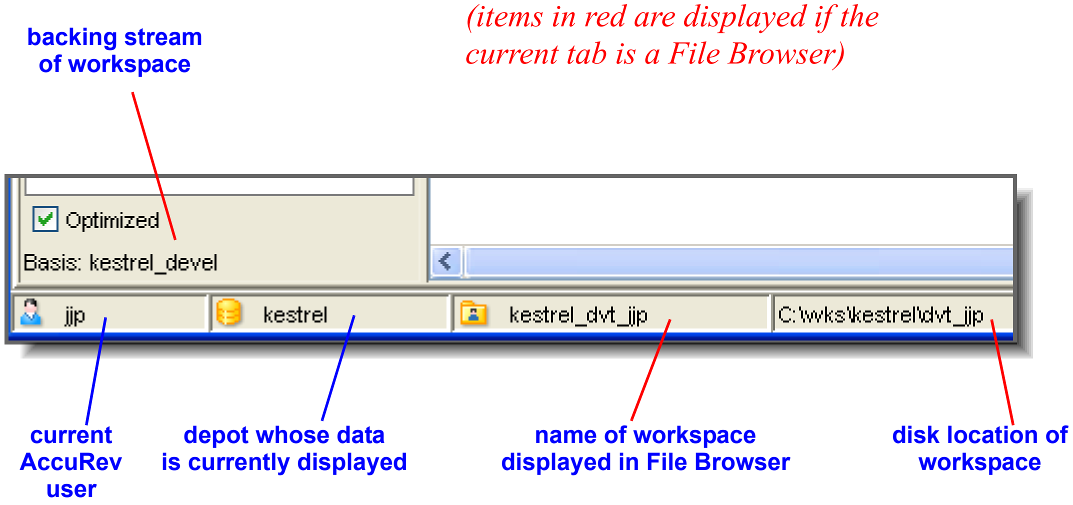


Toolbar Button	Description
New Workspace	Create a new workspace, based on an existing stream. CLI: **mkws**
Open Workspace	Select a workspace and open a File Browser to display its contents. CLI: change to workspace directory
Cut	Cut the selected text (use in Merge tool and Dispatch edit forms)
Copy	Copy the selected text (use in Merge tool and edit forms)
Paste	Paste cut/copied text at the current cursor location (use in Merge tool and Dispatch edit forms)
View Workspaces	List all your workspaces, or everyone's workspaces. From this list, you can open a workspace, deactivate ("remove") a workspace, or modify a workspace's specifications. CLI: **start**, **remove wspace**, **chws**

View Reference Trees	List the repository’s reference trees. From this list, you can open a reference tree, deactivate (“remove”) a reference tree, or modify a reference tree’s specifications. CLI: **start**, **remove ref**, **chref**
View Depots	List all the depots in the AccuRev repository. From this list, you can open a depot, display a depot’s transaction history, or rename a depot. CLI: **chdepot**, **hist**
View Streams	Display the hierarchy of streams, snapshots, and workspaces in the current depot. You can drag-and-drop to change the hierarchy, and you can perform a variety of operations on the streams, snapshots, and workspaces. See *The StreamBrowser* on page 153.
New Issue	Open an edit form to create a new Dispatch issue record in the current depot.
Open Issue	Retrieve an existing Dispatch issue record that is stored in the current depot.
Queries	Create and execute queries on the issue records stored in the current depot.
Security	Manage users, groups, and passwords, and access control lists (ACLs). CLI: **mkuser**, **chuser**, **remove user**, **mkgroup**, **addmember**, **remove group**, **setacl**, **lsacl**
Refresh	Update the screen to reflect changes that have occurred in the AccuRev data being displayed.

GUI Window: Overall Status Indicators

(items in red are displayed if the current tab is a File Browser)



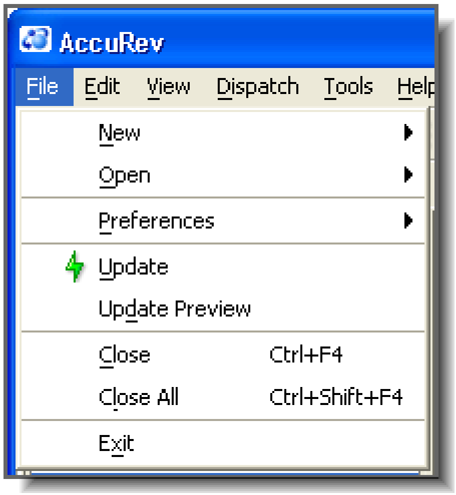
☑ Optimized

Basis: kestrel_devel

jjp | kestrel | kestrel_dvt_jjp | C:\wks\kestrel\dvt_jjp

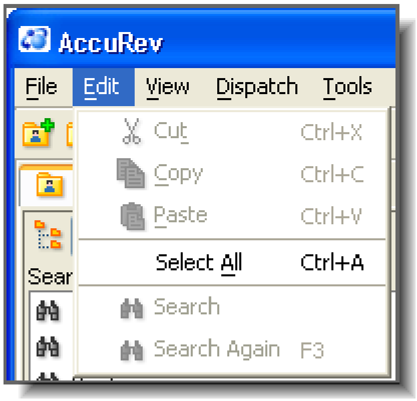
backing stream of workspace
current AccuRev user
depot whose data is currently displayed
name of workspace displayed in File Browser
disk location of workspace

GUI Window Main Menu: File Commands



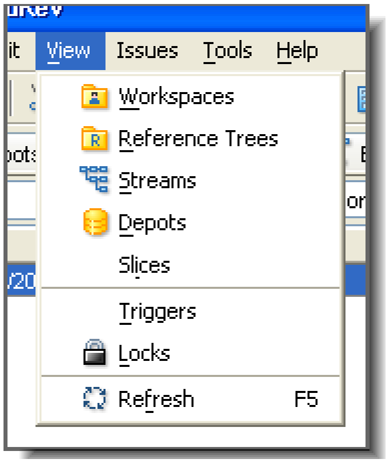
Command	Description
New	New Depot, New Workspace
Open	Open Depot, Open Workspace
Preferences	Set programs to be invoked instead of built-in Diff and Merge tools Enable the StreamBrowser's ability to scroll through the history of the stream hierarchy
Update	Update current workspace
Update Preview	Display list of elements that would be affected by an **Update** of current workspace
Close	Close the current tab.
Close All	Close all tabs in the GUI window.
Exit	End the AccuRev GUI program. A confirmation dialog box appears, to prevent you from exiting accidentally.

GUI Window Main Menu: Edit Commands

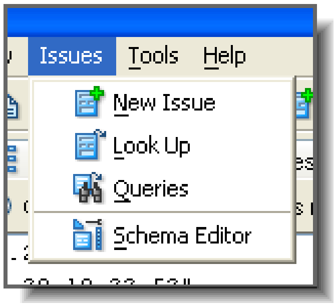


Command	Description
Cut	Cut the selected text (use in Merge tool and Dispatch edit forms)
Copy	Copy the selected text (use in Merge tool and Dispatch edit forms)
Paste	Paste cut/copied text at current cursor location (use in Merge tool and Dispatch edit forms)
Select All	Expand current selection to include all items.
Search	Search for text (Diff/Merge tool only).
Search Again	Repeat the last Diff/Merge search.

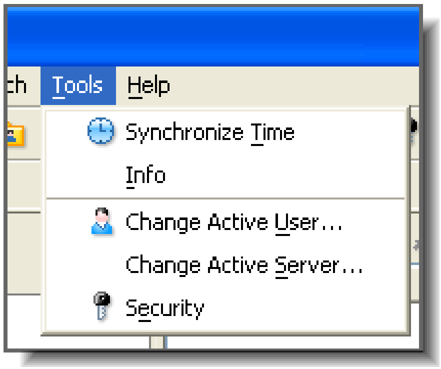
GUI Window Main Menu: View Commands

<table>
<tr><td rowspan="3"> </td><th>Command</th><th>Description</th></tr>
<tr><td>Workspaces</td><td>List all your workspaces, or everyone’s workspaces. From this list, you can open a workspace, deactivate (“remove”) a workspace, or modify a workspace’s specifications.
CLI: <b>start</b>, <b>remove wspace</b>, <b>chws</b></td></tr>
<tr><td>Reference Trees</td><td>List the repository’s reference trees. From this list, you can open a reference tree, deactivate (“remove”) a reference tree, or modify a reference tree’s specifications.
CLI: <b>start</b>, <b>remove ref</b>, <b>chref</b></td></tr>
<tr><td>Streams</td><td colspan="2">Display a list of all streams in the current depot. From this list, you can lock/unlock streams and display a stream’s transaction history
CLI: <b>show streams, hist</b></td></tr>
<tr><td>Depots</td><td colspan="2">List all the depots in the AccuRev repository. From this list, you can open a depot, display a depot’s transaction history, or rename a depot.
CLI: <b>chdepot</b>, <b>hist</b></td></tr>
<tr><td>Slices</td><td colspan="2">Display the disk location of each slice in the AccuRev repository.
CLI: <b>show slices</b></td></tr>
<tr><td>Triggers</td><td colspan="2">Display a list of the triggers defined for the current depot.
CLI: <b>show triggers</b></td></tr>
<tr><td>Locks</td><td colspan="2">Display a list of all locks defined for the current depot. From this list, you can remove locks.
CLI: <b>unlock</b>, <b>show locks</b></td></tr>
<tr><td>Refresh</td><td colspan="2">Update the screen to reflect changes that have occurred in the AccuRev data being displayed.</td></tr>
</table>

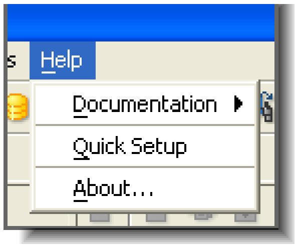
GUI Window Main Menu: Issues Commands (Dispatch)

	Command	Description
	New Issue	Open an edit form to create a new Dispatch issue record in the current depot.
	Look Up	Retrieve an existing Dispatch issue record that is stored in the current depot.
	Queries	Create and execute queries on the issue records stored in the current depot.
	Schema Editor	Specify the fields in the issues database for the current depot, and design the edit form.

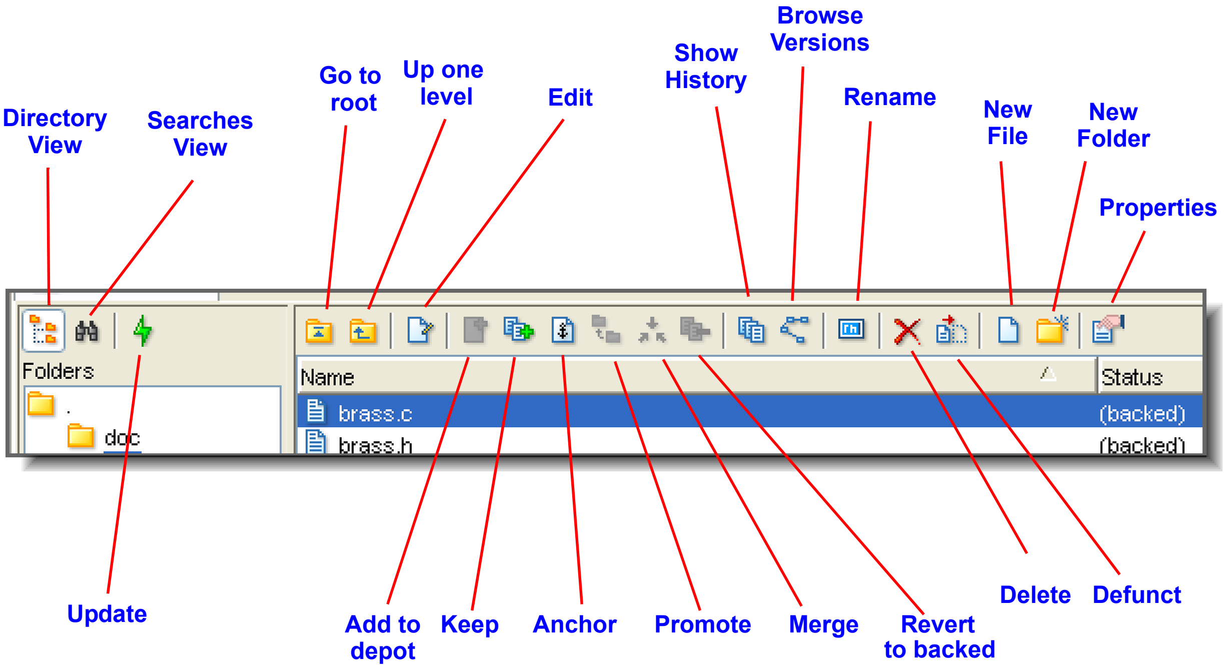
GUI Window Main Menu: Tools Commands

	Command	Description
	Synchronize Time	Change system clock on local machine to match clock on AccuRev server machine CLI: **synctime**
	Info	Display information on current AccuRev context CLI: **info**
	Change Active User	Switch to another principal-name CLI: ACCUREV_PRINCIPAL environment variable
	Change Active Server	Connect to another AccuRev server machine and its repository
	Security	Same as Security toolbar button (above)

GUI Window Main Menu: Help Commands

	Command	Description
	Documentation	Display the various books in the AccuRev documentation set (PDF format)
	Quick Setup	Launch a wizard that creates and populates a new depot
	About	Display program version information

The File Browser

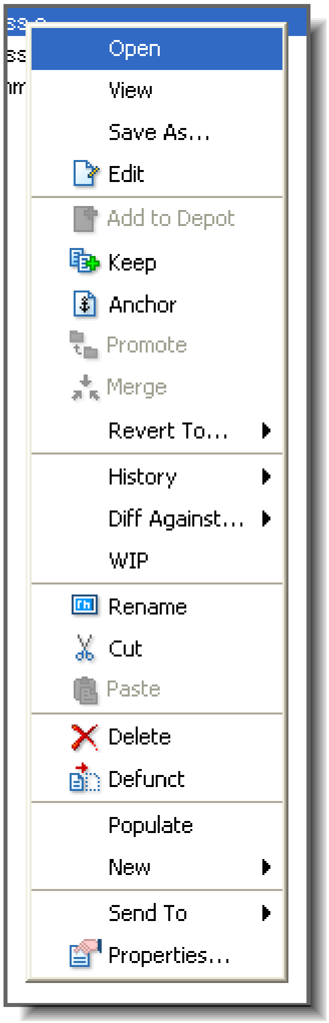


File Browser: Toolbars

Navigation Pane Buttons	**Description**
Directory View	Show the actual files and directories located in the workspace tree (in local disk storage). Optionally, shows elements that are **(missing)** from the workspace tree. CLI: **files**
Searches View	Show all files in the workspace tree that have a specified status. CLI: **stat**
Update	Run the **update** command for the current workspace.
Details Pane Buttons	**Description**
Go to root	Display the depot's top-level directory in the details pane.
Up one level	Go up one directory level in the details pane.
Edit	Open a text editor on the currently selected file (text files only).
Add to Depot	Run the **add** command on the currently selected file(s).
Keep	Run the **keep** command on the currently selected file(s).
Anchor	Run the **anchor** command on the currently selected file(s).

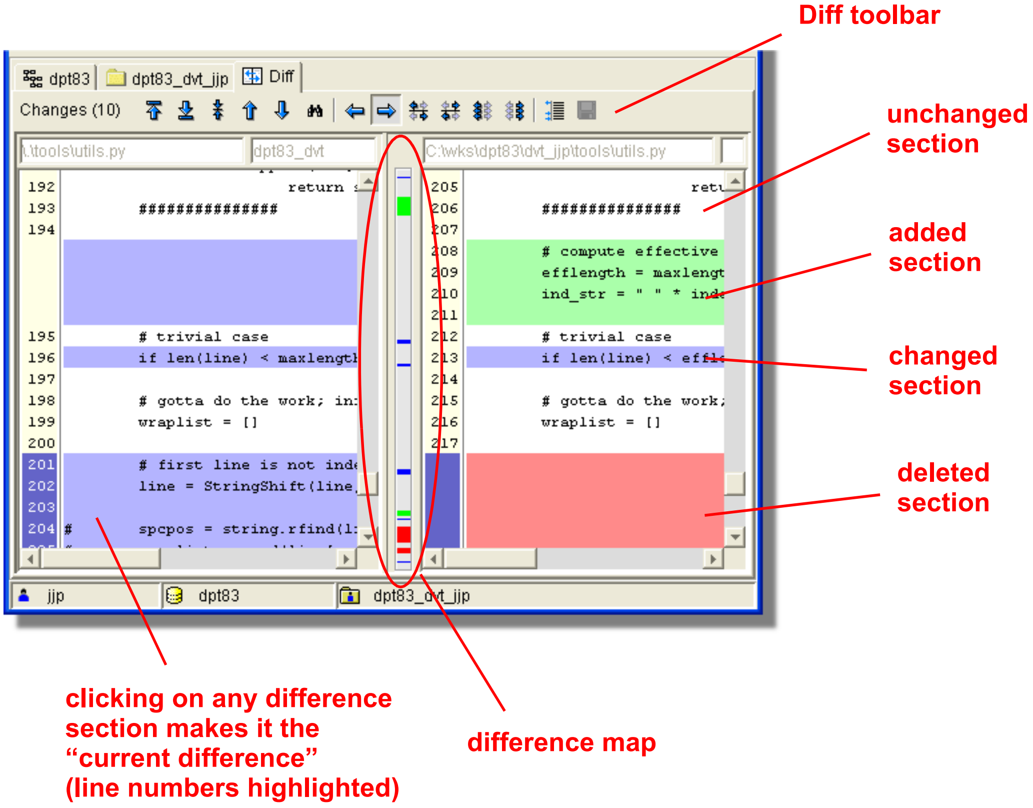
Promote	Run the **promote** command on the currently selected file(s). If the file(s) have modifications that have not yet been kept, performs a **keep** first.
Merge	Run the **merge** command on the currently selected file.
Revert to Backed	Run the **purge** command on the currently selected file(s).
Show History	Open a History Browser for the currently selected file.
Browse Versions	Open a Version Browser for the currently selected file.
Rename	Run the **move** command on the currently selected file.
Delete	Delete the currently selected file(s) from the workspace tree. This does not affect the workspace stream or anything else in the depot.
Defunct	Run the **defunct** command on the currently selected file(s).
New File	Create an empty new file, and optionally **add** it to the depot.
New Folder	Create an empty new folder (directory), and optionally **add** it to the depot.
Properties	Display the depot-relative pathname, the data-type of the most recent version, and the element-ID of the currently-selected element.

File Browser: Element’s Context Menu in Details Pane

<table>
<tr><td rowspan="11">
</td><th>Command</th><th>Description</th></tr>
<tr><td>Open</td><td>Run the appropriate command on the file, according to its file type.</td></tr>
<tr><td>View</td><td>Open a text editor on a temporary copy of the currently selected file (text files only).</td></tr>
<tr><td>Save As</td><td>Copy the currently selected file to another filename.</td></tr>
<tr><td>Edit</td><td>Open a text editor on the currently selected file (text files only).</td></tr>
<tr><td>Add to Depot</td><td>Run the <b>add</b> command on the currently selected file(s).</td></tr>
<tr><td>Keep</td><td>Run the <b>keep</b> command on the currently selected file(s).</td></tr>
<tr><td>Anchor</td><td>Run the <b>anchor</b> command on the currently selected file(s).</td></tr>
<tr><td>Promote</td><td>Run the <b>promote</b> command on the currently selected file(s).</td></tr>
<tr><td>Merge</td><td>Run the <b>merge</b> command on the currently selected file.</td></tr>
<tr><td>Revert To ...</td><td><b>Most Recent Version</b>: copy the version currently in the workspace stream to the workspace directory (like CLI <b>pop –O</b>).
<b>Backed Version</b>: remove all changes performed in the workspace, restoring the current backing stream version (like CLI <b>purge</b>).</td></tr>
<tr><td>History</td><td colspan="2">Open a History Browser (<b>Show History</b>) or a Version Browser (<b>Browse Versions</b>) for the currently selected file.</td></tr>
<tr><td>Diff Against</td><td colspan="2"><b>Most Recent Version</b>: compare the currently selected file to the version in your workspace stream. If the element is active, this is the last version you created with <b>keep</b>.
<b>Backed Version</b>: compare the currently selected file to the version in your workspace’s backing stream.</td></tr>
<tr><td>WIP</td><td colspan="2">Display work-in-progress for the currently selected elements.</td></tr>
<tr><td>Rename</td><td colspan="2">Run the <b>mv</b> command on the currently selected file.</td></tr>
<tr><td>Cut</td><td colspan="2">Mark the currently selected element to be moved to another directory in the same depot. To finish the relocation, right-click the destination directory (in either the navigation or details pane) and select <b>Paste</b>.</td></tr>
</table>

Paste	Specify the destination for an element that has been marked for relocation with the **Cut** command.
Delete	Delete the currently selected element(s) from the workspace tree. This does not affect the workspace stream or anything else in the depot.
Defunct	Run the **defunct** command on the currently selected element(s).
Populate	Run the **populate** command on the currently selected element. A dialog box lets you enable the overwrite and/or recursive option.
New	Create an empty new file or folder (directory), and optionally **add** it to the depot.
Send To	**Change Palette** (dynamic stream only): Load the selected versions into the Change Palette, so that they can be promoted to another stream. **Issue**: Record the selected versions in the change package section (Changes tab) of a particular issue record.
Properties	Display the depot-relative pathname, the data-type of the most recent version, and the element-ID of the currently-selected element.

The Diff Tool



```
                    return s
        ###############

        # trivial case
        if len(line) < maxlength

        # gotta do the work; in:
        wraplist = []

        # first line is not inde
        line = StringShift(line,

#       spcpos = string.rfind(l:
```

```
                    retu
        ###############

        # compute effective
        efflength = maxlengt
        ind_str = " " * ind

        # trivial case
        if len(line) < effl

        # gotta do the work;
        wraplist = []
```

Diff toolbar

unchanged section

added section

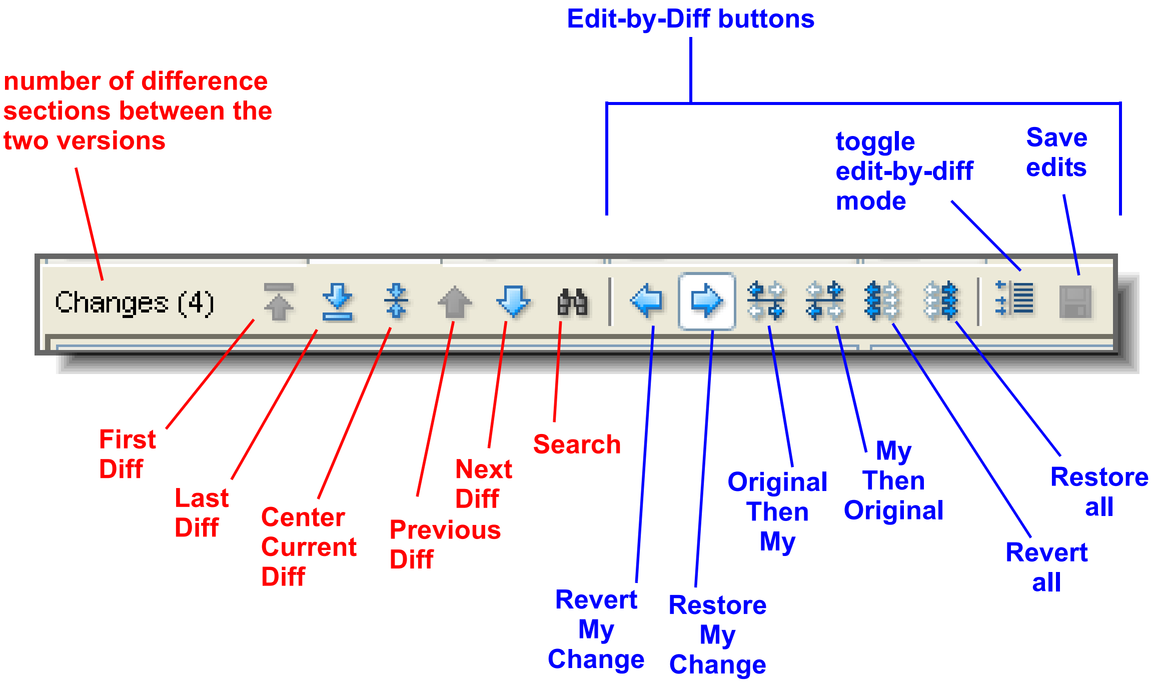
changed section

deleted section

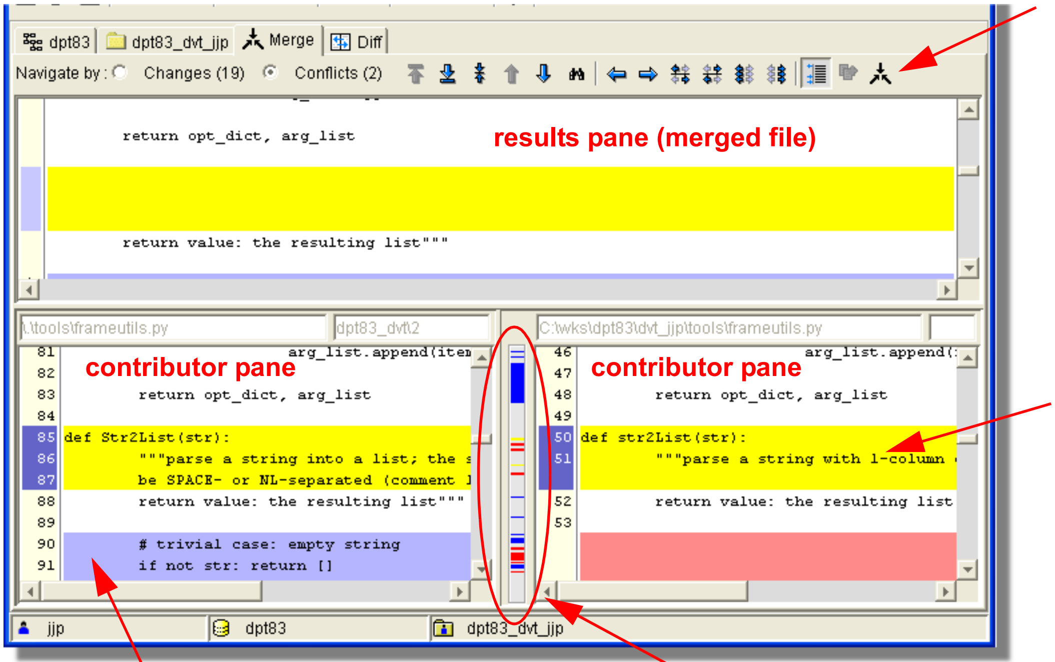
clicking on any difference section makes it the “current difference” (line numbers highlighted)

difference map

Diff Tool: Toolbar



The Merge Tool


dpt83
dpt83_dvt_jjp
Merge
Diff
Navigate by : ○ Changes (19) ● Conflicts (2)
\tools\frameutils.py
dpt83_dvt\2
C:\wks\dpt83\dvt_jjp\tools\frameutils.py
jjp
dpt83
dpt83_dvt_jjp

```
        return opt_dict, arg_list



        return value: the resulting list"""
```

```
                     arg_list.append(item
        return opt_dict, arg_list

def Str2List(str):
        """parse a string into a list; the s
        be SPACE- or NL-separated (comment l
        return value: the resulting list"""

        # trivial case: empty string
        if not str: return []
```

```
                     arg_list.append(:
        return opt_dict, arg_list

def str2List(str):
        """parse a string with 1-column
        return value: the resulting list
```

Merge toolbar

results pane (merged file)

contributor pane

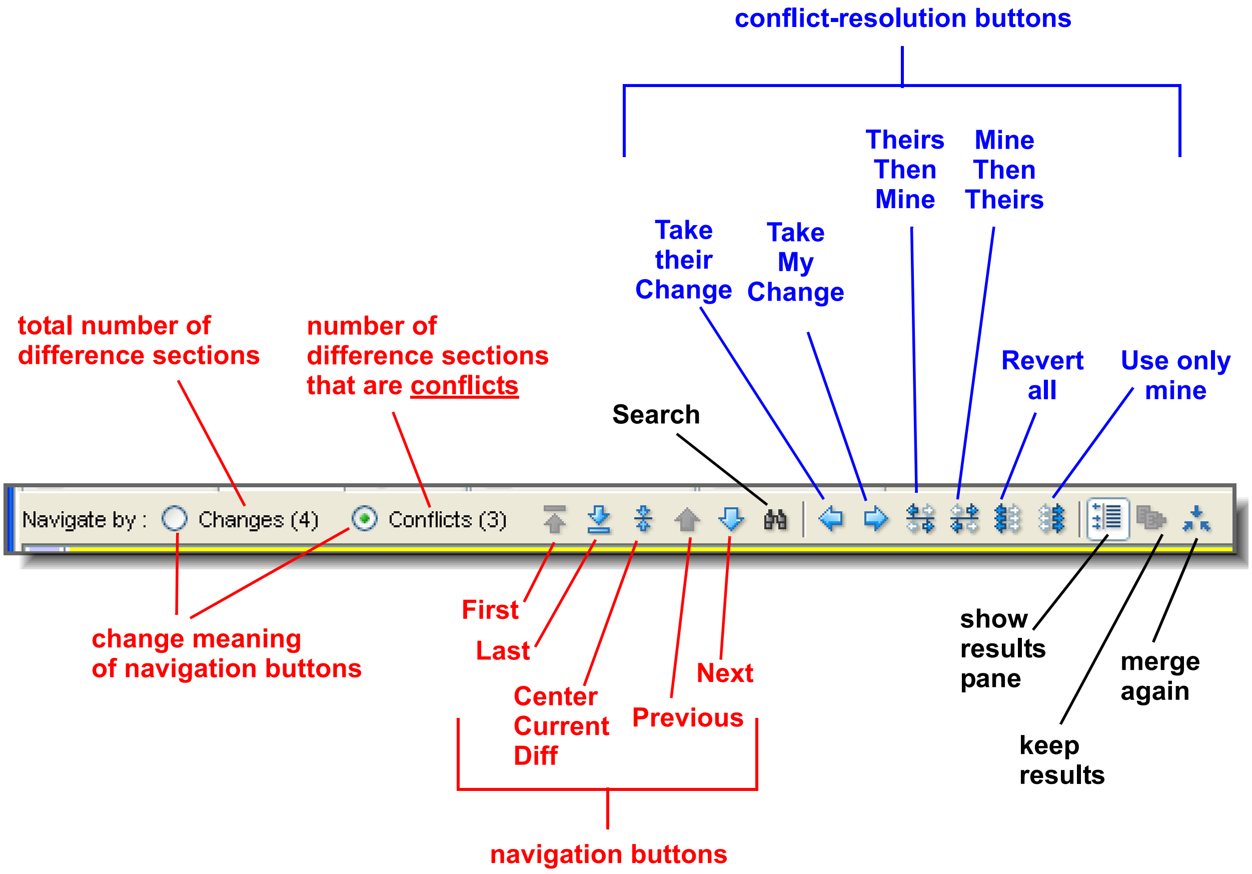
contributor pane

conflict section

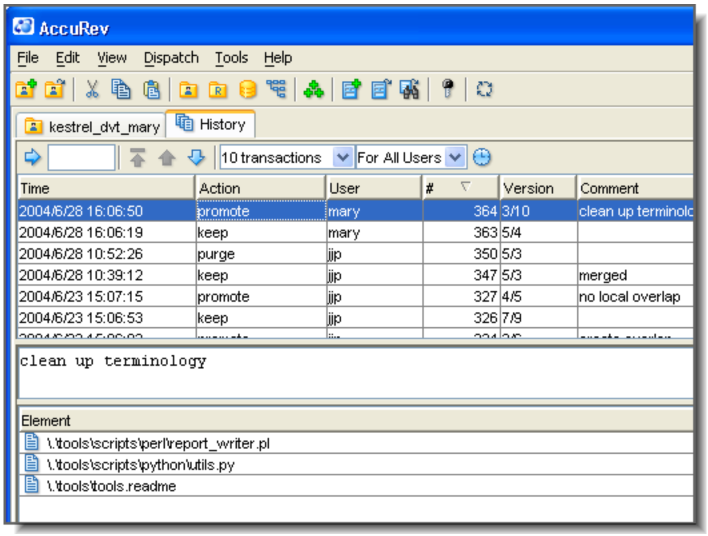
clicking on any difference section makes it the “current difference” (line numbers highlighted)

difference map

Merge Tool: Toolbar



The History Browser



Time	Action	User	#	Version	Comment
2004/6/28 16:06:50	promote	mary	364	3/10	clean up terminolc
2004/6/28 16:06:19	keep	mary	363	5/4	
2004/6/28 10:52:26	purge	jjp	350	5/3	
2004/6/28 10:39:12	keep	jjp	347	5/3	merged
2004/6/23 15:07:15	promote	jjp	327	4/5	no local overlap
2004/6/23 15:06:53	keep	jjp	326	7/9	

```
clean up terminology
```

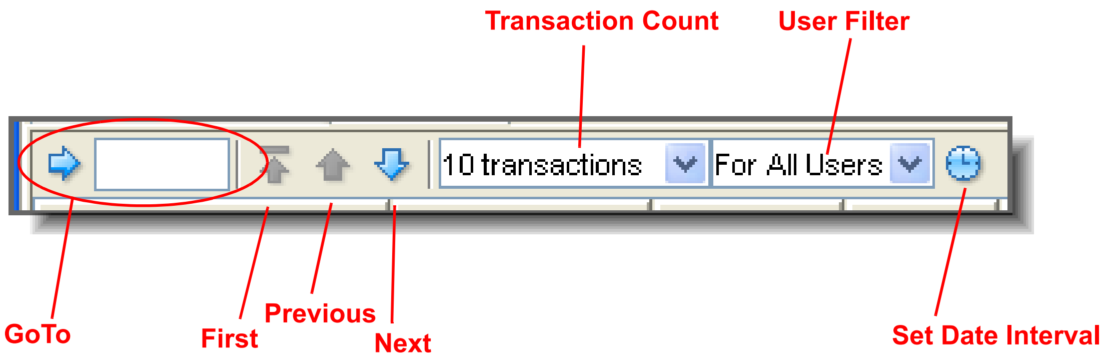
Element
\.\tools\scripts\perl\report_writer.pl
\.\tools\scripts\python\utils.py
\.\tools\tools.readme

Summary pane

Comment pane

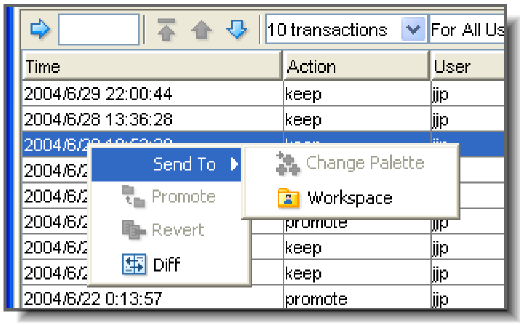
Versions pane

History Browser: Toolbar

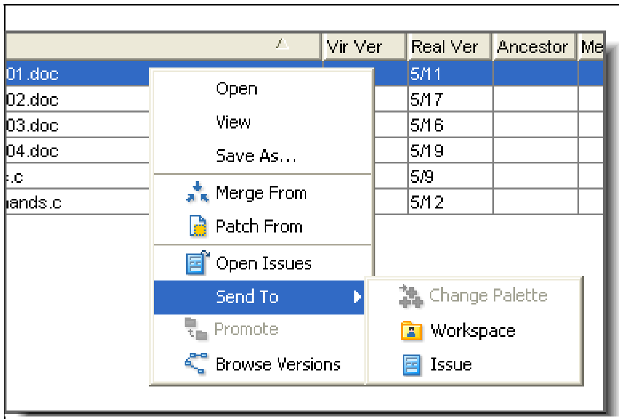


Toolbar Control	Description
Goto	Load the set of transactions beginning with the specified number.
First	Load the most recent set of transactions.
Previous	“Scroll up” to the next more recent set of transactions.
Next	“Scroll down” to the next older set of transactions.
Transaction Count	Select how many transactions are to be displayed in each “set” in the Summary pane.
User Filter	Display only the transactions performed by a particular user.
Set Date Interval	Display only the transactions that occurred in a specified interval.

History Browser: Context Menu for Items in Summary Pane

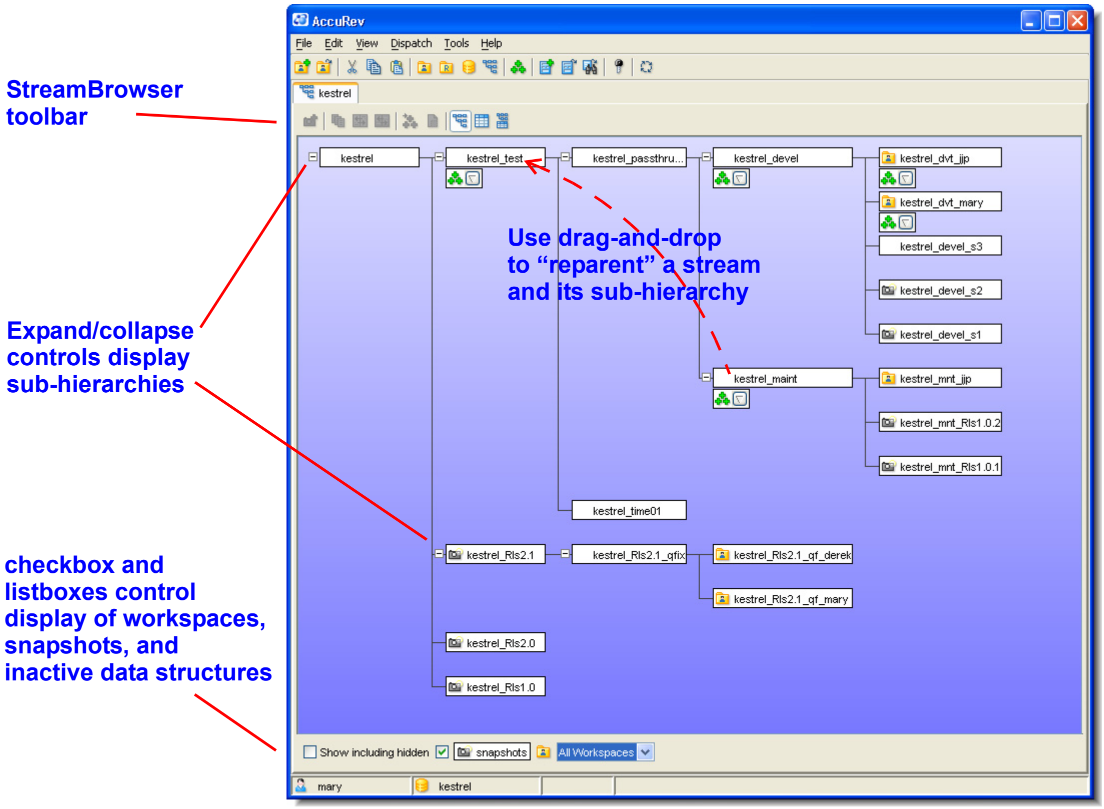
<table>
<tr><th rowspan="5"> </th><th>Command</th><th>Description</th></tr>
<tr><td>Send To</td><td><b>Change Palette</b> (dynamic stream only): Load the selected versions into the Change Palette, so that they can be promoted to another stream.
<b>Workspace</b>: Activate each of the transaction’s versions in a particular workspace.</td></tr>
<tr><td>Promote</td><td>Promote all of the transaction’s versions to the parent stream of the stream/workspace from which you invoked the History Browser.</td></tr>
<tr><td>Revert</td><td>(dynamic stream only) For the selected <b>Promote</b> transaction, performs the equivalent of an “undo” in a particular child workspace of the dynamic stream, using the CLI command <b>revert</b>.</td></tr>
<tr><td>Diff</td><td>(text-file element history only) Compare the version in the selected transaction with the version in another transaction.</td></tr>
</table>

History Browser: Context Menu for Items in Versions Pane

<table>
<tr><th colspan="2" rowspan="4"> </th><th>Command</th><th>Description</th></tr>
<tr><td>Open</td><td>Run the appropriate command on the file, according to its file type.</td></tr>
<tr><td>View</td><td>Open a text editor on a temporary copy of the currently selected file (text files only).</td></tr>
<tr><td>Save As</td><td>Copy the currently selected file to another filename.</td></tr>
<tr><td>Merge From</td><td colspan="3">Merge the selected version into the workspace from which you invoked the History Browser.</td></tr>
<tr><td>Patch From</td><td colspan="3">Patch the selected version into the workspace from which you invoked the History Browser.</td></tr>
</table>

Send To	**Workspace**: Activate the selected version in the workspace from which you invoked the History Browser. **Issue**: Record the selected version in the change package section (Changes tab) of a particular issue record. **Change Palette** (dynamic stream only): Load the selected versions into the Change Palette, so that they can be promoted to another stream.
Promote	Promote the selected version to the parent stream.
Browse Versions	Open a Version Browser on the selected version.

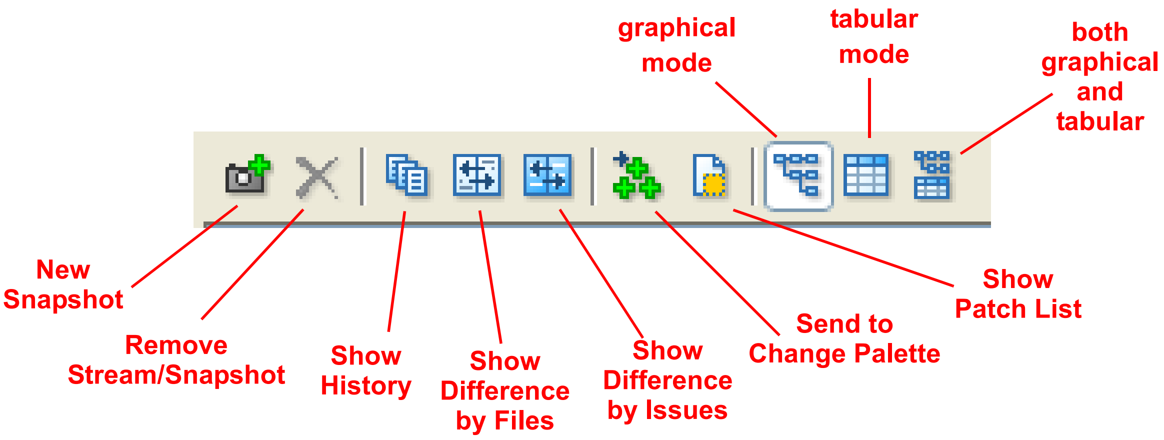
The StreamBrowser



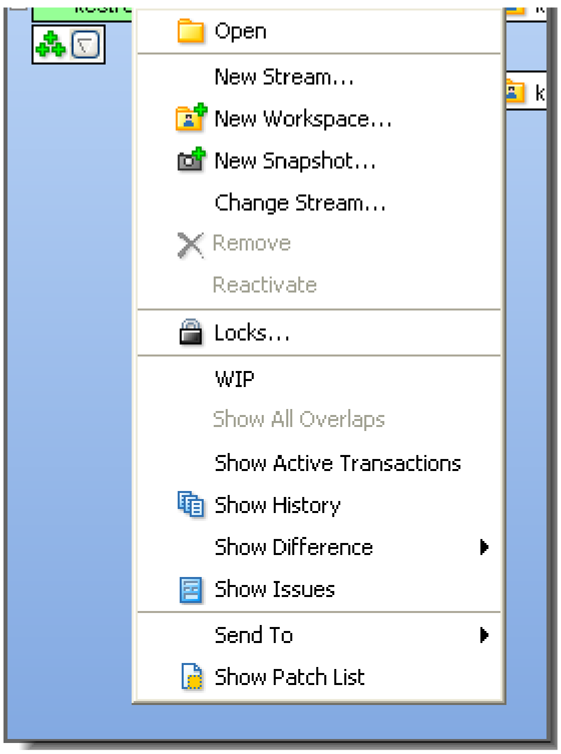
The StreamBrowser display is organized as follows:

- The depot's root stream (top-level) is at the left edge.
- A given stream's children (streams and snapshots) appear to its right; the children are arranged vertically, in this order:
    - Workspaces, in alphabetical order
    - Dynamic streams with no basis time, ordered by creation time (most recent first)
    - Dynamic Stream / Snapshot with basis times, ordered by basis time (most recent first)

StreamBrowser: Toolbar

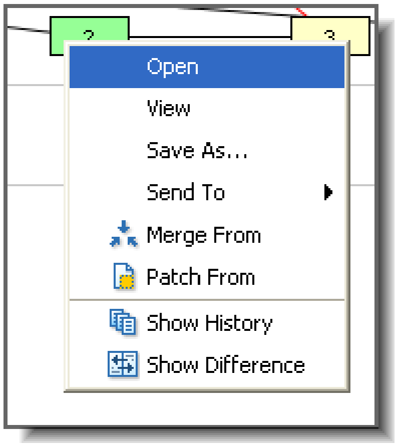


StreamBrowser: Context Menu

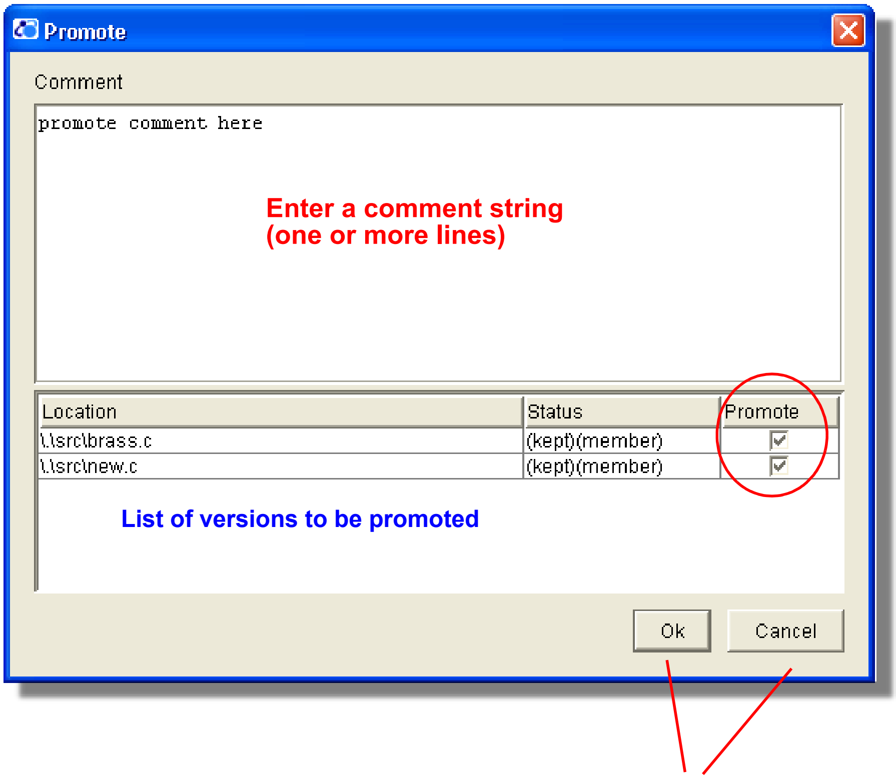
<table>
<tr><td rowspan="5">
</td><td><b>Command</b></td><td><b>Description</b></td></tr>
<tr><td>Open</td><td>Open a File Browser tab for the selected stream or snapshot
CLI: <b>files</b></td></tr>
<tr><td>New Stream</td><td>Create a new dynamic stream, passthrough stream, or snapshot as a child of the selected stream.
CLI: <b>mkstream</b></td></tr>
<tr><td>New Workspace</td><td>Create a new workspace based on the selected stream or snapshot.
CLI: <b>mkws</b></td></tr>
<tr><td>New Snapshot</td><td>Create a new snapshot based on the selected stream or snapshot.
CLI: <b>mksnap</b></td></tr>
<tr><td>Change Stream</td><td colspan="2">Change the specifications of a dynamic or passthrough stream.
CLI: <b>chstream</b></td></tr>
<tr><td>Remove</td><td colspan="2">Deactivate the selected dynamic stream, passthrough stream, snapshot, or workspace.
CLI: <b>remove</b></td></tr>
</table>

Reactivate	Reactivate a stream that was previously **Remove**d (but is still displayed, via the **Show hidden** checkbox).
Locks	Lock or unlock the selected stream. CLI: **lock**, **unlock**
WIP	Display work-in-progress for all elements. CLI: **wip**
Show All Overlaps	(workspace streams only) For files that are active in the selected stream, display all overlaps with versions in higher-level streams. CLI: **merge –B**
Show Active Transactions	Open a History Browser, displaying the transactions containing versions that need to be promoted out of the selected stream. CLI: **translist**
Show History	Open a History Browser, displaying the entire transaction history of the selected stream. CLI: **hist –s**
Show Difference By Files	Compare the selected stream with another stream that you click to specify. Differences in version-IDs and pathnames are displayed. CLI: **diff –a –i –v –V**
Show Difference By Issues	Compare the selected stream with another stream that you click to specify. Differences expressed in terms of change packages.
Show Issues	List the change packages that have been (partially or completely) incorporated into the stream, workspace, or snapshot.
Send to Change Palette	Load the Change Palette with the versions in the selected stream's default group. You must click to specify another stream as the destination for these versions. CLI: **mergelist**
Show Patch List	Lists all the individual versions that need to be patched to another stream, in order to have the other stream include all the changes in this stream. CLI: **patchlist**

The Version Browser

<table>
<tr><td rowspan="4">
</td><td><b>Command</b></td><td><b>Description</b></td></tr>
<tr><td>Open</td><td>Run the appropriate command on the file, according to its file type.</td></tr>
<tr><td>View</td><td>Display the selected version in a text editor (text files only).
CLI: <b>cat</b></td></tr>
<tr><td>Save As</td><td>Store contents of the selected version in a file
CLI: <b>cat</b></td></tr>
<tr><td>Send to</td><td colspan="2"><b>Workspace</b>: Activate this version of element in your workspace.
CLI: <b>co</b>
<b>Issue</b>: Record this version of element in the change package section of a particular issue record.</td></tr>
<tr><td>Merge From</td><td colspan="2">Perform merge from the selected version to your workspace version.
CLI: <b>merge</b></td></tr>
<tr><td>Patch From</td><td colspan="2">Perform patch from the selected version to your workspace version.
CLI: <b>patch</b></td></tr>
<tr><td>Show History</td><td colspan="2">Open a History Browser, displaying the transaction in which the selected version was created.
CLI: <b>hist –t</b></td></tr>
<tr><td>Show Difference</td><td colspan="2">Compare the selected version with another version that you click to specify.
CLI: <b>diff</b></td></tr>
<tr><td>Basic Mode button</td><td colspan="2">Eliminate some real versions from the display — show the equivalent virtual versions.</td></tr>
<tr><td>Expanded Mode button</td><td colspan="2">Include all versions, real and virtual, in the display.</td></tr>
</table>

Promote Dialog Box



```
promote comment here
```

Location	Status	Promote
\.\src\brass.c	(kept)(member)	☑
\.\src\new.c	(kept)(member)	☑

Enter a comment string (one or more lines)

List of versions to be promoted

Clear checkbox to remove a version from the set to be promoted

Complete or cancel the Promote transaction