SERENA®

PVCS® VERSION MANAGER™ 8.3

User's Guide

Serena Proprietary and Confidential Information

Publication date: January 2010

Table of Contents

Welcome to Version Manager

Thank you for choosing Serena PVCS Version Manager, a powerful and versatile version control system that will revolutionize the way you develop software. Version Manager helps you organize, manage, and protect your software development projects on every level—from storing and tracking changes to individual files, to managing and monitoring an entire development cycle.

Purpose of this manual

This manual contains conceptual and "how to" information on basic and advanced Version Manager tasks using the Version Manager desktop client.

For more information

Refer to the *Serena PVCS Version Manager Getting Started Guide* for a description of the Version Manager documentation set, a summary of the ways to work with Version Manager, and instructions for accessing the online help.

Edition status

The information in this edition applies to *Release 8.3 of Serena PVCS Version Manager* or later. This edition supersedes earlier editions of this manual.

Contacting Technical Support

Registered customers can log in to http://support.serena.com/.

Part 1

Getting Started with Version Manager

Chapter 1

Version Manager Basics

Why Use Version Manager?

Serena PVCS Version Manager is the industry standard for organizing, managing, and protecting your enterprise software assets. Version Manager enables teams of any size and varying locations to coordinate concurrent development, with secure access and a complete audit trail. Version Manager improves the quality of the final product and accelerates team development enterprise-wide by automating common tasks; increasing code reuse; and eliminating problems caused by lost changes, overwrites, and content errors.

Use Version Manager to:

- **Organize software assets.** Eliminates errors by documenting and controlling change through multiple revisions, increases code reuse, and provides a structured and efficient approach to team development.
- **Manage and enhance development workflow.** Automates common tasks, tracks changes, and enables parallel development. Promotion groups enable you to control when software is ready for the next stage of development.
- **Protect software assets.** Inventories and securely archives all project code, objects, and documentation while providing controlled user access and a complete audit trail.
- **Scale up with your needs.** Supports teams of any size from one to thousands, located across the hall or around the world.

Cross-Platform Development

Coordinate cross-platform development

Version Manager runs on multiple operating systems, platforms, and environments, giving continuity across all projects.

Version Manager enables you to use version control to manage changes made to project files on one platform, and access the files on another platform. For example, if a revision is created in a Windows environment, you could access this revision from a UNIX environment, change it, and store this new revision for later use by both platforms.

Supports Project-Oriented Version Control

Version Manager now supports projects and subprojects, which gives you the ability to mirror the directory structure in which you organize your files and folders.

You can set up your Version Manager environment to look as if you were working directly from a Windows or UNIX operating system window—except everything is under version control.

Version Manager Terminology

As you explore this latest release of Version Manager, you will notice that the desktop client contains several new features and concepts. Although the terms are new, they have much in common with functions you might already understand, such as using an operating system or working with Version Manager 6.0.

Project Terminology

Version Manager lets you organize your versioned files using project databases, projects, and subprojects to reflect the structure of your actual working directories. The table below illustrates how you might draw analogies between Version Manager project terms and the concepts that are used in operating systems or in earlier releases of Version Manager.

This Version Manager Term...	Is same as this Version Manager 5.3/6.0 Term...	And similar to this OS concept...
Project Database	Project Root	Drive or Volume
Project	Project	Directory or Folder
Subproject (multiple levels)	Folder (one level only)	Subdirectories or Subfolders (multiple levels)
Versioned File	Versioned File	File or Workfile

Project Configuration Terminology

Some of the new project configuration features can be compared to concepts introduced in earlier releases of Version Manager, as shown in the table below.

This Version Manager term...	Is similar to this Version Manager 5.3/ 6.0 term...	Description...
Project Database	Master Project	Configuration is now inherited from the project database instead of from the master project. The master project no longer exists.
Workfile location	Project Working Directory	Defines the location from which files are checked in and out of Version Manager.

Version Manager Concepts

The following concepts will help you understand how the different parts of Version Manager work with one another.

Project Database

A *project database* is a hierarchical collection of projects, subprojects, and versioned files. A project database defines a common configuration for all projects and subprojects contained within it. Users are defined for each project database.

Projects and Subprojects

Projects are logical groupings of subprojects and versioned files. *Subprojects* are projects contained within a project. Although a project database defines configuration values for each project and subproject contained within it, you can override the configuration options (if the master configuration file does not disallow the option) by using configuration files for projects and subprojects. You can also define users on a project and subproject basis.

Workfile

A *workfile* is any file that you are checking in to Version Manager to create a new revision or archive. When you check out a revision, Version Manager stores it as a workfile in your workfile location.

Versioned File

A *versioned file* is a file and its associated revisions under source control. A versioned file maintains information about the name and location of the archive where the revisions are physically stored, thus eliminating the need to specify this information each time you access a revision of the versioned file. The versioned file also stores the location of the workfile.

Archive

An *archive* is a Version Manager file that stores the evolutionary history of a versioned file. The history of a versioned file includes descriptions of changes, who made them, and when they were made. There is one archive for each versioned file. Archives also contain version label and promotion group information.

Revision

A *revision* is an instance of a versioned file that can be re-created. When you check in a workfile, the changes made to the workfile are stored as a revision in an archive. When you check out a revision, Version Manager creates a workfile in the specified workfile location.

Workspace

A *workspace* is a collection of work settings defined for a project database, which includes the work settings for all of the projects and versioned files contained within the project database. These work settings include workfile location, default version, base version, and branch version.

User

A *user* is a person who performs basic and advanced Version Manager tasks. These include, but are not limited to, creating projects, adding and checking in workfiles, checking out revisions, adding version labels, and promoting revisions.

Administrator

An *Administrator* is a person who performs administrative Version Manager tasks. These include, but are not limited to, creating and maintaining project databases, access control databases, and promotion models; setting up event triggers; and generating reports.

Version Label

A *version label* is a label used to identify a specific revision of a versioned file. Version labels can be used to efficiently check out and promote revisions that belong together, such as all of the revisions that were used to create a beta version of an application.

Default Version

Typically, the *default version* is the latest revision of a versioned file. However, the default version can be defined in the project's configuration file or workspace to be a version label.

Promotion Group

A *promotion group* is a phase of a promotion model that represents a milestone in the application development cycle. Some examples of typical promotion groups include Development, Quality Assurance, and Production.

Promotion Model

A *promotion model* is a hierarchy of promotion groups representing milestones in a software application's development process. The promotion model enforces active development work to the lowest level of the promotion model (for example, the Development promotion group). When you create a promotion model, Version Manager associates revisions with the promotion groups in the model. You can then perform operations (such as checking in and checking out) based on a promotion group.

Branch

A *branch* is a separate line of development consisting of one or more revisions that diverge from a revision on the trunk or on another branch. Branching lets you develop alternate variations of a file in parallel with the continued development of the revision from which it was branched.

Baselining

Baselining is the process of creating a snapshot of an existing project at a specific point in its development. To create the baseline, you copy the entire project into a new project based on a version label or promotion group.

The baselined project does not contain revision history; therefore, any revision that does not meet the version label or promotion group criteria is not copied into the baselined project. The files within the baselined project, which should be locked to prevent modification, can then serve as a reference source as development continues.

Event Triggers

An *event trigger* is a mechanism that causes an action to occur in response to a particular Version Manager event. When Version Manager detects the event, it executes the event trigger configured for that event. Event triggers are defined in configuration files.

Access Control Database

An *access control database* defines the users who are authorized to perform actions on projects. Access control databases are associated with project databases; however, you can also associate an access control database with individual projects within a project database to further restrict the control for each project. Using an access control database is optional.

Access List

An *access list* defines the groups of users and individual users who are authorized to perform actions on an archive. Each archive can have one access list associated with it. Before you can define an access list for an archive, you must define an access control database. An access list is a subset of the users defined in the access control database.

Configuration Options

Configuration options are settings that control how Version Manager operates. For example, configuration options control whether workfiles are deleted after check in and whether multiple locking is permitted. Configuration options are set in a configuration file.

Configuration Files

Version Manager stores configuration options in *configuration files*. The Version Manager desktop client uses two types of configuration files:

- Master configuration file, which contains configuration options for a project database and all of its projects. Version Manager automatically creates a master configuration file each time you create a project database.
- Project configuration file, which contains configuration options for a project that override the settings in the master configuration file, unless the option is not allowed in the master configuration file. Individual projects do not require that a configuration file be associated with them; they can use the settings in the master configuration file.

Version Manager Tasks

There are two types of Version Manager users:

- **Administrators** are responsible for setting up and configuring Version Manager within their organization so that users can effectively and efficiently use Version Manager to manage and control source files. For an in-depth look at Administrator tasks, see the *Serena PVCS Version Manager Administrator's Guide*.

- **Users** use Version Manager on a daily basis for managing their workfiles. Users can be classified into two categories:
  - **Project Leader** whose responsibilities can include creating projects, assigning version labels, promoting revisions, and building products.
  - **Project Team Members** whose responsibilities include editing workfiles, managing revisions, and building and testing products.

The way you use Version Manager within your organization will vary depending on the size of, and the development processes within, your organization.

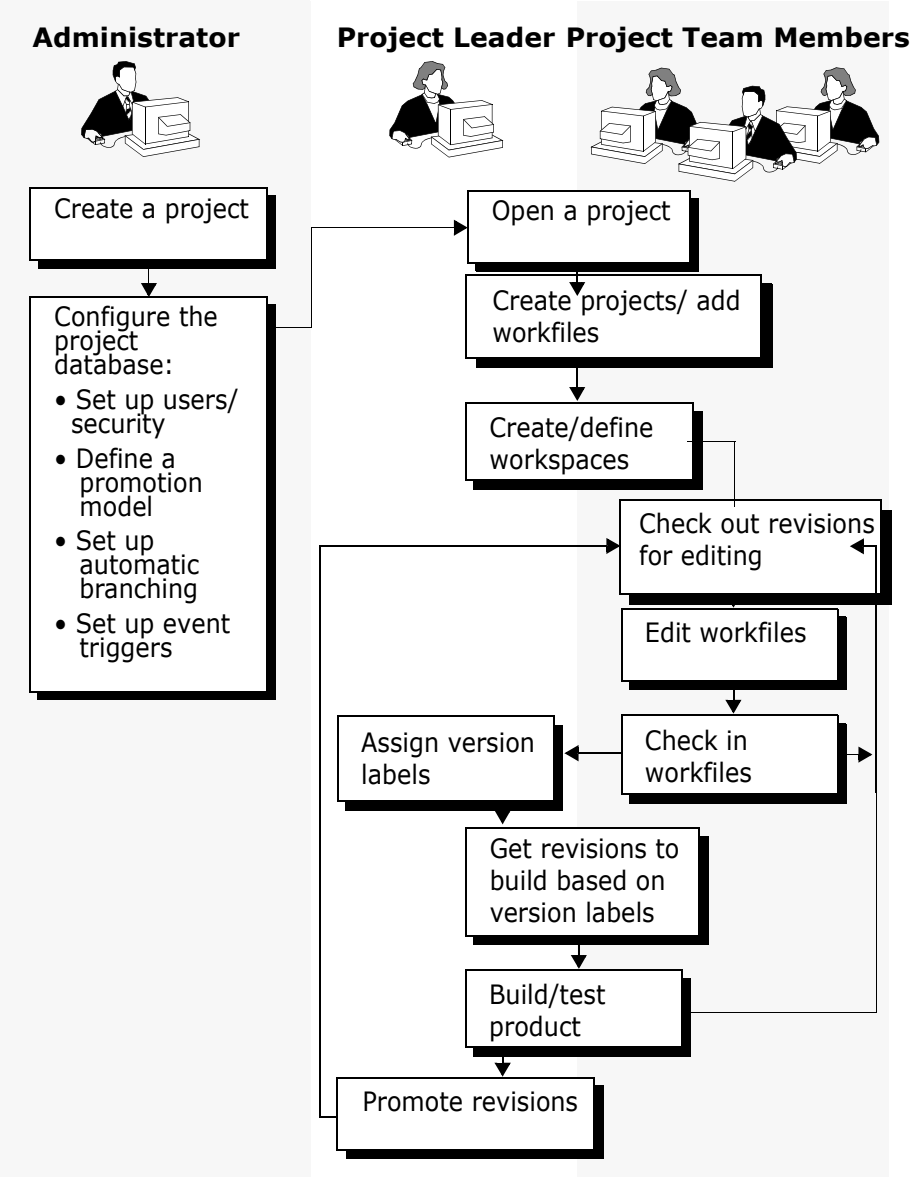
Categorizing User vs. Administrator Tasks

Version Manager tasks are organized into two main categories: User tasks and Administrator tasks.

Task	User	Administrator
Creating project databases		X
Creating access control databases		X
Configuring project databases/projects		X
Creating promotion models		X
Creating projects	X	X
Adding workfiles	X	X
Creating workspaces	X	X
Checking out revisions	X	
Checking in revisions	X	
Labeling revisions	X	X
Locking/unlocking revisions	X	
Branching and merging revisions	X	
Promoting revisions	X	X
Generating reports	X	X

Basic Workflow

The diagram below shows some of the fundamental Version Manager tasks performed by the Administrator, Project Leader, and the Project Team Members.



Workflow Tasks

- **Open a project database.** To create a project, you must open a project database. Projects, subprojects, and versioned files are contained in project databases. In most cases, the Administrator for your group sets up and configures project databases. Once this is done, you can begin creating projects.

  For more information, see "Working with Project Databases" on page 77.

- **Create projects/add workfiles.** If you already have workfiles that are organized in a directory structure, you can create projects by simply adding the entire workfile structure to Version Manager. Version Manager creates projects with the same name as the directories and adds the workfiles within the directories to the projects. If the

directories have subdirectories, subprojects are created as well, and populated with the appropriate workfiles.

For more information, see "Adding Workfiles" on page 89.

- **Create/define workspaces.** As a Project Leader, you want to make sure that a public workspace is defined for your project so that Project Team Members have access to the necessary projects and workfiles. As a Project Team Member, make sure you have a private workspace defined so that you can work locally.

  For more information, see "Using Workspaces" on page 109.

- **Check out revisions for editing.** Once workfiles are added to Version Manager, Project Team Members can begin checking out revisions for editing. When you check out a revision, a lock is placed on the versioned file to let others know that you are working on the file.

  For more information, see "Checking Out Revisions" on page 145.

- **Edit workfiles.** When you want to check out a revision and open the workfile in the appropriate editor for editing, edit the workfile. Double-clicking a versioned file in the File pane and selecting Edit (or selecting the versioned file and selecting Edit | Edit File) launches the application associated with the revision and displays a writable copy of the workfile.

  For more information, see "Working with Revisions" on page 161.

- **Check in workfiles.** When you want to preserve the state of your workfile, check it in. Once you check in a workfile, the workfile is stored as the latest revision in the archive and the lock is removed from the versioned file.

  For more information, see "Checking In Workfiles" on page 169.

- **Assign version labels.** When you want to identify a specific revision in an archive, you can assign a version label to it. Version labels are useful for checking out and promoting groups of revisions.

  For more information, see "Using Version Labels" on page 185.

- **Get revisions based on version labels.** Once version labels are assigned, the Project Leader can get copies of the latest revisions (by specifying the version label) and begin building the product. Project Team Members can get revisions for testing purposes, or to verify that a file is needed before they actually check it out.

  For more information, see "Getting Revisions" on page 155.

- **Build/test product.** Once the appropriate revisions have been copied, the Project Leader can begin building the product. Once the product is built, the Project Team Members can begin testing it.

- **Promote revisions.** As the Project Leader, if a promotion model is set up for your project, you can promote revisions to the next promotion group to begin the next stage of development once milestones are met. The project is finished once it reaches the top promotion level.

  For more information, see "Promoting Revisions" on page 229.

Advanced Tasks

During the development process, you may need to incorporate other Version Manager functions into your workflow.

- **Use Promotion Groups.** Use promotion groups when you have a well-defined process of development. Promotion groups will help you control the development of source code from the design phase to final release.

  To use promotion groups, your project must be set up with a promotion model. For more information on promotion models, see the *Serena PVCS Version Manager Administrator's Guide*.

  For more information, see "Promoting Revisions" on page 229.

- **Create Branches.** Create a branch when you want to begin parallel development from a specific revision. This is useful when you want to test changes that you don't want to immediately affect the main line of development, such as a bug fix.

  For more information, see "Branching Revisions" on page 197.

- **Compare Revisions.** Compare revisions so you can see the differences between files. Version Manager allows you to compare the files side by side in the desktop client window.

  For more information, see "Comparing Files" on page 207.

- **Merge Revisions.** You can merge files into an output file by using the Show Merge option. This option is useful when you want to merge the changes of a development branch back into the main line of development.

  For more information, see "Merging Files" on page 213.

- **Generate Reports.** You can generate reports to assist you in managing your Version Manager projects. The Journal Report provides specific information about archive activity, such as finding out when revisions were checked out of a particular archive or when version labels were assigned. The History Report provides archive information that you can use to monitor the development process, review archive histories, and check archive attributes.

  For more information, see "Using Reports" on page 239.

Using Online Help

The online help system provides comprehensive information about Version Manager procedures and concepts. It is HTML based and, on Windows systems, will open in your default HTML browser.

On Linux and UNIX systems, you must specify the location of a browser in Version Manager. See the Version Manager Installation Guide for more information.

NOTE Some help features are not supported on older browsers. See the Version Manager Installation Guide for current browser requirements.

Accessing Help

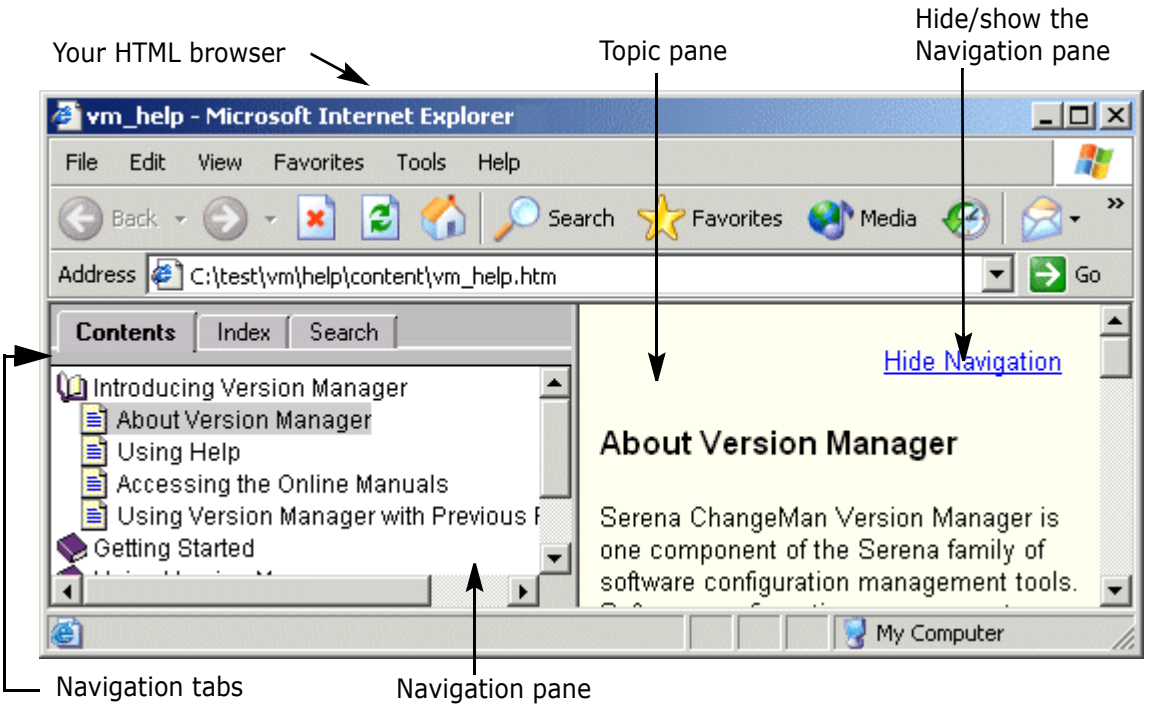
There are several ways you can access online help:

- From the menu bar, select Help | Help Topics to open the help system.

- From a dialog box, click **Help** to access a help topic specific to the dialog box.
- From a field, press F1 to access context-sensitive help specific to the field.
- From the main window, press F1 to open the help system.

The Help Window

The online help will appear similar to the following image, but will vary somewhat depending on which browser you are using.



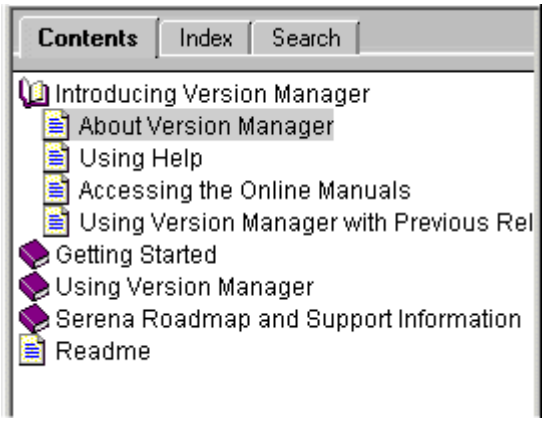
The left pane, the Navigation pane, has three tabs: Contents, Index, and Search. Topics selected in the left pane are displayed in the right pane, the Topic pane.

Show/Hide the Navigation Pane

When you launch help from a dialog box by clicking the **Help** button, the topic related to that dialog box is opened, but the Navigation pane is not displayed. To display the Navigation pane, click the Show Navigation link at the top of the topic. To hide the Navigation pane, click the Hide Navigation link.

Using the Table of Contents

The Contents tab appears in the left pane of your HTML browser.

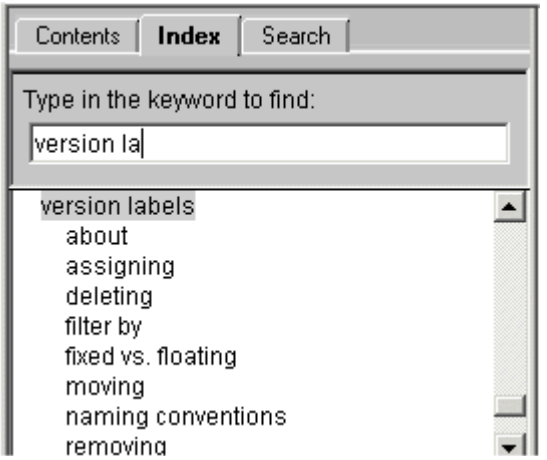


In the Table of Contents, you can:

- Select a help topic to display it in the Topic pane.
- Use the standard Windows navigation keys by highlighting a collapsed item and pressing the right arrow key to expand it or highlighting an expanded item and pressing the left arrow key to collapse it.

Using the Index

The Index tab appears in the left pane of your HTML browser.



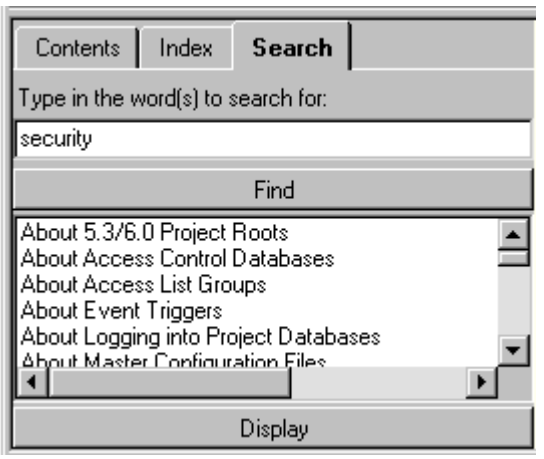
To find a word or phrase:

1. Enter one or more words in the **Type in the keyword to find** field. As you type, the keyword list scrolls to the closest match.
2. To see what topics are associated with a selected keyword, double-click it or click the **Display** button.
3. One of the following will occur:
   - If only one topic is associated with the keyword, the topic is opened in the Topic pane.

- Otherwise, a dialog box appears with a list of associated topics. Select a topic and double-click it. The topic opens in the Topic pane.

Using Search

The Search utility lets you perform a full-text search for a word or phrase throughout the entire help system. To use the Search utility, click the Search tab. The Search tab appears in the left pane of your HTML browser.



To search for a word or phrase:

1 Enter one or more words in the **Type in the word(s) to search for** field and click the **Find** button. The Search utility searches through all of the help topics and returns a list of topics that contain matches to the words you entered.

2 To view a topic, double-click it or select it and click the **Display** button. The topic is displayed in the Topic pane.

You can use the following Boolean operators to define your search.

To search for...	Enter...	Finds topics containing...
Two or more words	*word-1* AND *word-2*	Both *word-1* and *word-2* **NOTE:** Whenever you enter two or more words, the AND is assumed; the search will find all topics containing the words, not just those topics containing the string *word-1 word-2*.
At least one of a set of words	*word-1* OR *word-2*	Either *word-1* or *word-2* or both
One word while excluding another	*word-to-find* NOT *word-to-exclude*	The *word-to-find* but not the *word-to-exclude*

Printing Help Topics

To print a help topic, use the print feature of your HTML browser.

NOTE To print a topic, you may need to click in the Topic pane before selecting Print. Otherwise, if you last clicked in the Navigation pane, the Table of Contents may print instead of the topic you selected.

Using the User Scenarios

The *Serena PVCS Version Manager User's Guide* includes user scenarios that describe how a project team might use Version Manager to implement source control. These scenarios are distributed throughout this guide and are chronologically organized and interrelated. Together, they describe a real-world workflow for implementing and working with Version Manager.

The user scenarios are formatted so you can easily find them. Each scenario is located at the end of a chapter and clearly defined by a scenario graphic, as shown on the left. This design allows you to skim the book and easily locate each of the scenarios.

You can use these scenarios in the following ways:

- Use the scenarios as an introduction to the full set of features by first reading all of the scenario in succession.
- Use an individual scenario as an elaboration of a feature that you are interested in learning about.
- Use one or more scenarios as a tool for helping you implement a particular feature set. You may use the descriptions in the selected scenarios as a guide for your own implementation.

Scenario Background Information

For Fun is a large software company that designs computer games. Jacob is the senior Project Leader who oversees the development of all of the games. Currently, the developers are close to completing their work on two games: Chess and Bridge. The source code for Chess and Bridge is under version control in Version Manager 5.3. In addition, the team has started work on computerized versions of two additional games: Checkers and Solitaire. Because the company is about to migrate to the latest release of Version Manager, the Checkers and Solitaire code has not yet been incorporated into the version control environment.

In preparation for the migration to the latest Version Manager, Ruby, the System Administrator, has configured a project database that will organize hierarchically all of the game projects. She has also defined the appropriate security for the two teams, such as user names, passwords, and project privileges. She has also set up branching and merging features in Version Manager that simplify and automate parallel development for the team. In addition, starting with the newer release, the company will use promotion modeling in Version Manager to introduce their system development lifecycle to the development team. See the *Serena PVCS Version Manager Administrator's Guide* for

details on setting up automatic branching and merging, and defining a promotion model that supports your system development lifecycle.

Jacob's main goal for the upgrade is to keep the disruption of the development process to a minimum. The basic daily tasks, such as accessing files, editing and viewing them, and then checking file changes back in to the version control environment are obvious and easy to perform. On the other hand, he does expect a learning curve for the less frequently performed, more advanced tasks, such as working with version labels, branching, merging, and promoting files to the next system development milestone. He assumes, however, that once the advanced developers understand the processes, Version Manager will make it easy for them to perform these tasks.

Part 2

Using the Version Manager Desktop Client

Chapter 2

Common Project Database, Project, and Versioned File Tasks

Selecting Items

You can select every item in the Version Manager desktop client. You can perform actions such as check in, get, check out, lock, unlock, and assign version label on any of these items.

If you select a...	You can perform actions on...
Project database	The entire contents of the project database (including files in projects and subprojects)
Project	The entire contents of an individual project or multiple selected projects (including files in subprojects)
Folder	The Version Manager 5.3/6.0 folder and its contents or multiple selected folders and their contents
Versioned file	Individual or multiple selected versioned files
Revision	Individual or multiple selected revisions
Version label	Individual or multiple selected version labels
Promotion group	Individual or multiple selected promotion groups

Selecting Multiple Projects and Folders

Version Manager supports the multiple selection of projects in a project database and multiple selection of folders in a 5.3/6.0 project root. Multiple projects or folders can be selected in the project pane if they all have the same parent project or parent project database. Multiple projects may not be selected in a 5.3/6.0 project root.

To select more than one project/folder, click the items while holding down the SHIFT key to select a series of items or click the items while holding down the CTRL key to select non-adjacent items.

When multiple projects/folders are selected, all of the files contained within each selected item are shown in the file pane. A header line identifying the project/folder that the files are contained in is displayed for each item selected, followed by the files contained within the item. If you have the recursive file filter turned on (View | Filter | Recursive), then all of the subprojects and their files are also displayed.

Within the file pane, you can perform an action on a subset of files that you have selected. The selected files do not need to have the same parent project/folder.

Actions Allowed

The following actions are allowed when multiple projects/folders are selected.

- Check In
- Get
- Check Out
- Lock/Unlock
- Assign, Delete, Rename Version Labels
- Promote, Assign, Change, and Remove Promotion Groups

- Move
- Copy
- Delete
- Archive Attributes
- Set Workspace
- Set Workfile Location
- Show History Report
- Show Journal Report

NOTE You cannot drag and drop multiple projects/folders to perform an action. You must either use the menu, context menu, tool bar button, or the shortcut method to perform the action. For example, to check in revisions from multiple projects, you must either select Actions | Check In, select Check In from the context menu, click the Check In button on the tool bar, or press CTRL+I (the shortcut method).

Access Control Database Permissions

When multiple projects/folders are selected, Version Manager does not disable menu items regardless of the access control database privileges. Version Manager attempts to perform the chosen action on all of the selected items, and if the action is not allowed on a selected item, Version Manager generates an error and then continues processing the remaining selected items.

Promotion Groups

If you are performing an action that can use a promotion group (like check out or check in), Version Manager displays the promotion groups of the promotion model defined for the parent project database/project in the dialog box. If, when defining the action you want to perform, you select a promotion group that is incompatible with a project's promotion model, Version Manager generates an error and then continues processing the remaining selected items.

Workfile Locations

If you are performing an action that uses a workfile location (like check out, check in), Version Manager displays the workfile location of the parent project database/project in the dialog box. However, absolute workfile locations of selected projects/folders are respected and are not changed to the workfile location of the parent project database/project.

In the case of non-absolute workfile locations, Version Manager determines the workfile location by taking the workfile location of the parent project database/project and adding relative paths for each selected project/folder. The project/folder name will be appended to the parent's absolute workfile location to determine the project/folder's workfile location.

For example, if the parent's workfile location is `c:\producta\work` and the names of the selected projects are proj1, proj2, and proj3, the workfile locations for the selected projects are `c:\producta\work\proj1`, `c:\producta\work\proj2`, and `c:\producta\work\proj3`, respectively.

Expanding and Collapsing Items

If a project database, project, or subproject contains nested projects, a plus sign (⊞) appears next to it. To expand your view of the item, click the plus sign.

To collapse a project database, project, or subproject, click the minus sign (⊟) next to the item.

Renaming Items

Renaming projects/ subprojects

You can rename a project database, project, or subproject. When you rename a project or subproject, a new workfile directory is created to match the new project name. Once you change the name of a project or subproject, you will be checking out, copying, and checking in files from the new workfile location. The location is displayed in the Workfile location field of the Rename dialog box.

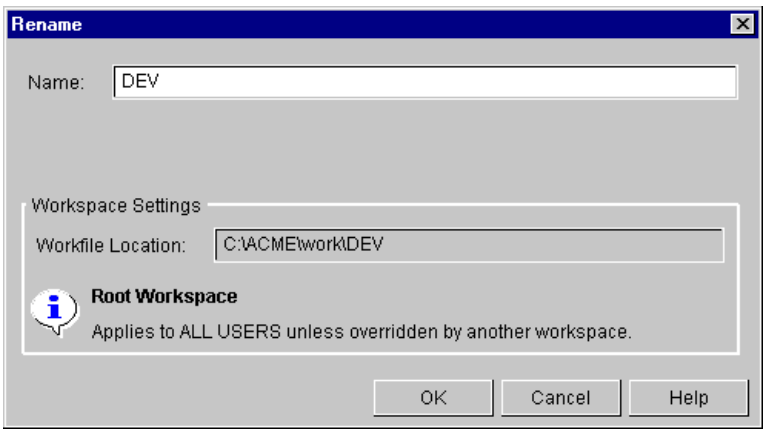
If someone has versioned files checked out while you are changing the name of the project or subproject, Version Manager retains the old workfile location. When the file is checked back in, the **Check In From** field displays the path where the file was originally checked out to. You have the option of checking it back in to the original workfile location or using the new workfile location.

Renaming project databases

When you rename a project database, only the name is changed. You do not need to specify new archive or workfile locations.

To rename a project database, project, or subproject:

1. Select the item you want to rename.
2. Select File | Rename. The Rename dialog box appears.



NOTE The Rename dialog box varies depending on the item you selected.

3. Enter the new name of the item and click **OK**. The item appears with its new name in the appropriate Version Manager pane.

Copying Items

You can copy versioned files, projects, 5.3/6.0 project roots and folders, and project databases. You can use the Menu bar or drag and drop to copy projects, folders, and versioned items. You must use the Menu bar to copy project databases.

The following table lists the items that you can copy:

You can copy a...		**To a...**				
		5.3/6.0			**New Format**	
		Project Root	**Project**	**Folder**	**Project Database**	**Project**
5.3/ 6.0	Project Root				New DB	
	Project				X	X
	Folder		X		X	X
	Versioned File			X	X	X
New Format	Project Database				New DB	
	Project				X	X
	Versioned File				X	X

Copying Versioned Files

Copy versioned files whenever you need the same versioned file in more than one project or project database. When you copy versioned files, the files are copied to the new location but continue to reference the existing archives in their current location.

You can copy versioned files into any open project or project database. You can also copy 5.3/6.0 versioned files into any 5.3/6.0 folder. You cannot copy versioned files in the new format into 5.3/6.0 project root, project, or folder.

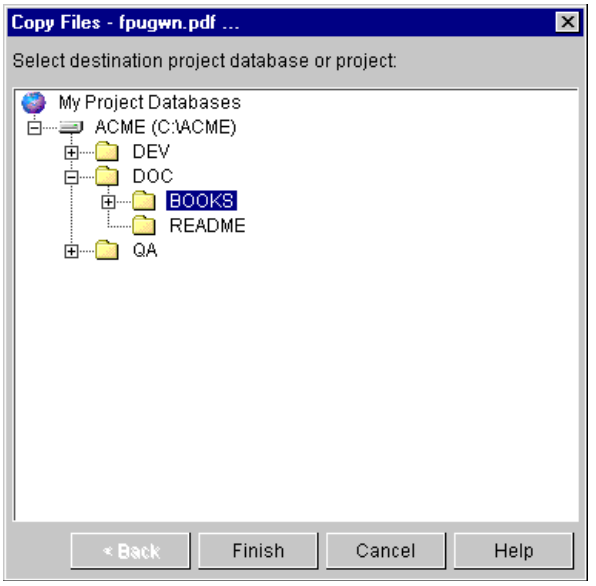
Copying Versioned Files Using the Menu Bar

Menu bar method

To copy versioned files using the Menu bar:

1 Select the versioned files you want to copy.

2 Select Edit | Copy. The Copy Files dialog box appears.



3 Select the destination project, project database, or 5.3 folder and click **Finish**.

Copying Versioned Files Using Drag and Drop

Drag and drop method

You can also copy files by dragging and dropping versioned files from the File pane to the Project pane.

To copy versioned files using drag and drop:

If you want to copy versioned files to a project in...	Then do this...
The same project database	**1** Select the versioned files you want to copy in the File pane. **2** Hold the CTRL key and drag the versioned files into the appropriate project or subproject.
A different project database	**1** Select the versioned files you want to copy in the File pane. **2** Drag the versioned files into the appropriate project or subproject.

Copying Projects

Copying a Version Manager project copies the project to a different project or project database. When you copy a project, you can:

- Copy all of its subprojects or only copy the project.
- Copy the archives to the new archive location or use the existing archive location.
- Copy existing configuration files to the new project location, use the existing configuration files, or create new configuration files.

- Copy the existing access control database to the new project location, use the existing access control database, or create a new access control database.

NOTE When you copy a project, the workspace settings for the project are not retained.

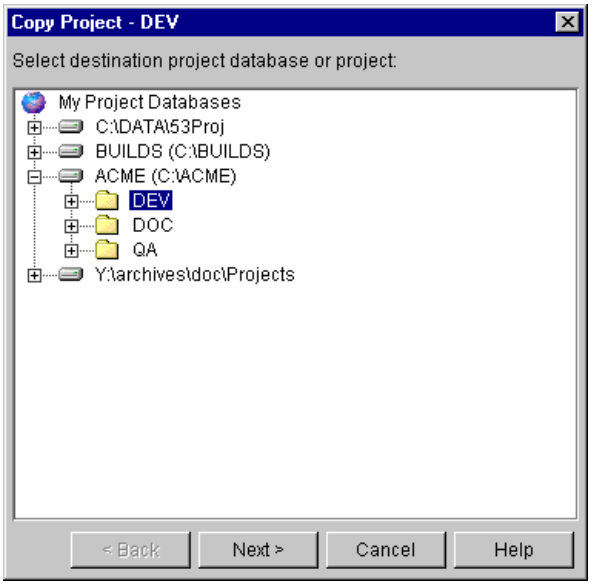
Copying Projects Using the Menu Bar

Menu bar method

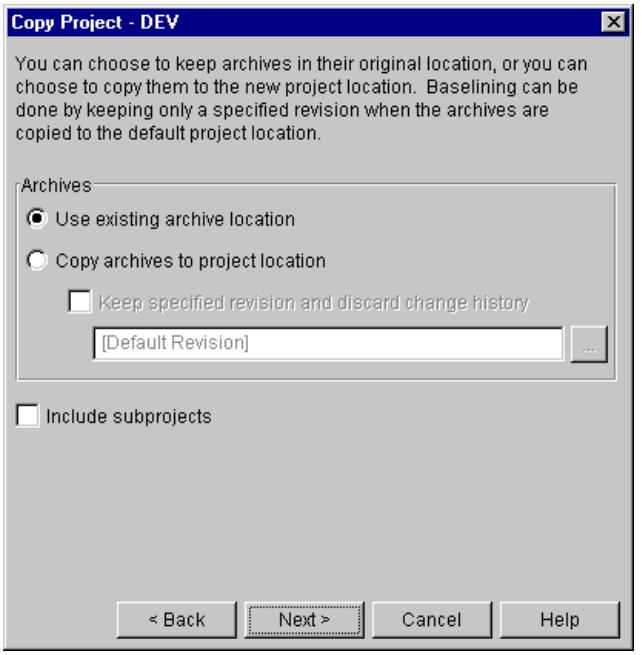
To copy projects using the Menu bar:

1 Select the project you want to copy.

2 Select Edit | Copy. The Copy Project dialog box appears.



3 Select a destination project or project database and click **Next**. A second Copy Project dialog box appears.



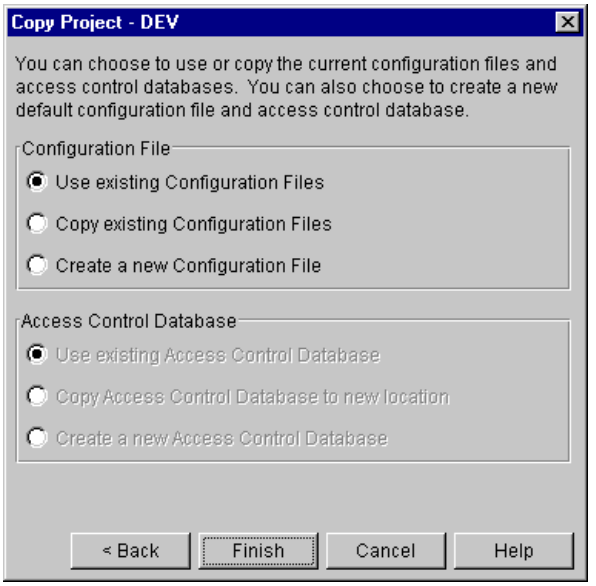
4 In the Archives group, select:

- **Use existing archive location** if you want to copy the project to a new location but continue to reference the existing archives in their current location. This is the default option.
- **Copy archives to project location** if you want to copy the project's archive directory (and its contents) and place it beneath the archives directory of the target project database. The location of the new archive directory mirrors the hierarchy of the target project database within the archives directory.
- **Keep specified revision and discard change history** to specify a base version by selecting a version label or promotion group for baselining.

  To use a revision other than the default revision as the base version, enter the version label or promotion group in the **Default Revision** field, or click the Browse button. Revisions prior to the base version will not be copied.

5 If the project you are copying contains subprojects that you want to copy, select the **Include subprojects** check box.

6 Click **Next**. The third Copy Project dialog box appears.



7 If your project does not use a configuration file, then the options in the Configuration File group are grayed. If your project uses a configuration file, select:

- **Use existing Configuration Files** if you want to copy the project to a new location but continue referencing the existing configuration file. The copied project shares the configuration file with the original project. This is the default option.
- **Copy existing Configuration Files** if you want to make a copy of the existing configuration file and place it in the archives directory beneath the target project database.
- **Create a new Configuration File** if you want to create a default configuration file and place it in the archives directory beneath the target project database. The default settings of your configuration file are controlled by your Administrator.

8 If your project does not use an access control database, then the options in the Access Control Database group are grayed. If your project uses an access control database, select:

- **Use existing Access Control Database** if you want to copy the project to a new location but continue referencing the existing access control database. This is the default option.
- **Copy Access Control Database to new location** if you want to make a copy of the existing access control database and place it in the archives directory beneath the target project database.
- **Create a new Access Control Database** to create a new *empty* access control database and place it in the archives directory beneath the target project database. This database contains the Administrator user and default privileges.

9 Click **Finish**.

Copying Projects Using Drag and Drop

Drag and drop method

You can also copy projects by dragging and dropping them within the Project pane.

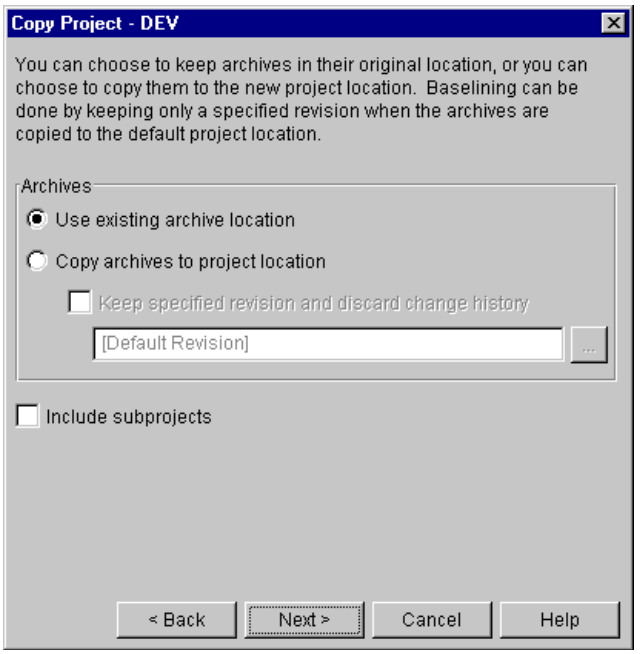
To copy projects using drag and drop:

1 Select the project you want to copy.

If you want to copy a project...	Then do this...
Within the same project database	**a** Select the project you want to copy in the Project pane. **b** Hold the CTRL key and drag the project into the destination project.
To a different project database	**a** Select the project you want to copy in the Project pane. **b** Drag the project into the destination project on a different project database.

The Confirm Item Copy message appears.

2 Click **Yes**. The Copy Project dialog box appears.



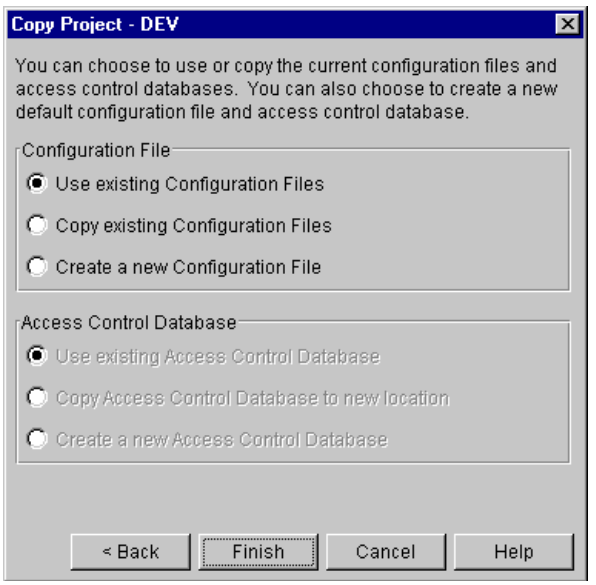
3 In the Archives group, select:

- **Use existing archive location** if you want to copy the project to a new location but continue to reference the existing archives in their current location. This is the default option.
- **Copy archives to project location** if you want to copy the project's archive directory (and its contents) and place it beneath the archives directory of the target project database. The location of the new archive directory mirrors the hierarchy of the target project database within the archives directory.
- **Keep specified revision and discard change history** to specify a base version by selecting a version label or promotion group for baselining.

To use a revision other than the default revision as the base version, enter the version label or promotion group in the **Default Revision** field, or click the Browse button. Revisions prior to the base version will not be copied.

4 If the project you are copying contains subprojects that you want to copy, select the **Include subprojects** check box.

5 Click **Next**. The third Copy Project dialog box appears.



6 If your project does not use a configuration file, then the options in the Configuration File group are grayed. If your project uses a configuration file, select:

- **Use existing Configuration Files** if you want to copy the project to a new location but continue referencing the existing configuration file. The copied project shares the configuration file with the original project.
- **Copy existing Configuration Files** if you want to make a copy of the existing configuration file and place it in the archives directory beneath the target project database. This is the default option.
- **Create a new Configuration File** if you want to create a default configuration file and place it in the archives directory beneath the target project database. The default settings of your configuration file are controlled by your Administrator.

7 If your project does not use an access control database, then the options in the Access Control Database group are grayed. If your project uses an access control database, select:

- **Use existing Access Control Database** if you want to copy the project to a new location but continue referencing the existing access control database. This is the default option.
- **Copy Access Control Database to new location** if you want to make a copy of the existing access control database and place it in the archives directory beneath the target project database.
- **Create a new Access Control Database** to create a new *empty* access control database and place it in the archives directory beneath the target project database. This database contains the Administrator user and default privileges.

8 Click **Finish**.

Copying Project Databases

You can copy an entire project database; however, you must select a new project database name. You must use the Menu bar to copy a project database; drag and drop is not supported.

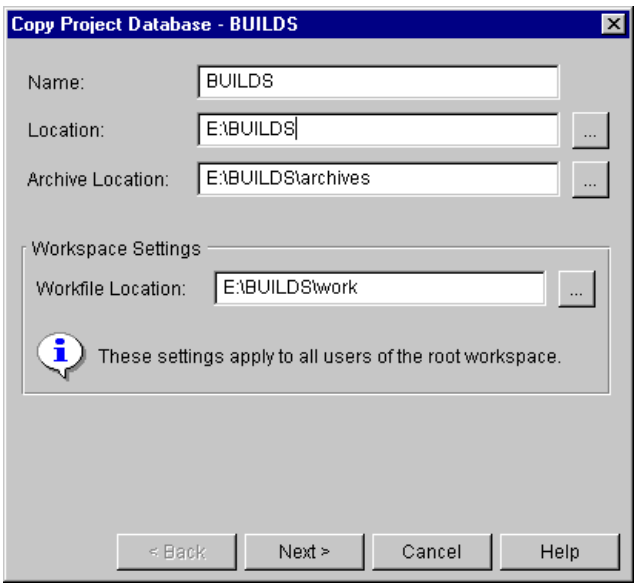
When you copy a project database, you can:

- Copy the entire contents of the project database or just the versioned files at the project database level
- Copy the archives to the new archive location or use the existing archive location
- Copy the existing configuration files to the new project database, use the existing configuration files, or create new configuration files
- Copy the existing access control database to the new project database, use the existing access control database, or create a new access control database

To copy a project database:

1 Select the project database you want to copy.

2 Select Edit | Copy. The Copy Project Database dialog box appears.



3 Enter a new project database name in the **Name** field. Project database names cannot begin or end with a tab or blank space. No other character restrictions apply to project database names.

4 Enter a location for the new project database in the **Location** field or click the Browse button to select a location. Notice that the location you enter is also reflected in the **Archive Location** field.

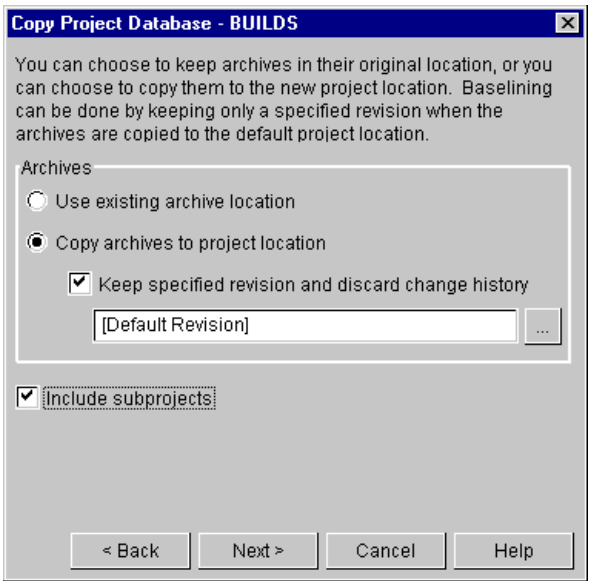
5 A new archive directory relative to the project database location you entered is specified in the **Archive Location** field. If you want to specify a different archive location, enter it in this field or click the Browse button to select a location.

6 You must specify a workfile location in the **Workfile Location** field. We recommend that you specify a local drive that is available to most users. Remember that Project

Team Members can always change the workfile location by creating a private workspace that specifies a different workfile location.

Workfile locations can contain `$HOME` at the root of their paths (for example, on UNIX, `$HOME/work/projecta` could expand to `/usrs/cherylc/work/projecta`). Version Manager substitutes the value of the HOME environment variable to compute the workfile location. If a user's HOME environment variable is not defined, Version Manager substitutes a blank space when computing the workfile location.

7 Click **Next**. The second Copy Project Database dialog box appears.



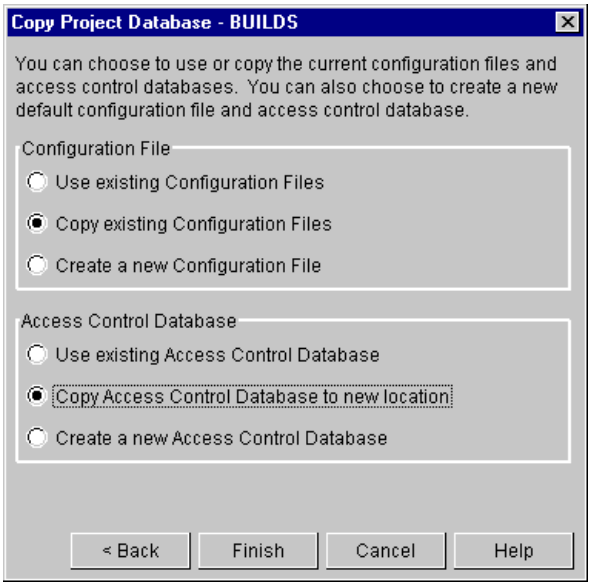
8 In the Archives group, select:

- **Use existing archive location** if you want to copy the project database to a new location but continue to reference the existing archives in their current location. This is the default option.
- **Copy archives to project location** if you want to copy the archives and place them in the archives directory of the project database you are copying.
- **Keep specified revision and discard change history** to allow selection of a base version for baselining.

  To use a revision other than the default revision as the base version, select the version label or promotion group in the **Default Revision** field. Revisions prior to the base version will not be copied.

9 If the project database you are copying contains projects and subprojects that you want to copy, select the **Include subprojects** check box. By default, Version Manager only copies the versioned files at the root level of the project database.

10 Click **Next**. The third Copy Project Database dialog box appears.



11 If your project database does not use a configuration file, then the options in the Configuration File group are grayed. If your project database uses a configuration file, select:

- **Use existing Configuration File** if you want to copy the project database to a new location but continue referencing the existing configuration file. This is the default option.
- **Copy existing Configuration Files** if you want to make a copy of the existing configuration file and place it in the archives directory of the project database you are copying.
- **Create a new Configuration File** if you want to create a default configuration file and place it in the archives directory of the project database you are copying. The default settings of your configuration file are controlled by your Administrator.

12 If your project database does not use an access control database, then the options in the Access Control Database group are grayed. If your project database uses an access control database, select:

- **Use existing Access Control Database** if you want to copy the project to a new location but continue referencing the existing access control database. This is the default option.
- **Copy Access Control Database to new location** if you want to make a copy of the existing access control database and place it in the archives directory of the project database you are copying.
- **Create a new Access Control Database** to create a new *empty* access control database and place it in the archives directory of the project database you are copying. This database contains the Administrator user and default privileges.

13 Click **Finish**.

Copying 5.3/6.0 Folders

You can copy a 5.3/6.0 folder into any open project or project database, or into a different 5.3/6.0 project. You cannot copy a 5.3/6.0 folder within the same 5.3/6.0 project.

When you copy a 5.3/6.0 folder into a project or project database, Version Manager upgrades the 5.3/6.0 folder into a project. Before you upgrade your 5.3/6.0 folder, make sure to read the section "Working with Version Manager 5.3/6.0 Project Roots" in the *Serena PVCS Version Manager Getting Started Guide*.

When you copy a 5.3/6.0 folder into a different 5.3/6.0 project, you do not have the option of copying the archives to the new project location.

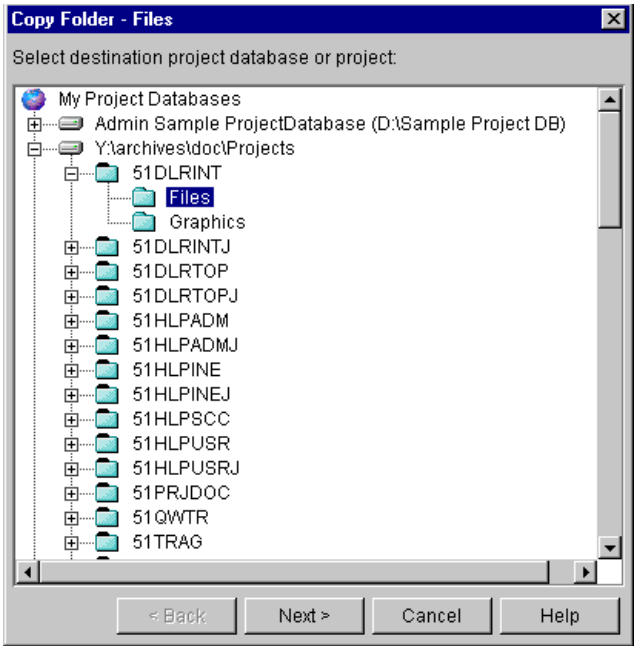
Copying 5.3/6.0 Folders Using the Menu Bar

Menu bar method

To copy 5.3/6.0 folders using the Menu bar:

1 Select the 5.3/6.0 folder or folders that you want to copy.

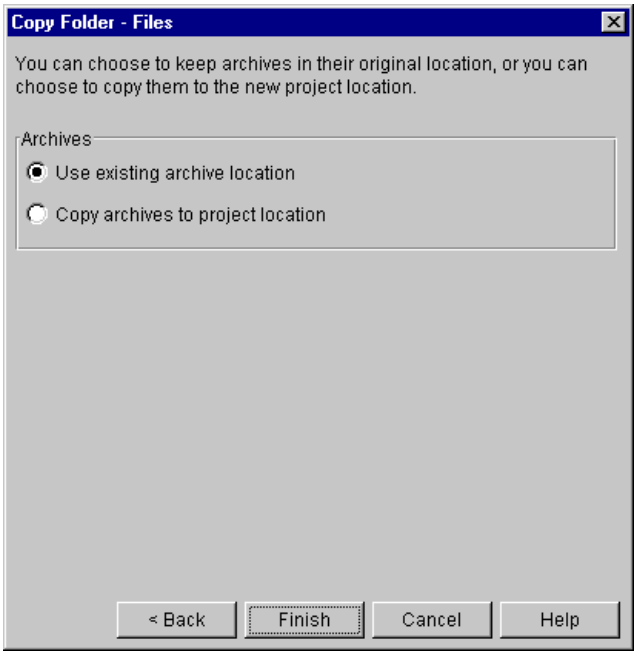
2 Select Edit | Copy. The Copy Folder dialog box appears.



3 Select a copy destination. If you select a:

- Different 5.3/6.0 project, the folder is copied to its destination and the copy action is completed when you click **Next**. The archives remain in their current location and the two projects share the existing archives.

- Project or project database as your copy destination, the second Copy Folder dialog box appears when you click **Next**.



4 In the Archives group, select:

- **Use existing archive location** if you want to copy the folder to a new location but continue to reference the existing archives in their current location. This is the default option.
- **Copy archives to project location** if you want to copy the folder's archive directory and place it beneath the archives directory of the target project database. The location of the new archive directory mirrors the hierarchy of the target project database within the archives directory.

5 Click **Finish**.

Copying 5.3/6.0 Folders Using Drag and Drop

Drag and drop method

You can also copy 5.3/6.0 folders by dragging and dropping them within the Project pane.

To copy 5.3/6.0 folders using drag and drop:

1 Select the folder you want to copy.

If you want to copy a 5.3/6.0 folder...	Then do this...
To a project or project database	**a** Select the folder you want to copy in the Project pane. **b** Drag the folder into the destination project or project database.
To a different 5.3/6.0 project	**a** Select the folder you want to copy in the Project pane. **b** Hold the CTRL key and drag the folder into a different 5.3/6.0 project.

The Confirm Item Copy message appears.

2 Click **Yes**. If you select a:

- Different 5.3/6.0 project as your copy destination, the folder is copied to its destination and the copy action is completed. The archives remain in their current location and the two projects share the existing archives.
- Project or project database as your copy destination, the Copy Folder dialog box appears.

3 In the Archives group, select:

- **Use existing archive location** if you want to copy the folder to a new location but continue to reference the existing archives in their current location. This is the default option.
- **Copy archives to project location** if you want to copy the folder's archive directory and place it beneath the archives directory of the target project

database. The location of the new archive directory mirrors the hierarchy of the target project database within the archives directory.

4 Click **Finish**.

Copying 5.3/6.0 Projects

You can copy 5.3/6.0 projects into any open project or project database that has a newer Version Manager format. When you copy a 5.3/6.0 project into a new project, Version Manager upgrades the 5.3/6.0 project to the newer format.

You can copy (upgrade) 5.3/6.0 project roots that were created using the Version Manager 5.3/6.0 desktop client or one of the Version Manager Development Interfaces. For complete information about how to upgrade 5.3/6.0 projects that were created in one of the Version Manager Development Interfaces, refer to the implementation guide for your development environment.

Before you upgrade your 5.3/6.0 project, make sure to read the section "Working with Version Manager 5.3/6.0 Project Roots" in the *Serena PVCS Version Manager Getting Started Guide*. If you want your 5.3/6.0 projects to have complete access to the Version Manager functions, you must copy them into the new Version Manager project format.

Before you copy a 5.3/6.0 project to a project database, make sure that you have an open project database where you can copy the project.

When you copy a 5.3/6.0 project, you can:

- Organize the new project based on the structure of the existing 5.3/6.0 project or base it on the 5.3/6.0 workfile hierarchy.
- Copy the archives to the new archive location or use the existing archive location.

NOTE When copying a project that contains multiple folders, remember that the subfolders may have different archive locations. If you copy a project and use the existing archive locations, the new project still references all of the different archive locations.

Any new archives added to the project after it is copied are placed in the archive directory of the project database.

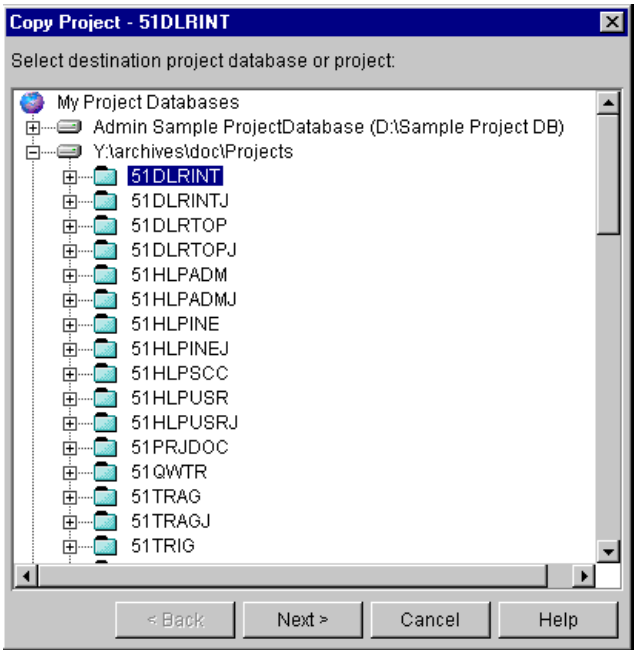
- Copy the existing configuration files to the new project location, use the existing configuration files, or create new configuration files.
- Copy the 5.3/6.0 project's existing access control database to the new project location, use the existing access control database, or create a new access control database.

Copying 5.3/6.0 Projects Using the Menu Bar

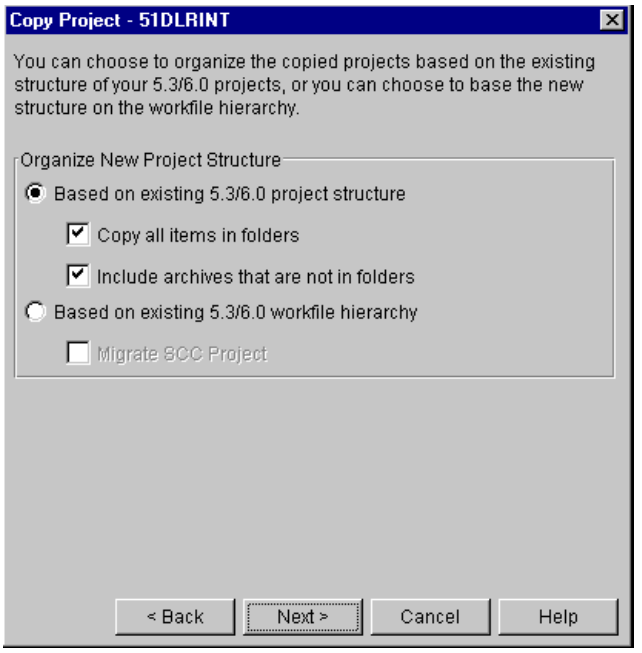
To copy a 5.3/6.0 project using the Menu bar:

1 Select the 5.3/6.0 project you want to copy.

2 Select Edit | Copy. The Copy Project dialog box appears.



3 Select a destination project or project database and click **Next**. A second Copy Project dialog box appears.



4 In the Organize New Project Structure group, select:

- **Based on existing 5.3/6.0 Project structure** if you want to organize the new project based on the structure of the existing 5.3/6.0 project. This structure will contain no more than two levels, a project and a subproject. This is the default option.

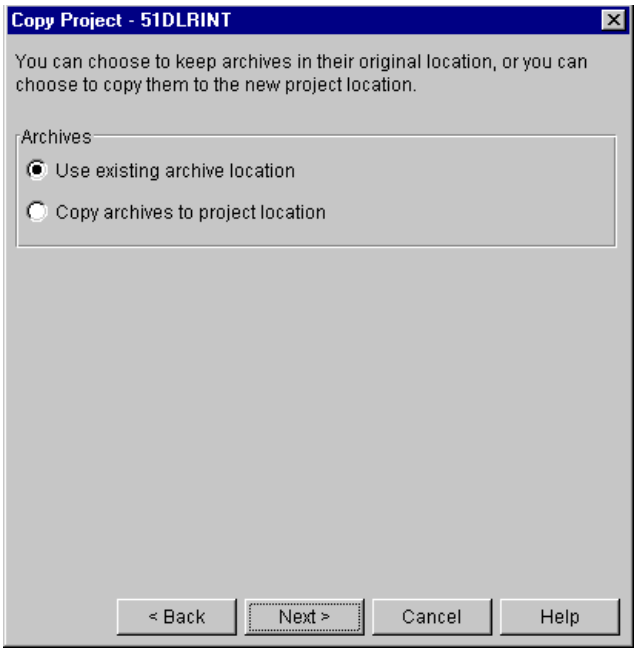
  If you want to copy all archives that are stored in folders, check the **Copy all items in folders** check box.

If you want to copy archives that are not in folders (archives located at the root of the 5.3/6.0 project), check the **Include archives that are not in folders** check box. This is the default option.

- **Based on existing 5.3/6.0 workfile hierarchy** if you want to create a project structure to match the workfile structure. This structure will contain as many subprojects as needed to reflect the exact location of the workfiles.
- **Migrate SCC project** if you want to migrate an SCC 5.3/6.0 project. You must also select the **Based on existing 5.3/6.0 workfile hierarchy** option.

NOTE If you are migrating an SCC project and other Development Interface projects share archives with the project you are converting, we do not recommend moving the archives to the new project directory. Moving the archives deletes the association of the archives with the other projects.

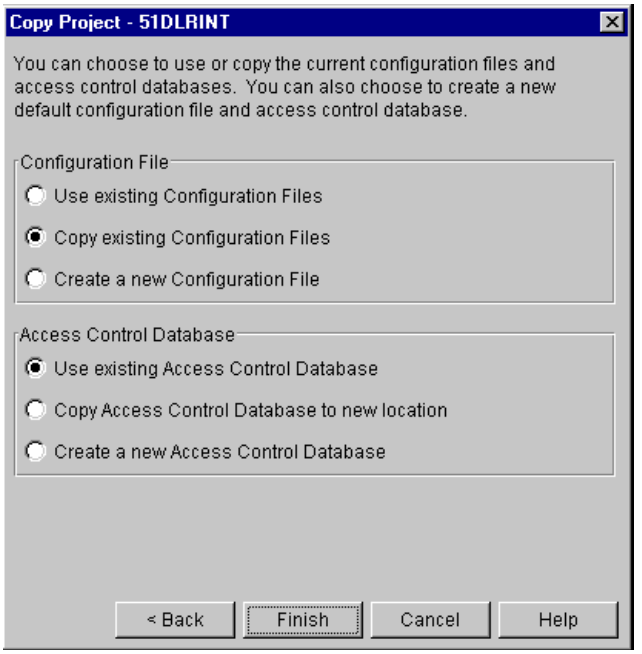
5 Click **Next**. A third Copy Project dialog box appears.



6 In the Archives group, select:

- **Use existing archive location** if you want to copy the project to a new location but continue to reference the existing archives in their current location. This is the default option.
- **Copy archives to project location** if you want to copy the project's archive directory (and its contents) and place it beneath the archives directory of the target project database. The location of the new archive directory mirrors the hierarchy of the target project database within the archives directory.

7 Click **Next**. A fourth Copy Project dialog box appears.



8 If your project does not use a configuration file, then the options in the Configuration File group are grayed. If your project uses a configuration file, select:

- **Use existing Configuration Files** if you want to copy the project to a new location but continue referencing the existing configuration file. The copied project shares the configuration file with the original project.

IMPORTANT! We do not recommend sharing configuration files if the 5.3/6.0 project you are copying uses an access control database.

- **Copy existing Configuration Files** if you want to make a copy of the existing configuration file and place it in the archives directory beneath the target project database. This is the default option.
- **Create a new Configuration File** if you want to create a default configuration file and place it in the archives directory beneath the target project database. The default settings of your configuration file are controlled by your Administrator.

9 If your project does not use an access control database, then the options in the Access Control Database group are grayed. If your project uses an access control database, select:

- **Use existing Access Control Database** if you want to copy the project to a new location but continue referencing the existing access control database. This is the default option.
- **Copy Access Control Database to new location** if you want to make a copy of the existing access control database and place it in the archives directory beneath the target project database.
- **Create a new Access Control Database** to create a new *empty* access control database and place it in the archives directory beneath the target project database. This database contains the Administrator user and default privileges.

10 Click **Finish**.

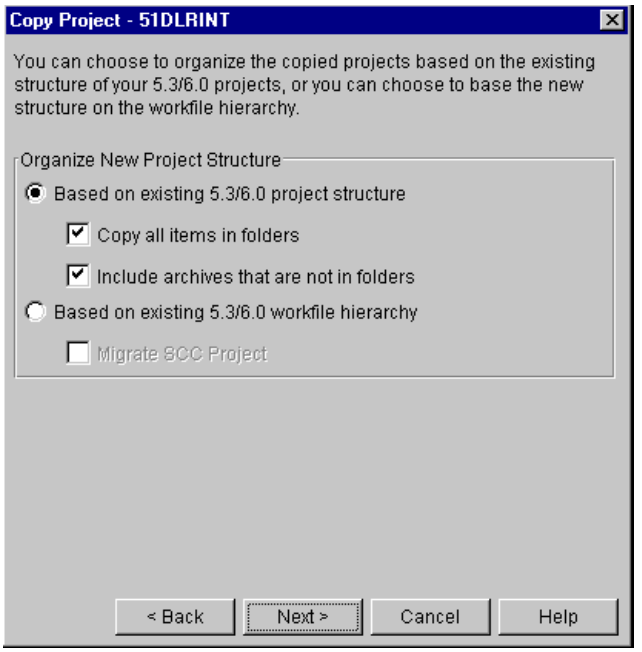
Copying 5.3/6.0 Projects Using Drag and Drop

To copy 5.3/6.0 projects using drag and drop:

1 Select the project you want to copy.

2 Drag the project into the destination project or project database. The Confirm Item Copy message appears.

3 Click **Yes**. The Copy Project dialog box appears.



4 In the Organize New Project Structure group, select:

- **Based on existing 5.3/6.0 Project structure** if you want to organize the new project based on the structure of the existing 5.3/6.0 project. This structure will contain no more than two levels: a project and a subproject. This is the default option.

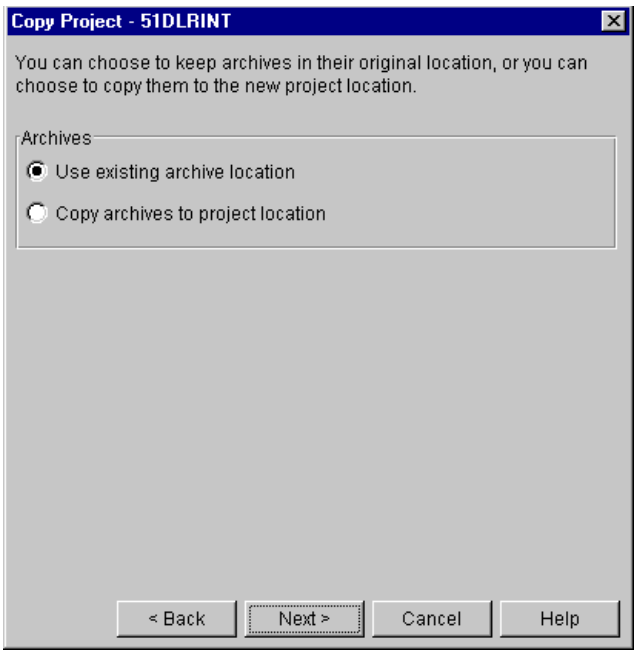
  If you want to copy all archives that are stored in folders, check the **Copy all items in folders** check box.

  If you want to copy archives that are not in folders (archives located at the project level), check the **Include archives that are not in folders** check box. This is the default option.

- **Based on existing 5.3/6.0 workfile hierarchy** if you want to organize the new project based on the 5.3/6.0 workfile hierarchy.

- **Migrate SCC project** if you want to migrate an SCC 5.3/6.0 project. You must also select the **Based on existing 5.3/6.0 workfile hierarchy** option.

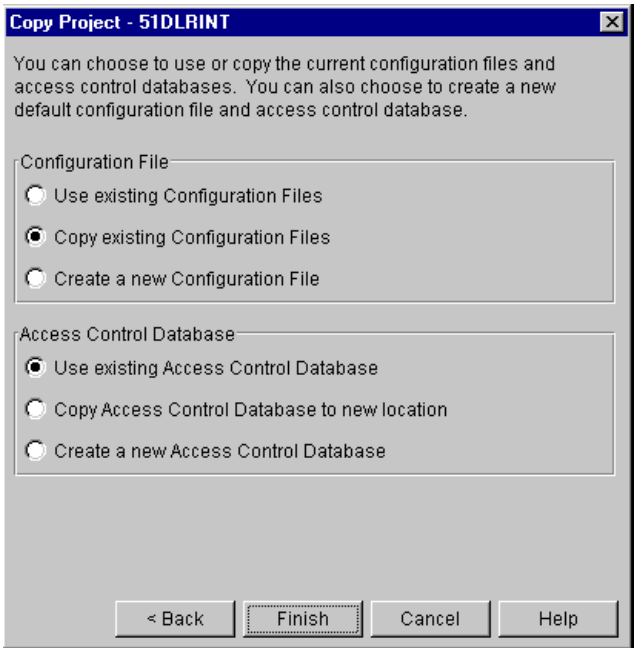
5 Click **Next**. A second Copy Project dialog box appears.



6 In the Archives group, select:

- **Use existing archive location** if you want to copy the project to a new location but continue to reference the existing archives in their current location. This is the default option.
- **Copy archives to project location** if you want to copy the project's archive directory (and its contents) and place it beneath the archives directory of the target project database. The location of the new archive directory mirrors the hierarchy of the target project database within the archives directory.

7 Click **Next**. A third Copy Project dialog box appears.



8 If your project does not use a configuration file, then the options in the Configuration File group are grayed. If your project uses a configuration file, select:

- **Use existing Configuration Files** if you want to copy the project to a new location but continue referencing the existing configuration file. The copied project shares the configuration file with the original project.

IMPORTANT! We do not recommend sharing configuration files if the 5.3/6.0 project you are copying uses an access control database.

- **Copy existing Configuration Files** if you want to make a copy of the existing configuration file and place it in the archives directory beneath the target project database. This is the default option.
- **Create a new Configuration File** if you want to create a default configuration file and place it in the archives directory beneath the target project database. The default settings of your configuration file are controlled by your Administrator.

9 If your project does not use an access control database, then the options in the Access Control Database group are grayed. If your project uses an access control database, select:

- **Use existing Access Control Database** if you want to copy the project to a new location but continue referencing the existing access control database. This is the default option.
- **Copy Access Control Database to new location** if you want to make a copy of the existing access control database and place it in the archives directory beneath the target project database.
- **Create a new Access Control Database** to create a new *empty* access control database and place it in the archives directory beneath the target project database. This database contains the Administrator user and default privileges.

10 Click **Finish**.

Copying 5.3/6.0 Project Roots

You can copy (upgrade) 5.3/6.0 project roots to the newer project database format to take advantage of all the product functions in the newest release. When you copy a 5.3/6.0 project root, Version Manager upgrades the 5.3/6.0 project root into a project database. You must use the Menu bar to copy a 5.3/6.0 project root; drag and drop is not supported.

You can copy (upgrade) 5.3/6.0 project roots that were created using the Version Manager 5.3/6.0 GUI or one of the Version Manager Development Interfaces. For complete information about how to upgrade 5.3/6.0 projects that were created in one of the Version Manager Development Interfaces, refer to the implementation guide for your development environment.

Before you upgrade your 5.3/6.0 project root, make sure to read the section "Working with Version Manager 5.3/6.0 Project Roots," in the *Serena PVCS Version Manager Getting Started Guide*.

If you are copying a 5.3/6.0 project root to a project database, do not create a project database before you begin. Version Manager creates a new project database as part of

the copy procedure. You cannot copy a 5.3/6.0 project root into an existing project database.

When you copy a 5.3/6.0 project root, you can:

- Organize the new project database based on the structure of the existing 5.3/6.0 project root or base it on the 5.3/6.0 workfile hierarchy. Organizing the new project database based on the:
  - Structure of the 5.3/6.0 project root creates a project database structure identical to the project root structure in the 5.3/6.0 desktop client. The 5.3 project root becomes a project database, each 5.3/6.0 project becomes a project, and each folder becomes a subproject.
  - 5.3/6.0 workfile hierarchy creates a project database structure identical to your workfile hierarchy. If your workfile hierarchy is nested, Version Manager creates a nested project structure to match your workfile hierarchy.
- Copy the archives to the new archive location or use the existing archive location.

NOTE When copying a project root that contains multiple projects, remember that the projects may have different archive locations. If you copy a project root and use the existing archive locations, the new project database still references all of the different archive locations.

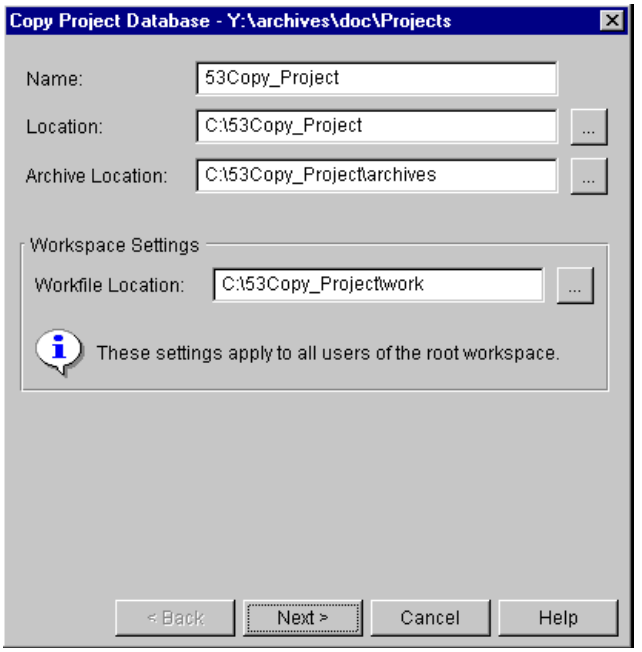
Any new archives added to the project root after it is copied are placed in the archives directory of the project database.

- Copy the existing configuration files to the new project database location, use the existing configuration files, or create new configuration files.
- Copy the 5.3/6.0 project root's existing access control database to the new project database location, use the existing access control database, or create a new access control database.

To copy a 5.3/6.0 project root:

1 Select the 5.3/6.0 project root you want to copy.

2 Select Edit | Copy. The Copy Project Database dialog box appears.



3 Enter a new project database name in the **Name** field. Project database names cannot begin or end with a tab or blank space. No other character restrictions apply to a project database name.

4 Enter a location for the new project database in the **Location** field or click the Browse button to select a location. Notice that the location you enter is also reflected in the **Archive** field.

This location must be accessible to all users who will be accessing this project database. You cannot create a project database beneath another project database; therefore, we recommend that you do not create a project database at the root level of a drive. You also cannot create a project database in the same location as an existing 5.3/6.0 project root.

5 A new archive directory relative to the project database location you entered is specified in the **Archive Location** field. If you want to specify a different archive location, enter it in this field or click the Browse button to select a location. This location must be accessible to all users who will be accessing the archives.

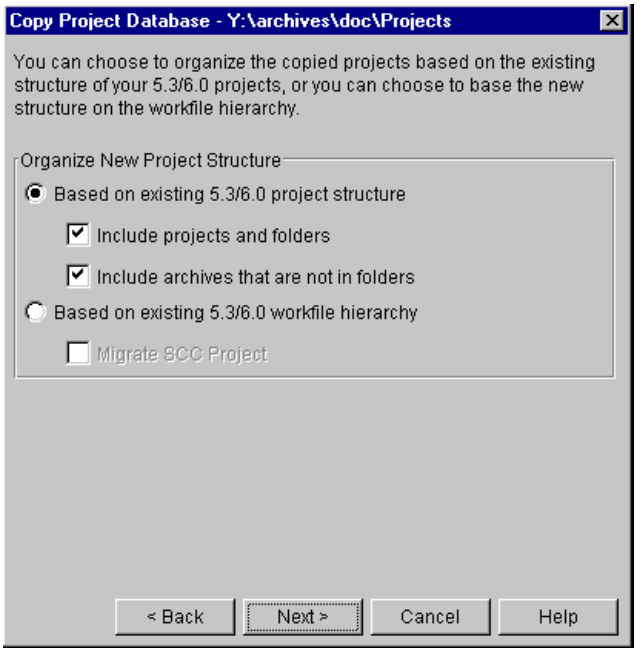
6 Specify a workfile location by entering it in this field or click the Browse button to select a location.

Workfile locations can contain `$HOME` at the root of their path (for example, on UNIX, `$HOME/work` could expand to `/usrs/cherylc/work`). Version Manager substitutes the value of the HOME environment variable to compute the workfile location. The use of `$HOME` allows you to define a path that is automatically individualized for your users according to the value of their HOME environment variable. If a user's HOME environment variable is not

defined, Version Manager substitutes a blank space when computing the workfile location.

NOTE We recommend that you specify a local drive that is available to most users. Remember that users can choose to create a private workspace to change the workfile location to a different location.

7 Click **Next**. The second Copy Project Database dialog box appears.



8 In the Organize New Project Structure group, select:

- **Based on existing 5.3/6.0 Project structure** if you want to organize the new project database based on the structure of the existing 5.3/6.0 project root. This structure will contain no more than two levels: a project and a subproject. This is the default option.

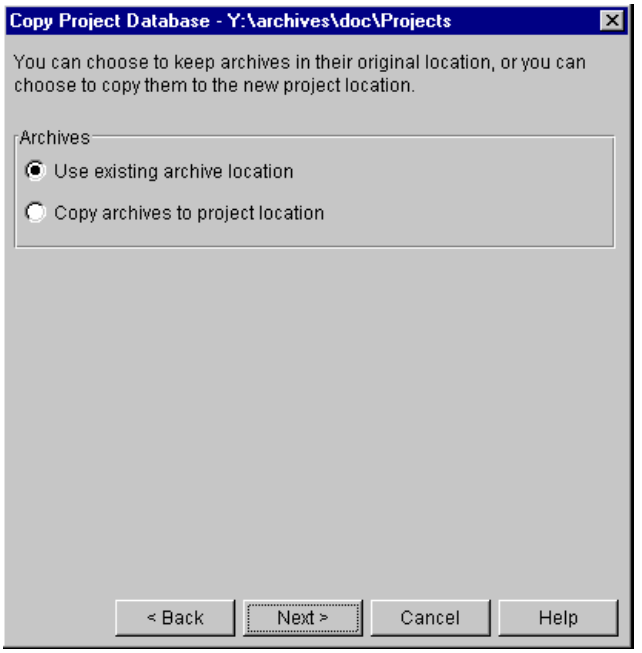
  If you want to copy all archives that are stored in folders, check the **Copy all items in folders** check box. This option is available only if you choose to create the project based on the existing 5.3/6.0 project structure.

  If you want to copy archives that are not in folders (archives located at the project root level), check the **Include archives that are not in folders** check box. This option is only available if you choose to create the project based on the existing 5.3/6.0 project structure. This is the default option.

- **Based on existing 5.3/6.0 workfile hierarchy** if you want to organize the new project database based on the 5.3/6.0 workfile hierarchy. The advantage of this option is that Version Manager will create a nested project structure if you used subdirectories to organize the workfiles.

- **Migrate SCC project** if you want to migrate an SCC 5.3/6.0 project. You must also select the **Based on existing 5.3/6.0 workfile hierarchy** option.

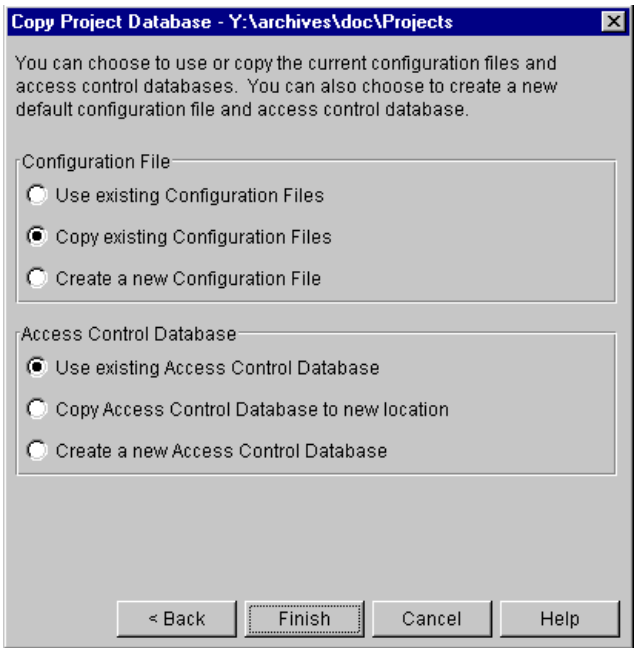
9 Click **Next**. The third Copy Project Database dialog box appears.



10 In the Archives group, select:

- **Use existing archive location** if you want to copy the project root to a new location but continue to reference the existing archives in their current location. This is the default option.
- **Copy archives to project location** if you want to copy the project's archives and place them in directories beneath the archives directory of the new project database. The location of the archives mirrors the hierarchy of the new project database within the archives directory.

11 Click **Next**. A fourth Copy Project dialog box appears.



12 If your project root does not use a configuration file, then the options in the Configuration File group are grayed. If your project root uses a configuration file, select:

- **Use existing Configuration Files** if you want to copy the project root to a new location but continue referencing the existing configuration file. The new project database shares the configuration file with the 5.3/6.0 project root.

IMPORTANT! We do not recommend sharing configuration files if the 5.3/6.0 project root you are copying uses an access control database.

- **Copy existing Configuration Files** if you want to make a copy of the existing configuration file and place it in the archives directory beneath the new project database. This is the default option.
- **Create a new Configuration File** if you want to create a default configuration file and place it in the archives directory beneath the new project database. The default settings of your configuration file are controlled by your Administrator.

13 If your project root does not use an access control database, then the options in the Access Control Database group are grayed. If your project root uses an access control database, select:

- **Use existing Access Control Database** if you want to copy the project root to a new location but continue referencing the existing access control database. This is the default option.
- **Copy Access Control Database to new location** if you want to make a copy of the existing access control database and place it in the archives directory beneath the new project database.
- **Create a new Access Control Database** to create a new *empty* access control database and place it in the archives directory beneath the new project database. This database contains the Administrator user and default privileges.

14 Click **Finish**.

Moving Items

You can move items using the Menu bar or dragging and dropping the items. You can move the following items:

You can move a...	To...
5.3/6.0 folder	Another project within the same 5.3/6.0 project root
5.3/6.0 versioned file	Another folder within the same 5.3/6.0 project root
project/subproject	Another project or subproject within the same project database
versioned file	The root of the project database or to a project or subproject within the same project database

You cannot move versioned files or projects to different project databases. You cannot move project databases. You also cannot move 5.3/6.0 projects.

Moving versioned files

When you move versioned files, the archives that they reference do not move; versioned files continue to reference the archives in their original location.

Moving projects/ folders

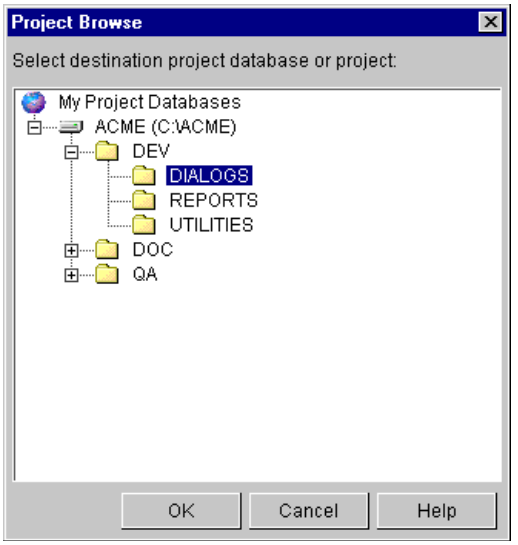
When you move a project or folder, Version Manager moves the entire contents of the project (including subprojects) or folder to the selected destination. Projects and subprojects continue to reference their configuration files in their original location. They also retain their workspace settings.

Moving Items Using the Menu Bar

Menu bar method

To move items using the Menu bar:

1. Select the items you want to move.
2. Select Edit | Move. The Project Browse dialog box appears.



3. Select the destination project or the project database (to move the project directly under the project database) and click **OK**.

Moving Items Using Drag and Drop

Drag and drop method

Move versioned files by dragging them from the File pane and dropping them into the Project pane. Move projects by dragging and dropping them within the Project pane.

To move items using drag and drop:

1. Select the items you want to move.
2. Drag the items into the destination project or the project database.

Deleting Items

If your administrator has assigned to you the necessary privileges, you can delete versioned files, revisions, projects, and subprojects. Deleting versioned files, projects, and subprojects does *not* delete the corresponding archives.

For information about the privileges necessary to delete items, see the "Using Security" chapter in the *Serena PVCS Version Manager Administrator's Guide*.

To delete an item:

1. Select the item you want to delete.
2. Select File | Delete. The Confirm Item Deletion message appears.
3. Click **Yes** to delete the item.

If you selected a...	Then...
Project or subproject	The project is removed from the Project pane, but the archives remain in the archives directory.
Versioned file	The versioned file is removed from the File pane, but the archive remains in the archives directory.
Revision	The revision is removed from the Revision pane. **IMPORTANT!** The revision is permanently removed from the archive and is *not* recoverable.

Restoring Versioned Files

If you have mistakenly deleted versioned files and now need to add them back to their original source control project, you can recreate the versioned files by importing the archives back into the project. See "Importing Archives" on page 97.

About Cleaning Up Unwanted Archive Files

Archive files are *not* deleted when you delete versioned files. If you wish to clean up these files, first make sure that *all* of the following are true:

- Your company's procedures allow the removal of archive files.
- The archive files are not being used by other projects (for example, as linked versioned files or shared archives).

TIP If you have a Serena support account, you can download a PCLI script that lists unused archive files. Search for KnowledgeBase article 34462 at support.serena.com/.

- The archive files are not needed to reproduce or maintain past work.
- The archive files have been backed up.

Then the Version Manager administrator could cleanup the unneeded archives.

IMPORTANT! *Always* backup the archive files before working directly with them.

Filtering Your View

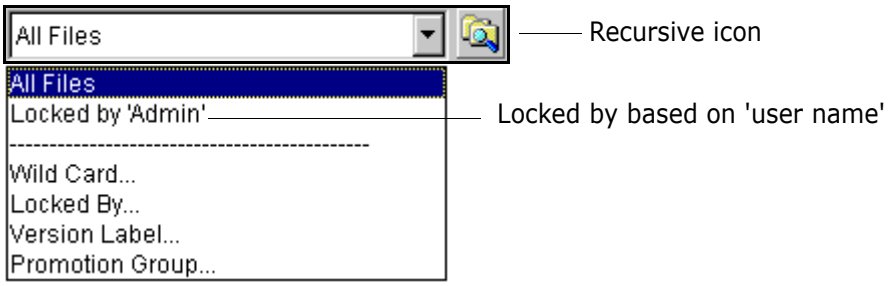
By default, when you select a project database, project, or folder, Version Manager displays all of the versioned files associated with your selection in the File pane. The file filter allows you to filter (customize) the File pane so that only the versioned files that match your filter criteria appear. Version Manager saves the last four filters you have defined so that you can reuse the filters.

Why filter files?

The file filter is very useful when you want to perform actions on a group of versioned files that match a filterable criteria (such as the same version label, similar filename, same user lock).

When you filter your view, the file filter affects all of the project databases that are currently open. Filters are available by selecting View | Filters from the Menu bar, or from the file filter list on the tool bar.

The file filter is shown below. The All Files option is the default option and displays all versioned files in all projects and folders.



Types of filters

You can filter your view to display only the versioned files that:

- Are locked by specific users or by any user
- Match a wildcard filename pattern
- Match a specified promotion group
- Match a specified version label
- Differ or match in terms of revision number, modification time, or revision contents between:
  - Two version labels
  - Two promotion groups
  - A version label and a promotion group

Performing actions on a filtered project

If you perform an action (such as a check out) on a project by selecting it in the Project pane, Version Manager performs the action on the entire project, *including* any files not displayed by the current filter. To limit an action to only the versioned files displayed by the current filter, select the versioned files in the File pane and then select the action.

NOTE A filtered view remains in effect until you select a different one. To avoid confusion, you may want to clear a filtered view by returning to the All Files view before going on to other work.

Viewing Files Recursively

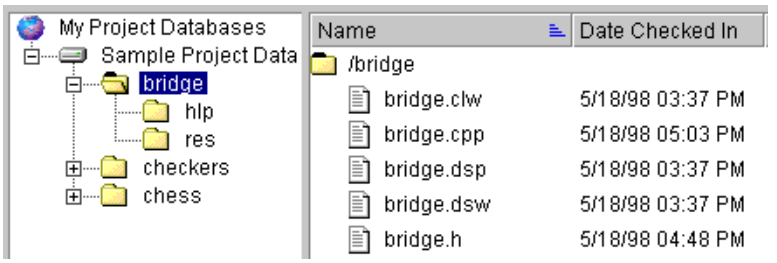
By default, the File pane displays only the versioned files located at the root of the item (project database, project, or subproject) you have selected in the Project pane. Unless you select the recursive view, projects, subprojects, and the files contained within nested projects and subprojects are not displayed.

To switch between recursive and non-recursive views:

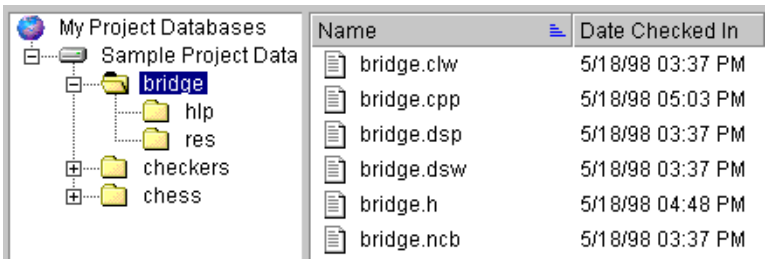
1 In the Project pane, select the project database or project that you want to view.

2 Click the **Recursive/Non-recursive** button. The appearance of the button toggles to indicate which filter mode is in effect:

- **Recursive:** The File pane displays all projects, subprojects, and versioned files contained within the item selected in the Project pane.



- **Non-recursive:** The File pane displays only the versioned files contained at the root of the item selected in the Project pane.

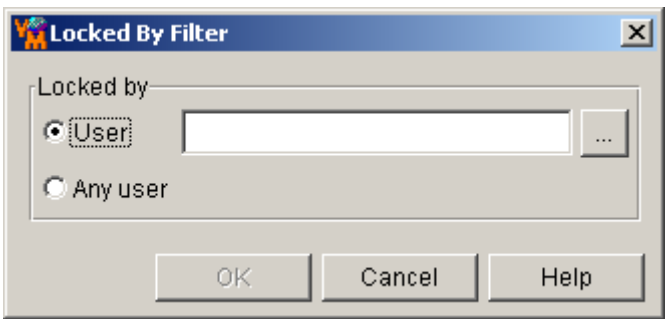


Filtering by Locker

Use this filter to find versioned files locked by specified users or by any user.

To set a locked by filter:

1 Select View | Filter | Locked By, or select the **Locked By** option from the file filter list on the tool bar. The Locked By Filter dialog box appears.



2 Do one of the following:

- To display versioned files locked by specific users, select the **Users** option and enter one or more user IDs. Separate multiple user IDs with the delimiter character. The default delimiter is a semicolon (;). For information on setting the delimiter character, see "Specifying a Delimiter for Items Entered in Fields" on page 134.
- To display all locked versioned files regardless of who has them locked, select the **Any user** option.

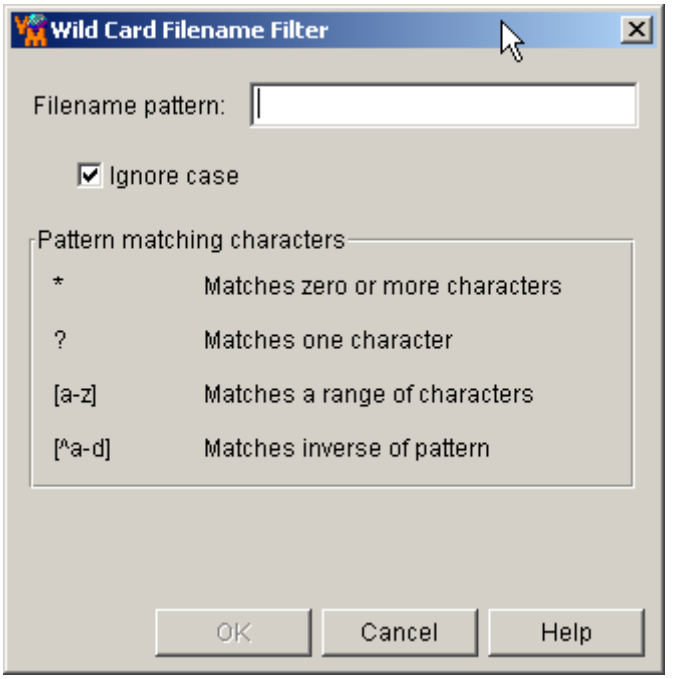
3 Click **OK**.

Filtering by Wildcard Filename

Use this filter to find versioned files that match specified wildcard name patterns.

To set a wildcard filter:

1 Select View | Filter | Wild Card, or select the **Wild Card** option from the file filter list on the tool bar. The Wild Card Filename Filter dialog box appears.



2 Enter filter criteria into the **Filename pattern** field.

Use this wild card...	To search for filenames that match...
*	Zero or more characters. For example, if you want to search for all versioned files with a DLL extension, you would enter `*.DLL`.
?	One character. For example, if you want to search for versioned files that had a pattern in the filename, such as TEST01.DLL, TEST02.DLL, … TESTXX.DLL, you would enter `TEST??.DLL`.

Use this wild card...	To search for filenames that match...
[-]	A range of characters. For example, if you want to search for all versioned files that start with A, B, C, or D, you would enter `[A-D]*`.
^	A negative character set matching any character not enclosed (for example, [^A-D]* for all file names that *do not* start with A, B, C, or D).

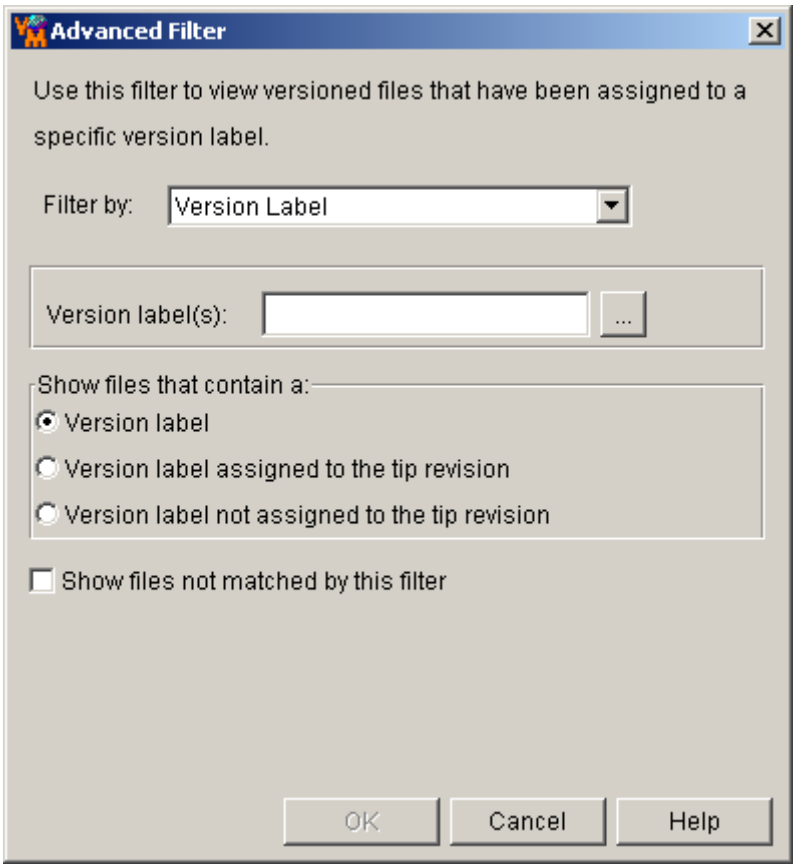
3 Click **OK**.

Filtering by Version Label

Use this filter to find versioned files that include specified version labels.

To set the version label filter:

1 Select View | Filter | Advanced Filter, or select the **Advanced Filter** option from the file filter list on the tool bar. The Advanced Filter dialog box appears.



2 Enter the name of the version labels that you want to filter by in the **Version label(s)** field or browse to select them. Separate multiple version labels with the delimiter character. The default delimiter is a semicolon (;). For information on setting

the delimiter character, see "Specifying a Delimiter for Items Entered in Fields" on page 134.

NOTE Version labels are case sensitive. Make sure that you use the correct case.

3 Select one of the following options to show versioned files that contain the specified version labels in:

- Any revision: **Version label**.
- The tip revision: **Version label assigned to the tip revision**.
- Any revision except for the tip revision: **Version label not assigned to tip revision**.

4 To inverse the logic of the filter, select the **Show files not matched by this filter check box**.

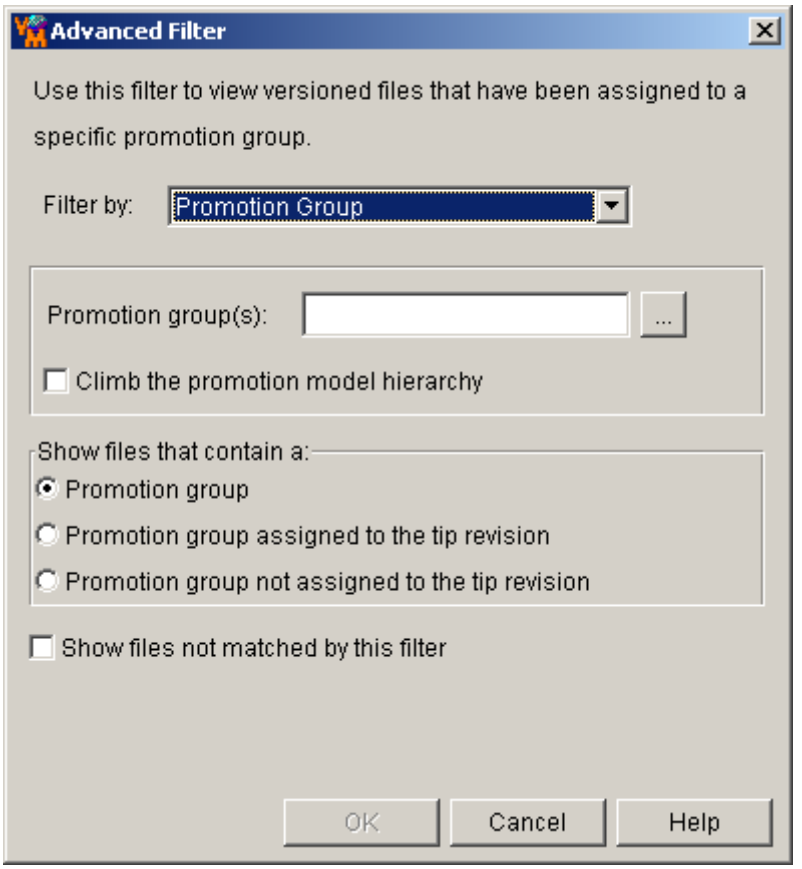
5 Click **OK**.

Filtering by Promotion Group

Use this filter to find files that include specified promotion groups.

To set the promotion group filter:

1 Select View | Filter | Advanced Filter, or select the **Advanced Filter** option from the file filter list on the tool bar. The Advanced Filter dialog box appears.



2 Select the **Promotion Group** option from the **Filter by** list.

3 Enter the names of the promotion groups that you want to filter by in the **Promotion group(s)** field or browse to select them. Separate multiple promotion groups with the delimiter character. The default delimiter is a semicolon (;). For information on setting the delimiter character, see "Specifying a Delimiter for Items Entered in Fields" on page 134.

4 If the specified promotion group is not found, the filter can search for the next highest promotion group, continuing up the promotion model hierarchy until a match is found or the highest level of the promotion model is reached. To enable this feature, select the **Climb the promotion model hierarchy** check box.

5 Select one of the following options to show versioned files that contain the specified promotion groups in:

- Any revision: **Promotion group**.
- The tip revision: **Promotion group assigned to the tip revision**.
- Any revision except for the tip revision: **Promotion group not assigned to tip revision**.

6 To inverse the logic of the filter, select the **Show files not matched by this filter check box**.

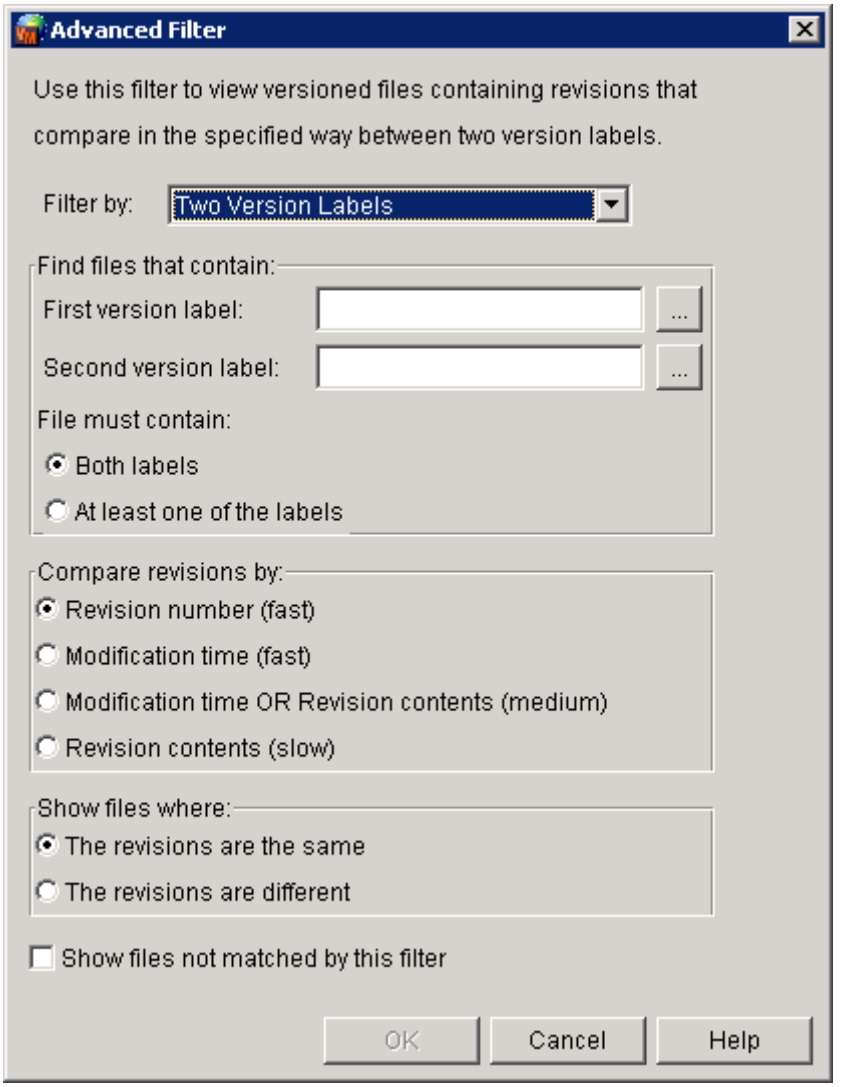
7 Click **OK**.

Filtering by Comparing Two Version Labels

Use this filter to find versioned files with revisions that differ or match between two specified version labels.

To set a compare filter using two version labels:

1 Select View | Filter | Advanced Filter, or select the **Advanced Filter** option from the file filter list on the tool bar. The Advanced Filter dialog box appears.



2 Select the **Two Version Labels** option from the **Filter by** list.

3 Enter the name of a version label that you want to filter by in the **First version label** field or browse to select one.

NOTE Version labels are case sensitive. Make sure that you use the correct case.

4 Enter the name of a version label that you want to filter by in the **Second version label** field or browse to select one.

5 Specify whether the files must contain:

- **Both labels**
- **At least one of the labels**

6 Select one of the following options to specify how the revisions are compared:

- **Revision number.**
- **Modification time.**
- **Modification time OR Revision contents:** First compares the modification time of the revisions. If the modification time does not differ, then compares the revision contents. This option takes longer if many revisions have the same modification time.
- **Revision contents:** Compares the contents of revisions. This option takes the most time.

7 Specify whether to show files where:

- **The revisions are the same**
- **The revisions are different**

8 To inverse the logic of the filter, select the **Show files not matched by this filter check box**.

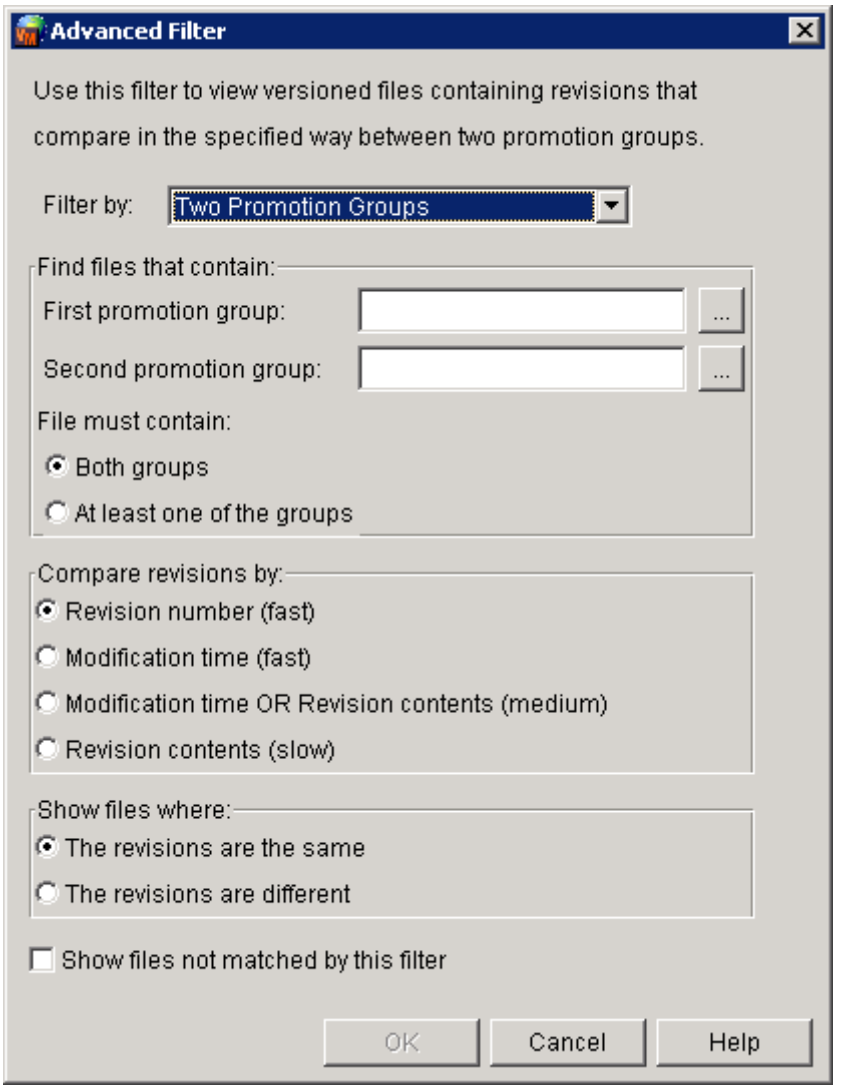
9 Click **OK**.

Filtering by Comparing Two Promotion Groups

Use this filter to find versioned files with revisions that differ or match between two specified promotion groups.

To set a compare filter using two promotion groups:

1 Select View | Filter | Advanced Filter, or select the **Advanced Filter** option from the file filter list on the tool bar. The Advanced Filter dialog box appears.



2 Select the **Two Promotion Groups** option from the **Filter by** list.

3 Enter the name of a promotion group that you want to filter by in the **First promotion group** field or browse to select one.

4 Enter the name of a promotion group that you want to filter by in the **Second promotion group** field or browse to select one.

5 Specify whether the files must contain:

- **Both groups**
- **At least one of the groups**

6 Select one of the following options to specify how the revisions are compared:

- **Revision number.**
- **Modification time.**

- **Modification time OR Revision contents:** First compares the modification time of the revisions. If the modification time does not differ, then compares the revision contents. This option takes longer if many revisions have the same modification time.
- **Revision contents:** Compares the contents of the revisions. This option takes the most time.

7 Specify whether to show files where:

- **The revisions are the same**
- **The revisions are different**

8 To inverse the logic of the filter, select the **Show files not matched by this filter check box**.

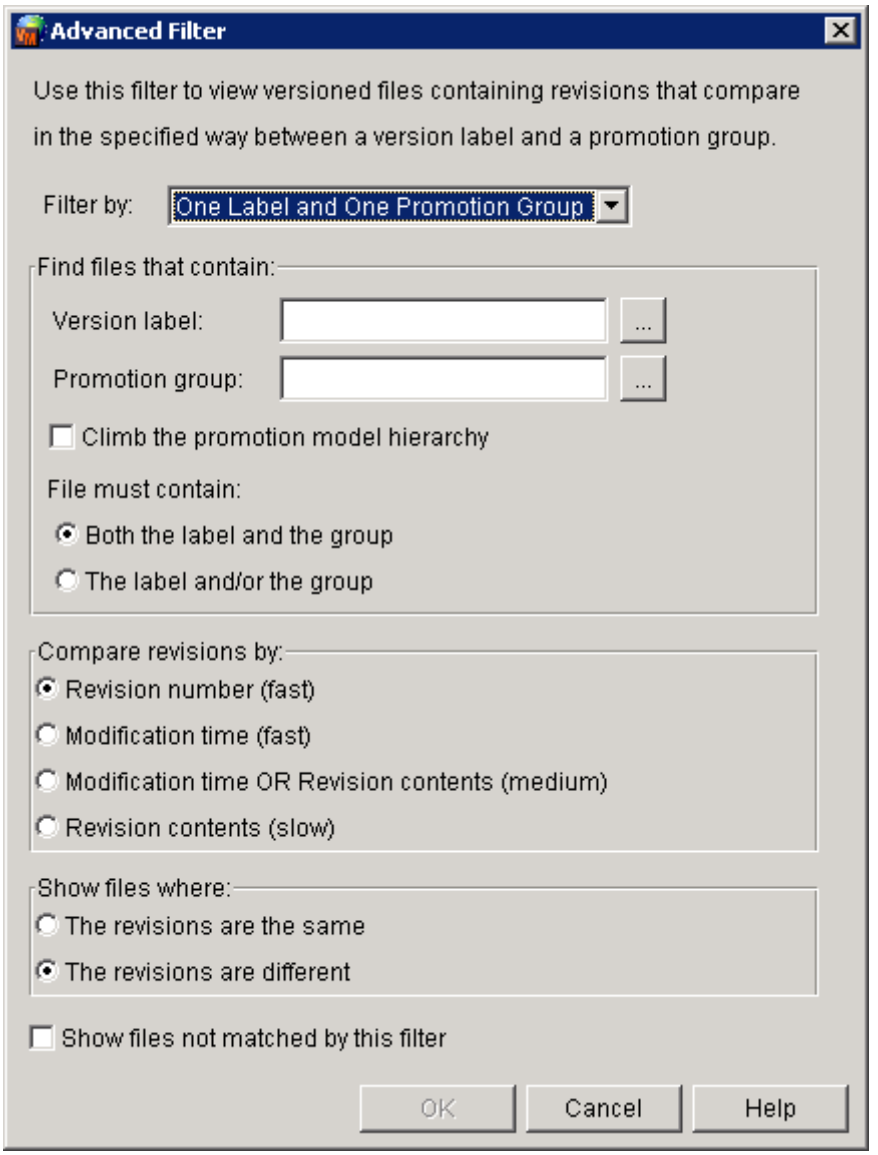
9 Click **OK**.

Filtering by Comparing a Version Label and a Promotion Group

Use this filter to find versioned files with revisions that differ or match between a specified version label and a specified promotion group.

To set a compare filter using a version label and a promotion group:

1 Select View | Filter | Advanced Filter, or select the **Advanced Filter** option from the file filter list on the tool bar. The Advanced Filter dialog box appears.



2 Select the **One Label and One Promotion Group** option from the **Filter by** list.

3 Enter the name of the version label that you want to filter by in the **Version label** field or browse to select one.

NOTE Version labels are case sensitive. Make sure that you use the correct case.

4 Enter the name of the promotion group that you want to filter by in the **Promotion group** field or browse to select one.

5 Specify whether the files must contain:

- **Both the label and the group**

- **The label and/or the group**

6 If the specified promotion group is not found, the filter can search for the next highest promotion group, continuing up the promotion model hierarchy until a match is found or the highest level of the promotion model is reached. To enable this feature, select the **Climb the promotion model hierarchy** check box.

7 Select one of the following options to specify how the revisions are compared:

- **Revision number.**
- **Modification time.**
- **Modification time OR Revision contents:** First compares the modification time of the revisions. If the modification time does not differ, then compares the revision contents. This option takes longer if many revisions have the same modification time.
- **Revision contents:** Compares the contents of the revisions. This option takes the most time.

8 Specify whether to show files where:

- **The revisions are the same**
- **The revisions are different**

9 To inverse the logic of the filter, select the **Show files not matched by this filter check box**.

10 Click **OK**.

Viewing All Versioned Files (No Filter)

To clear a filtered view and display all versioned files, select View | Filter | All Files, or select the **All Files** option from the file filter in the tool bar.

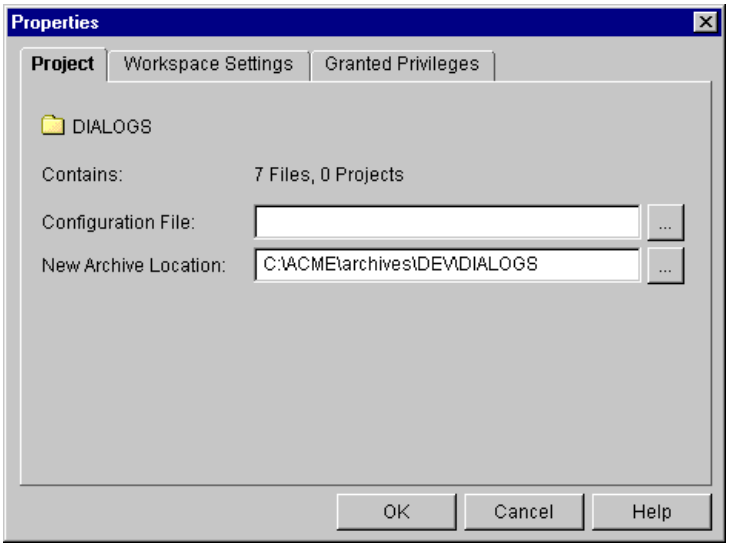
Reviewing Properties

You can review the properties of any item in the Version Manager desktop client.

To review an item's properties:

1 Select the item.

2 Select File | Properties. The Properties dialog box appears.



The appearance of the Properties dialog box and the properties that appear varies depending on the item selected.

If you selected a...	You can...
Project database	View or change the contents, configuration file, archive location, tool bar configuration file, workspace settings, and your user privileges.
Project or subproject	View or change the contents, configuration file, archive location, workspace settings, and your user privileges.
5.3/6.0 project root	View or change the contents, configuration file, tool bar configuration file, and workspace settings.
5.3/6.0 project	View or change the contents, configuration file, and workspace settings.
5.3/6.0 folder	View or change the contents and workfile location.
Versioned file	View archive information and change the description or workfile location. You can also display version labels, promotion groups, and branches associated with a revision.
Revision	View revision information and change the revision description. You can also display version labels, promotion groups, and branches associated with the revision.
Version label	View revision information and change the revision description. You can also display version labels, promotion groups, and branches.
Promotion group	View revision information and change the revision description. You can also display version labels, promotion groups, and branches.

Chapter 3

Working with Project Databases

About Project Databases

A project database is a container for project structures. From within the desktop client, a project database contains projects, subprojects, and versioned files. For each project database, there is a set of configuration options, one or more workspaces, one tool bar configuration file, and possibly an access control database.

By default, the location of the project database on the operating system contains the project database configuration file, and several directories for the user data, workfiles, and archives.

Typically, your Administrator creates project databases. As a user, you add directories of workfiles to the project database to create projects of versioned files.

For information about the structure of a project database, or how to create a project database, see the *Serena PVCS Version Manager Administrator's Guide*.

About the New Project Database

The first time you invoke Version Manager, you can create a new project database by selecting the **Create a new project database** check box. If you cannot create a new project database, check with your Administrator as this privilege may have been disallowed. You can use this new project database, along with the sample project database, to practice Version Manager tasks, or you can rename it and set it up to store your actual workfiles.

By default, the project database is created beneath the installation directory:

- For Windows: `Program Files\Serena\vm\Newdb`
- For UNIX: `/usr/serena/vm/unix/newdb`

The directories and files that are created on your operating system for the new project database are as follows:

This directory...	Contains...
Archives directory (`\newdb\archives`)	When you add projects to the project database, the archives are stored in this directory. This directory also contains the configuration file for the new project database and a default access control database.
Pvcsuser directory (`\newdb\pvcsuser`)	Contains user information, such as the last workspace you set on the project database and other user options.

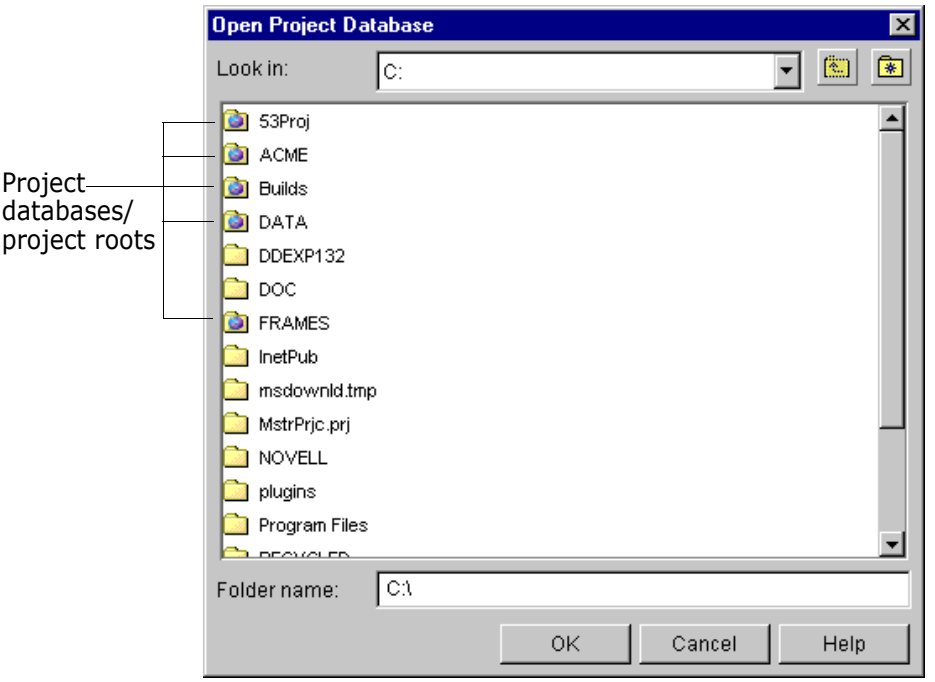
For more information about setting up a project database, see the *Serena PVCS Version Manager Administrator's Guide*.

Opening Project Databases

To access projects, subprojects, or versioned files under version control, you must open a project database.

To open a project database:

1 Select File | Open Project Database. The Open Project Database dialog box appears.



2 Browse to the location of the project database you want to open. Project databases and 5.3/6.0 project roots appear in the Open Project Database dialog box as a folder with the Serena Globe ( ).

NOTE For 5.3/6.0 projects roots, this is the location entered in Options | Data File Locations within the Version Manager 5.3/6.0 desktop client. It contains a Pvcsproj.pub file.

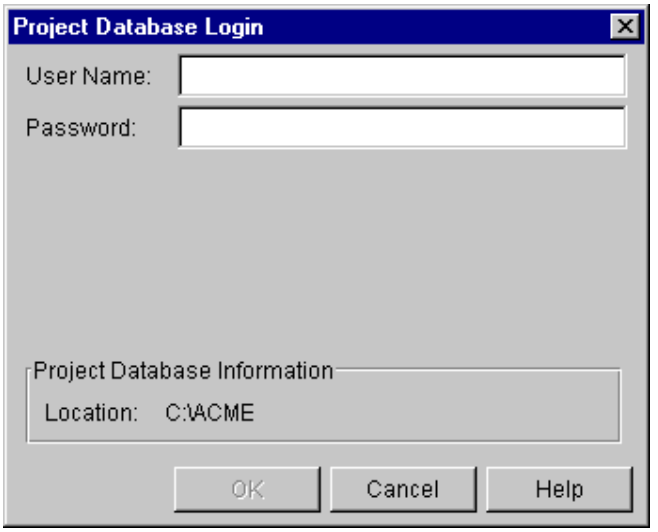
At the minimum, the project database directory contains a pvcsuser directory and Version Manager project database files (pvcsid.ser, pvcsroot.old, and pvcsroot.ser). Typically, unless the defaults are overridden, project databases contain an archives directory and a work directory.

NOTE Select **File Servers** from the **Look in** list to view a list of project databases available on Version Manager file servers.

3 Click **OK**. The project database you selected appears in the Project pane.

Logging In to Project Databases

If your Administrator requires that you enter a user name and password, the Project Database Login dialog box appears. Typically, your Administrator will assign you your user name and password.



To log in to a project database:

1 Enter your user ID in the **User Name** field.

2 Enter your password in the **Password** field.

3 Click **OK**. The project database appears in the Project pane.

NOTE If you cannot log in to the project database, contact your Administrator to verify your user ID and password.

Closing Project Databases

Project databases remain open in the Project pane until you close them.

To close a project database:

1 Select the project database you want to close.

2 Select File | Close Project Database.

Scenario: Opening and Logging In to an Existing Project Database

In preparation for rolling out Version Manager, Ruby, the System Administrator, tells Jacob the network locations for the product installation and the Games project database:

Version Manager is stored at `h:\vm` and the project database is stored at `h:\vm\games`. Jacob installs the product on his workstation and selects Serena | Version Manager | Version Manager from the Start menu.

Jacob browses to the location of the Games project database and attempts to open it. Because the Administrator has configured the project database to require a password, Jacob must enter his user name and password supplied by Ruby—**JacobL, PrjLead**. (See the section on Login Sources in the *Serena PVCS Version Manager Administrator's Guide* for information on specifying the user identification process.) The Games project database opens and displays in the Project pane.

The Games project database does not yet contain projects or versioned files. Jacob will populate the database with the four projects in the next scenario; however, the predefined project database has the following characteristics:

- **Configuration settings** that specify the default behavior for Version Manager features. To ensure that Version Manager operates the same for all users and to protect against inadvertent data corruption, Ruby has embedded a master configuration file that controls how Version Manager operates for all project databases, not just for the Games project database and its projects. In the master configuration file, Ruby has defined global options and disallowed the options she does not want others to reset for any project database or project. For example, Ruby's master configuration file specifies that all project databases consistently perform the following tasks:

  - Create archives when checking in files
  - Identify users via Netware IDs, Host IDs, and then the Login Dialog
  - Store the access control database at h:\vm
  - Allow multiple locks
  - Store semaphore files in a common network directory
  - Work with the same promotion model
  - Use –arc as the suffix for all archive files

  While users cannot modify the settings associated with these characteristics, they can modify the settings for the rest of the configuration options if they have the appropriate privileges. Refer to the *Serena PVCS Version Manager Administrator's Guide* for more information.

- **Workspace characteristics** that specify project database work settings. A workspace stores settings for the following:

  - **Default workfile locations** for the projects, subprojects, and versioned files contained in the project database. A default workfile location is the directory to which you check out revisions and from which you check in workfiles. When Jacob adds the projects, the workfile locations, unless overridden, are relative to the workfile location already specified by Ruby for the database. For example, the default workfile location for the Games project database is `h:\vm\games\work`. When Jacob adds the first project, Chess, Version Manager suggests a workfile location of `h:\vm\games\work\chess`. Subsequent scenarios describe how Jacob and the team modify these workfile locations to better support their development efforts.
  - **Default revision** acted upon when a user performs an action without explicitly selecting a particular revision. For example, the Administrator can specify a particular version label as the basis for acting on a file. In this case, Ruby has

defined the default revision as the latest, or *tip*, revision. When a Games Developer checks out a file without selecting a particular revision, Version Manager automatically checks out the latest version of the file.

- **Setup options for automatic branching and merging**. Subsequent scenarios in this guide describe branching and merging; the *Serena PVCS Version Manager Administrator's Guide* describes how to automate branching and merging tasks.

- **Security definitions,** stored in an access control database, identify the authorized users for the Games project database and the actions those users can perform. Previously, Jacob gave Ruby a complete list of the Developers, their roles, and their assigned projects. She used this information to add users and assign their privileges. Refer to the *Serena PVCS Version Manager Administrator's Guide* for details on setting up the access control database.

- **Custom tool bar** options provide shortcuts for frequently used applications or tasks. An Administrator can associate a different customized tool bar with each project database. Ruby has not defined either of these custom elements for the Games project database. Refer to the *Serena PVCS Version Manager Administrator's Guide* for more information.

Chapter 4

Working with Version Manager 5.3/6.0 Project Roots

About 5.3/6.0 Project Roots

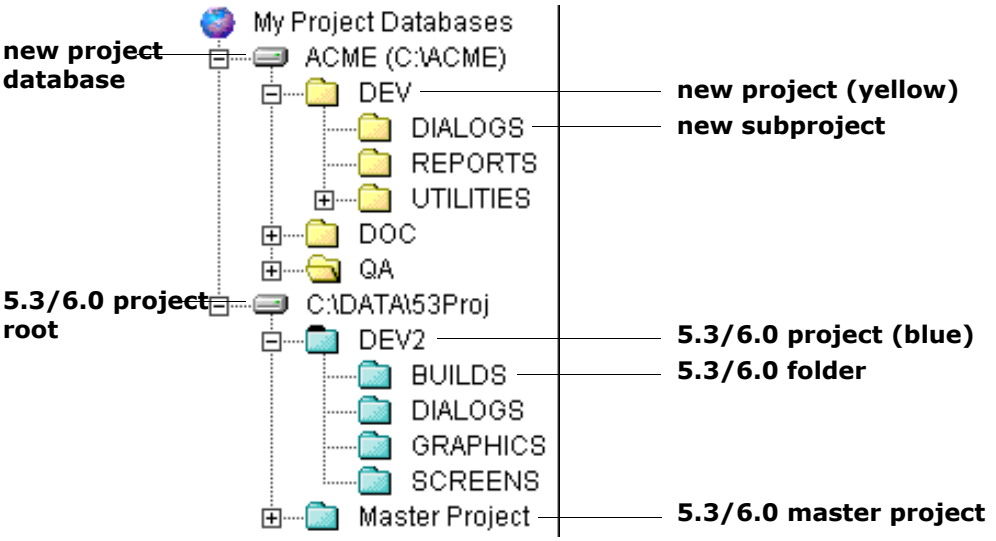
5.3/6.0 vs. new format

When working with version 5.3/6.0 Version Manager projects, you must determine whether you want to upgrade the project to the new Version Manager project format, or leave the project in the existing 5.3/6.0 format.

Before you upgrade your 5.3/6.0 project root, make sure to read the section "Working with Version Manager 5.3/6.0 Project Roots," in the *Serena PVCS Version Manager Getting Started Guide*.

Difference in appearance

You can open a 5.3/6.0 project root in the new Version Manager desktop client and continue to work with the projects and versioned files contained within it without upgrading to the new project format. When you open a 5.3/6.0 project root, it appears slightly different than a new project database, as shown below:



Some of these differences in appearance include:

- 5.3/6.0 project roots are displayed using only a project path and not a name.
- 5.3/6.0 project roots include master projects, which are not used in the project databases.
- The projects and folders within a 5.3/6.0 project root display in blue, whereas the projects and subprojects within a project database display in yellow so you can easily distinguish them in the desktop client.

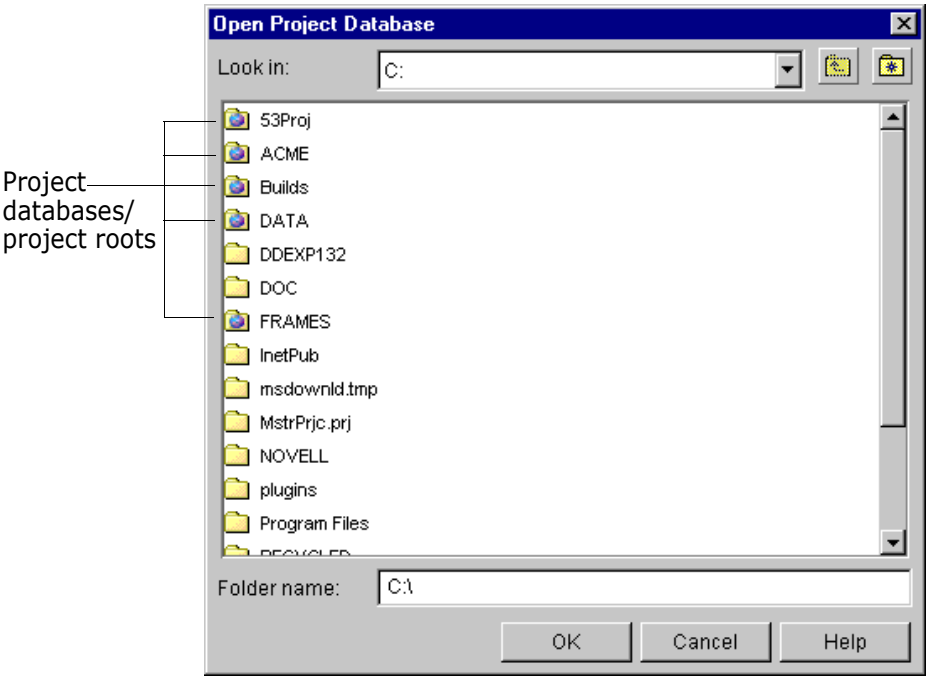
Opening 5.3/6.0 Project Roots

When you open a 5.3/6.0 project in the new desktop client, you will only have a subset of the functions available to you when working with the 5.3/6.0 project roots. These restrictions are necessary to ensure compatibility with 5.3/6.0 projects roots.

To access all of the new Version Manager features, you must copy the 5.3/6.0 project root into a new project database. For more information, see the section "Working with Version Manager 5.3/6.0 Project Roots" in the *Serena PVCS Version Manager Getting Started Guide*.

To open a 5.3/6.0 project root:

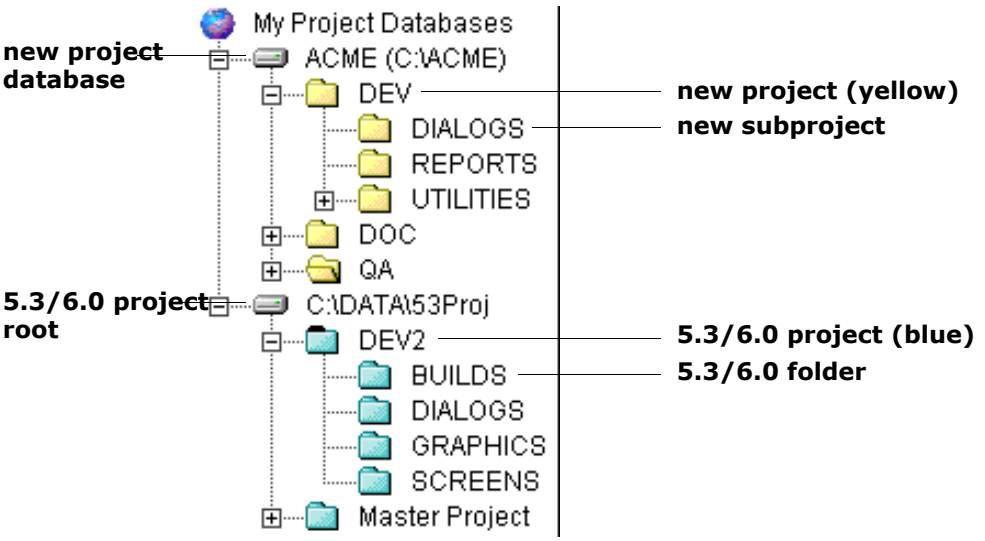
1 Select File | Open Project Database. The Open Project Database dialog box appears.



2 Browse to the location of the project root you want to open. A project root/project database appears in the Open Project Database dialog box as a folder with the Serena Globe ( ).

NOTE The location of the project root is the same as the Data File Location of the 5.3/6.0 project.

3 Select the 5.3/6.0 project root you want to open and click **OK**. The 5.3/6.0 project root appears in the Project pane.



4 Double-click the 5.3/6.0 project root to display the projects and folders associated with the project root.

Copying 5.3/6.0 Project Roots

You can copy 5.3/6.0 project roots into Version Manager databases. For information on copying 5.3/6.0 project roots, see "Copying 5.3/6.0 Project Roots" on page 56.

Closing 5.3/6.0 Project Roots

Version Manager 5.3/6.0 project roots remain open in the Project pane until you close them.

To close a 5.3/6.0 project root:

1. Select the 5.3/6.0 project root you want to close.
2. Select File | Close Project Database.

Scenario: Opening 5.3/6.0 Project Roots in Version Manager

Jacob is ready to add the four projects to the Games project database. To facilitate the smooth transition to Version Manager, Jacob wants to ensure that the Version Manager project database structure replicates that of the file hierarchies the Developers have been working with up to this point. The existing files are grouped by function under four main network directories:

Chess	Bridge	Checkers	Solitaire
K:\chess \client \board \images \server \library	K:\bridge \hlp \res	K:\checkers \client \board \images \server \library	K:\solitaire \lib \source \include \resource

Both the Chess and Bridge projects are currently under version control in Version Manager 5.3. Jacob wants to upgrade the Bridge project to take advantage of the new enhancements. However, the Administrator wants to preserve the original 5.3 project so that it reflects the previous release of the Bridge game. Jacob lets Ruby know that he is going to move the Bridge project into the new environment as a new project. To preserve the original project as a baseline before Jacob upgrades it to the new environment, Ruby locks the Bridge project in the 5.3 Version Manager environment. Jacob then opens the Bridge project in Version Manager. He selects the Bridge project, selects the Copy Project option, and chooses the following options:

- The Games project database as the target for the copied project.
- Copy Archives to Project Location to indicate that he wants to place a copy of the archives in a new directory under the main Games directory.
- Keep Specified Revision and Discard Change History to indicate that he is going to enter a new starting point for the new version of the Bridge project. He then enters

Bridge—1.0 as the Version Label that defines the starting point for the new version of the Bridge application. Version Manager does not copy over any of the revisions prior to those associated with this version label.

- Use an Existing Configuration File to indicate that the Bridge project will inherit the configuration options defined for the Games project database.
- Use an Existing Access Control Database to indicate that the Bridge project will inherit the security options defined for the Games project database.

Jacob selects Finish. The project now displays as a new project.

Although Jacob wants Developers to be able to access the Chess project via the new Version Manager interface, he does not want to upgrade it into the new Games project database. Some of the Chess team members work remotely from the rest of the team. These developers access the versioned files using the Version Manager web client, a browser-based interface to Version Manager. At this time, you cannot access project roots via the web client. However, Jacob does not want to support two different releases of Version Manager—5.3 and the new version—nor does he want developers to have to access two different versions of the product. Therefore, he asks the Chess developers to access the files through the new interface without upgrading to the new project format. Remote developers will continue to access the files through the Version Manager web client.

Jacob is now ready to create the new Version Manager projects.

Chapter 5

Adding Workfiles

About Adding Workfiles

When to add workfiles

Add workfiles whenever you want to populate a project database, project, subproject, or 5.3/6.0 project or folder.

Adding Workfiles to Project Databases/Projects

When adding workfiles, you can add individual files, a single directory, or an entire directory tree to a project database, project, or subproject. When you add workfiles, Version Manager automatically creates:

- A project with the same name as the directory (when you add a directory) and adds the files in the directory to the project
- An archive for each added workfile in the directory and checks in the initial revision
- A versioned file that references the newly created archive
- A subproject with the same name as the subdirectory (when you add a directory that contains a subdirectory) and optionally repeats the process recursively for all subdirectories of the directory being added

Associating issues

If you have installed TrackerLink or SourceBridge, you can associate issues with the versioned files from the Add Workfiles dialog box by clicking the **Associate Issues** button.

NOTE TrackerLink or SourceBridge is invoked automatically if you have set up Version Manager to require that you associate issues when adding workfiles.

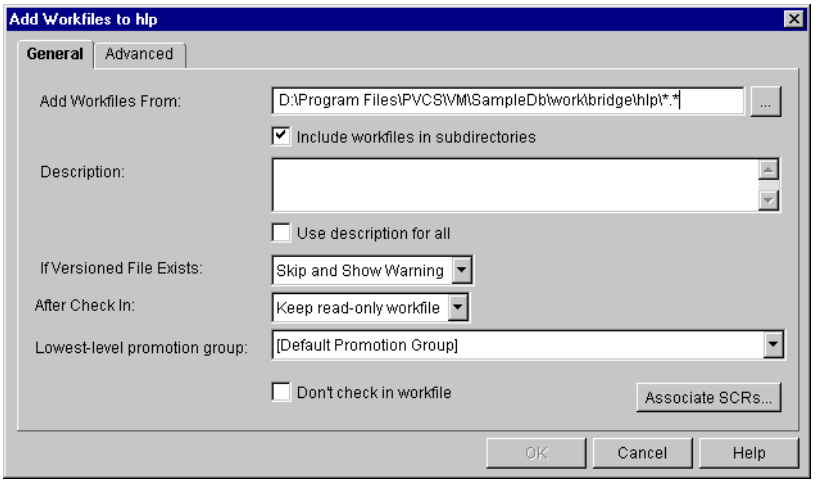
NOTE You can set which issue management integration will be used by the next invocation of the Version Manager Desktop Client. Any currently open client sessions will not be affected.

1. Launch the Serena Issue Management Integration utility (from the Serena folder of the Windows Start menu, select Version Manager | Issue Management Integration).
2. Select **TeamTrack SourceBridge** or **Tracker TrackerLink**.
3. Click the **OK** or **Launch Version Manager** button.

To add workfiles to project databases/projects:

1. Select the project database, project, or subproject to which you want to add workfiles.

2 Select File | Add Workfiles. The Add Workfiles dialog box appears.



3 Under the General tab, do the following:

a In the **Add Workfiles From** field, enter the location from which you want to add the workfiles or click the Browse button to select one.

NOTE If you specify a path with a filter other than `*` or `*.*` (for example, if you specify `c:\files\*.cpp`), the files matching the filter will be added directly without creating a top-level project. However, if subdirectories are also added, subprojects will be created for them and only files matching the filter will be added.

b If a directory is selected in step a, then you may also select the **Include workfiles in subdirectories** check box if you want Version Manager to add all of the files from the subdirectories of the selected directory.

c In the **Description** field, enter a workfile description for the files you are adding. This is a required field; the OK button is not enabled until you enter a description.

If you are adding a file for the first time, the text you enter becomes the archive description, and Version Manager enters "Initial revision" for the revision description in the Revision Pane. The archive description can be viewed by highlighting the versioned file and selecting File | Properties.

This process differs from the process of checking in a file. When you check in a file, the text entered in the **Description** field is displayed as the revision description in the Revision Pane.

d (For adding multiple workfiles only.) If you want Version Manager to prompt for a unique description for each workfile you are adding, deselect the **Use change description for all** check box. Otherwise, the same description will be used for every file.

e In the **If Versioned File Exists** field, select an action from the drop-down list that you want to occur if a versioned file with the same name exists in the project:

- **Skip** to skip the versioned file if it already exists in the project and do not display a warning.
- **Skip and Show Warning** to skip the versioned file and display a message indicating that a versioned file with the same name already exists in the project. This is the default.

f In the **After Check In** field, select an action from the drop-down list that you want to occur after Version Manager checks in your modified workfile:

- **Keep read-only workfile** to keep a read-only workfile in the workfile location. This is the default.
- **Keep revision locked** to lock the new revision after checking it in.
- **Delete workfile** to delete the workfile from the workfile location after checking it in.

g (Only available if a promotion model is in effect.) In the **Lowest-level promotion group** field, select a value from the drop-down list:

- Select a lowest-level promotion group to associate with the first revision of the workfile that you are adding.
- Select the value **[None]** to **not** associate a promotion group with the revision.

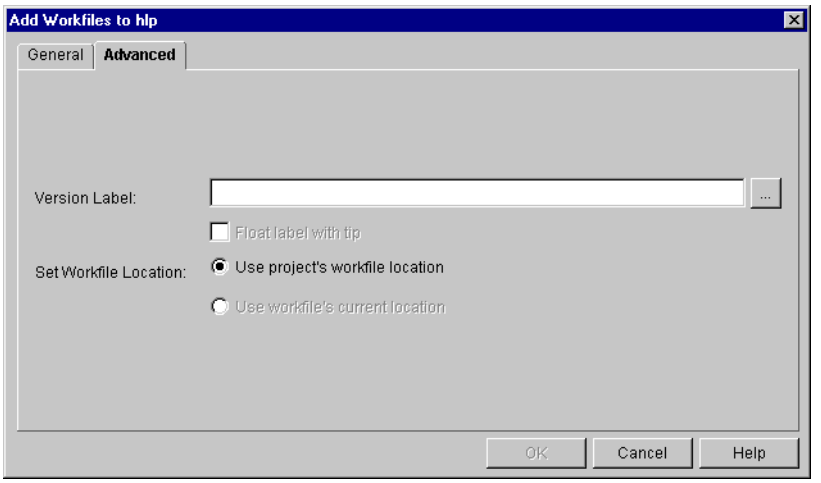
The default value is **[Default Promotion Group]**, which is a workspace setting that defines a lowest-level promotion group to use for this action. If a value for this workspace setting is **not** defined and you do not select a value, then no promotion group is associated with the revision.

h (Only available if you have Superuser or Unlimited privileges.) Select the **Don't check in workfile** check box to add the workfile (create an archive), but not check in the first revision of the versioned file.

i (For Serena TrackerLink and SourceBridge users only.) Click the **Associate Issues** button if you want to associate the workfiles you are adding with issues. This displays the association dialog box.

NOTE TrackerLink or SourceBridge is invoked automatically if you or your administrator has set it up to require that you associate issues when adding workfiles.

4 Under the Advanced tab, if you are checking in your workfiles, you can optionally do the following:



- In the **Version Label** field, assign a version label to the checked in revision by entering a version label or by clicking the Browse button to select an existing version label that is assigned to a versioned file. Note that version labels are case sensitive.

- Select the **Float label with tip** check box to keep the version label associated with the latest revision.
- Set the location of the workfile you are adding by choosing one of the following options (not available if the **Don't check in workfile** check box is selected on the General tab):
  - Select the **Use project's workfile location and copy workfile(s) into it** option to copy the file from its current location to the project's workfile location. The workfile will use the project's workfile location as its workfile location.

    This option also allows you to choose what you want to do if a workfile already exists in the workfile location. The options include **Skip** or **Overwrite**.

NOTE The above option is only available if you are not adding files from the workfile location. If they are already in the workfile location, then you cannot reset either radio button and the label reads **Use project's workfile location**.

  - Select the **Use workfile's current location** option to make the file's current location its workfile location.

5 Click **OK**.

Adding Workfiles to 5.3/6.0 Projects

You can add workfiles to a 5.3/6.0 project or folder. You cannot add workfiles to 5.3/6.0 project roots. You can add individual workfiles or a directory of workfiles. When adding a directory of workfiles, Version Manager only adds the workfiles located at the root level of the directory; it does not add workfiles located in subdirectories and it does not create projects or folders.

You cannot add files with the same name to a 5.3/6.0 project or folder. However, if you used the 5.3/6.0 desktop client to add files with the same name by defining separate archive and workfile locations, the new Version Manager desktop client will honor the duplicate names.

When you add workfiles, Version Manager automatically creates:

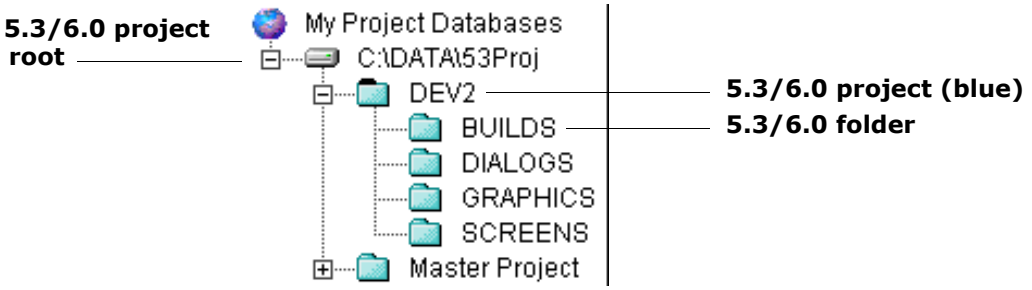
- An archive in the specified archive directory for each added workfile and checks in the initial revision.
- A versioned file that references the new archive.

Choosing a Workfile Destination

In the 5.3/6.0 desktop client, to get the workfiles to appear in a folder, you have to add the workfiles to the project and manually copy them to a project folder by selecting Folder | Change Folder Member.

You can still add workfiles directly to a 5.3/6.0 project and copy the versioned file to a 5.3/6.0 folder; however, Version Manager now supports adding workfiles directly to a 5.3/6.0 folder. When you add workfiles to a 5.3/6.0 folder, Version Manager creates a

versioned file in the folder and automatically copies the versioned file into the 5.3/6.0 project so you can access the archive from both the project and folder level.



Adding workfiles to a 5.3/6.0 folder automatically populates the 5.3/6.0 project.

5.3/6.0 project workfile location

Version Manager 5.3/6.0 projects only support one workfile location. If you want to store the existing workfiles in the 5.3/6.0 project's workfile location instead of copying the workfiles to the workfile location set for your current workspace, make sure that the:

- Workfiles you want to add are located in the 5.3/6.0 project's working directory as specified in the 5.3/6.0 desktop client (Project | Configure Project | Project Options).
- Workfile location for your current workspace is set for the 5.3/6.0 project's working directory. For more information on setting the workfile location for your current workspace, see "Using Workspaces" on page 109.

If the 5.3/6.0 project working directory and the workfile location of your current workspace are not in sync, your only option is to copy the workfiles to the workfile location set for your current workspace.

5.3/6.0 folder workfile location

Since 5.3/6.0 folders support multiple workfile locations, when adding workfiles, you have the option of using the existing workfile location or copying the workfiles to the workfile location set for your current workspace.

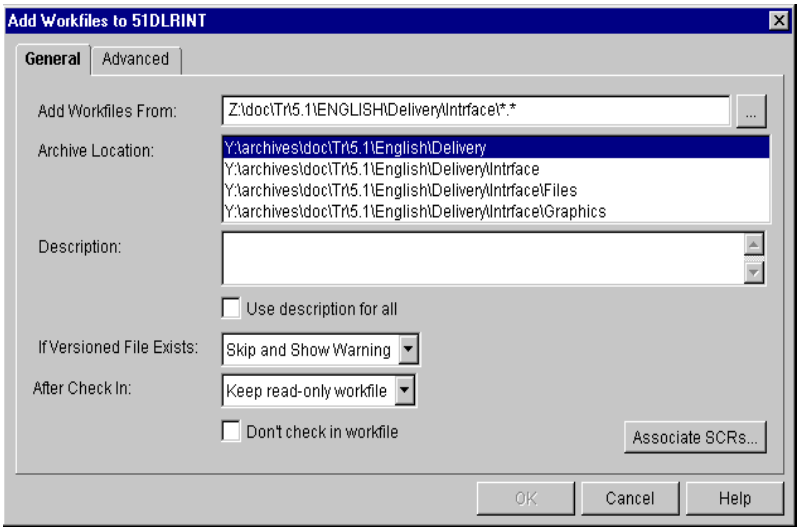
Choosing an Archive Directory

When you add workfiles to 5.3/6.0 projects, Version Manager requires you to select an archive directory; it does not automatically create archive directories based on your workfile hierarchy. If the archive directory does not exist, you must add the archive directory using the 5.3/6.0 desktop client.

To add workfiles to a 5.3/6.0 project:

1 Select the 5.3/6.0 project or folder to which you want to add workfiles.

2 Select File | Add Workfiles. The Add Workfiles dialog box appears.



NOTE The Lowest-level promotion group field does not show up on the screen because a promotion model is not defined for this project. See step g for a description of this field.

3 Under the General tab, do the following:

a In the **Add Workfiles From** field, enter the location from which you want to add the workfiles or click the Browse button to select one.

b In the **Archive Location** field, select the location in which to create the archives from the archive directories specified in the configuration file.

c In the **Description** field, enter a workfile description for the files you are adding. This field is required; the OK button is not enabled until you enter a description.

d (For adding multiple workfiles only.) If you want Version Manager to prompt for a unique description for each workfile you are adding, deselect the **Use change description for all** check box. Otherwise, the same description will be used for every file.

e In the **If Versioned File Exists** field, select an action from the drop-down list that you want to occur if a versioned file with the same name exists in the project:

- **Skip** to skip the versioned file if it already exists in the project and do not display a warning.
- **Skip and Show Warning** to skip the versioned file and display a message indicating that a versioned file with the same name already exists in the project. This is the default.

f In the **After Check In** field, select an action from the drop-down list that you want to occur after Version Manager checks in your modified workfile:

- **Keep read-only workfile** to keep a read-only workfile in the workfile location. This is the default.
- **Keep revision locked** to lock the new revision file after checking it in.
- **Delete workfile** to delete the workfile after checking it in.

g (Only available if a promotion model is in effect.) In the **Lowest-level promotion group** field, select a value from the drop-down list box:

- Select a lowest-level promotion group to associate with the first revision of the workfile that you are adding.
- Select the value **[None]** to **not** associate a promotion group with the revision.

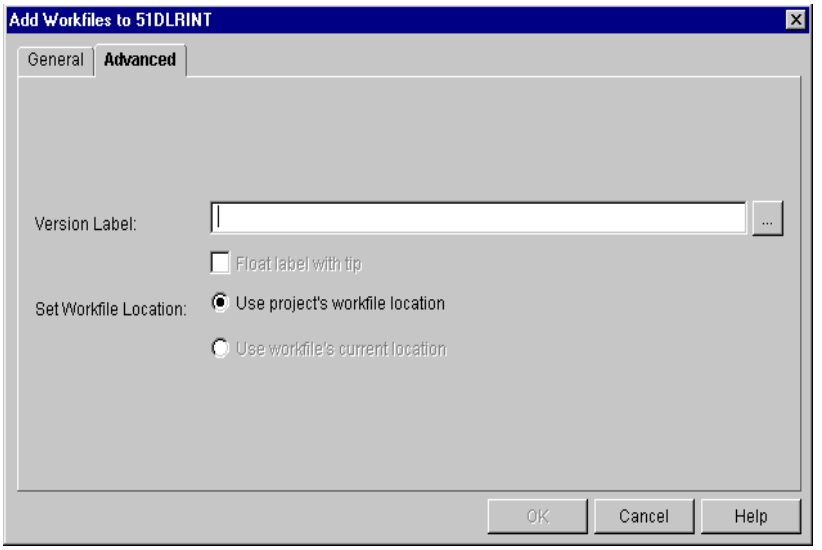
The default value is **[Default Promotion Group]**, which is a workspace setting that defines a lowest-level promotion group to use for this action. If a value for this workspace setting is **not** defined and you do not select a value, then no promotion group is associated with the revision.

h (Only available if you have Superuser or Unlimited privileges). Select the **Don't check in workfile** check box to add the workfile but not check in the file.

i (For Serena TrackerLink and SourceBridge users only). Click the **Associate Issues** button if you want to associate the workfiles you are adding with issues. This displays the association dialog box.

NOTE TrackerLink or SourceBridge is invoked automatically if you or your administrator has set it up to require that you associate issues when adding workfiles.

4 Under the Advanced tab, if you are checking in your workfiles, you can optionally do any of the following:



a In the **Version Label** field, assign a version label to the checked in revision by entering a version label or by clicking the Browse button to select an existing version label that is assigned to a versioned file. Note that version labels are case sensitive.

b Select the **Float label with tip** check box to keep the specified version label associated with the latest revision.

c Set the location for the workfile you are adding by choosing from the following options:

- **5.3/6.0 project:** If the **Use project's workfile location and copy workfiles into it** is your only option, the files' archives are copied to the

selected archive directory and the workfiles are copied to the workfile location set for your current workspace.

This option also requires you to choose what you want to do if a workfile already exists in the workfile location. The options include **Skip** or **Overwrite**. The default value is Skip.

NOTE The above option is only available if you are not adding files from the workfile location. If they are already in the workfile location, then you cannot reset either radio button and the labels reads **Use project's workfile location**.

- If the **Use workfile's current location** is your only option, the files' archives are added to the project but the workfiles remain in their current location.

  For information on how to control which option is available to you, see "5.3/6.0 project workfile location" on page 94.

- **Folder: Use project's workfile location and copy workfiles into it** is the default option. If this option is not available, the workfiles you selected are already located in the workfile location set for your current workspace.

  You can also use the **Use workfile's current location**, which means that the files' archives are added to the project but the workfiles remain in their current location.

5 Click **OK**.

Importing Archives

Importing archives allows you to access existing archives and use them in projects. While it provides a function similar to copying versioned files and projects, importing archives provides other functions that copying does not.

Importing archives provides a method of:

- **Accessing archives created in a command-line interface only environment.** The desktop and web clients require archives to be associated with projects and versioned files. The command-line does not create projects or versioned files; it works directly with the archives. By importing command-line created archives into the desktop client, the archives are associated with a project (or project database) and a versioned file is created. The versioned file provides continued access to the archives from the command-line interface, as well as access from the desktop and web clients via projects.
- **Re-creating versioned files that have been deleted.** If you accidently delete a versioned file, you no longer have desktop client access to the versioned file's archive. However, you can re-create the versioned file by importing the archive back into the project from which it was deleted. This re-creates the versioned file and reestablishes the association between the versioned file and the archive.
- **Accessing all of the archives within an archive directory.** Desktop client projects don't necessarily display all of the archives within the selected archive directories (unless all of the workfiles were added to the directory or all of the archives within the archive location were imported). Copying/moving a desktop client project would skip the archives that are not displayed in the desktop client window. When you choose to *import* an archive directory, all of the archives are imported.

- **Restoring the reference between a versioned file and its archive when the archive is moved.** When the location of an archive is changed, you must import the archive to restore the link between the versioned file and the archive. For more information on how to restore the reference between a versioned file and its archive, see the *Serena PVCS Version Manager Administrator's Guide*.

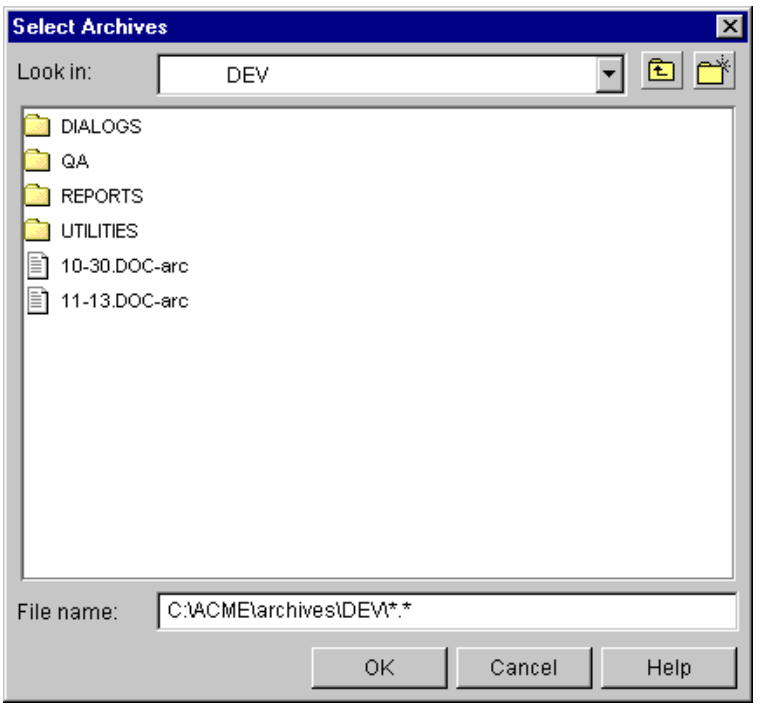
Importing Archives into Project Databases, Projects, and Subprojects

When importing archives into project databases, projects, and subprojects, you can import individual files, a single directory, or an entire directory tree. When you import archives, Version Manager creates a versioned file that references the new archive.

To import archives into project databases/projects:

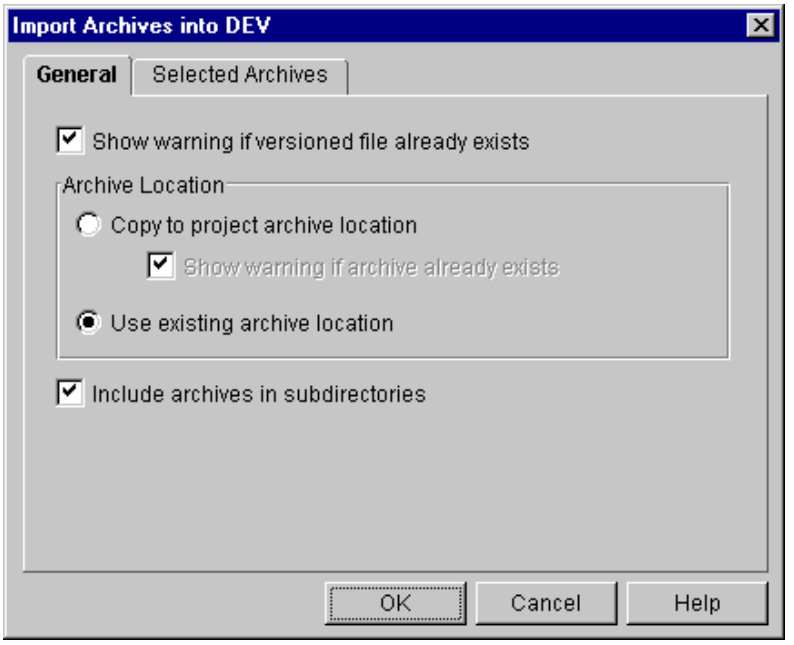
1 Select the project database or project to which you want to import archives.

2 Select Admin | Import Archives. The Select Archives dialog box appears.



The location displayed in the selection pane is the current archive location of the selected project or project database.

3 Select the archive or directory that you want to add and click **OK**. The Import Archives dialog box appears.



4 Do the following:

a By default, Version Manager displays a warning message if a versioned file already exists in the project you selected to import archives. If you do not want Version Manager to prompt you with a warning message, clear the **Show warning if versioned file already exists** check box.

b In the Archive Location group, select the archive location for the archives you are importing. **Use existing archive location** is the default option, which means you are referencing the archives but leaving the archives in their current location.

Your other option is **Copy to project archive location**, which means you want to copy the archives to the project's archive location and use this location as the new archive location.

By default, Version Manager displays a warning message if an archive you are importing already exists in the specified archive location. If you do not want Version Manager to display a warning message, clear the **Show warning if archive already exists** check box.

c If you are adding a directory of archives, an **Include archives in subdirectories** check box is available. Select this check box if you want Version Manager to import the archives located in subdirectories.

5 Click **OK**.

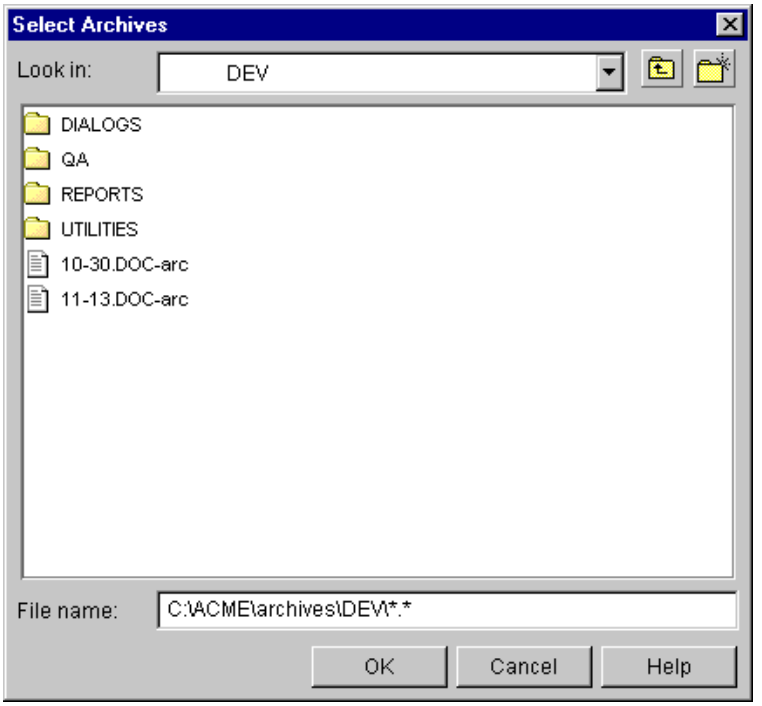
Importing Archives into 5.3/6.0 Projects

You can import archives into a 5.3/6.0 project or folder. You cannot import archives into 5.3/6.0 project roots. You can add individual archives or a directory of archives. When importing a directory of archives, Version Manager only imports the archives located at the root level of the directory; it does not import archives located in subdirectories. When you import archives, Version Manager automatically creates a versioned file that references the new archive.

To import archives into 5.3/6.0 projects:

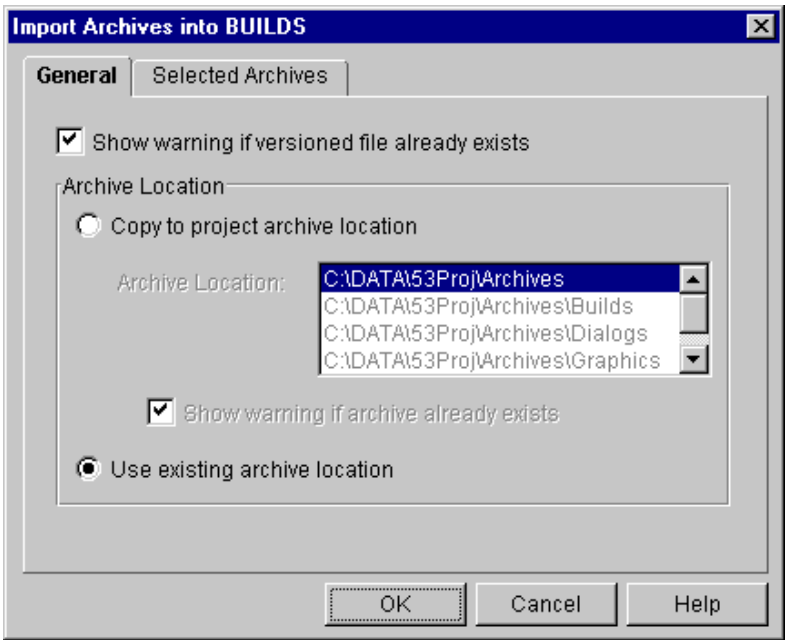
1 Select the 5.3/6.0 project or folder to which you want to import archives.

2 Select Admin | Import Archives. The Select Archives dialog box appears.



The location displayed in the selection pane is the current archive location of the selected project or project database.

3 Select the archive or directory that you want to add and click **OK**. The Import Archives dialog box appears.



4 Do the following:

   a By default, Version Manager displays a warning message if a versioned file already exists in the project you selected to import archives. If you do not want Version Manager to display a warning message, clear the **Show warning if versioned file already exists** check box.

   b In the Archive Location group, select the archive location for the archives you are adding. If you are adding archives to a:

   - **5.3/6.0 project:** Your only option is **Copy to project archive location**. This option copies the archives to the project's archive location. To specify the archive location, select the location in the **Archive Location** list.

     By default, Version Manager displays a warning message if an archive you are importing already exists in the specified archive location. If you do not want Version Manager to display a warning message, clear the **Show warning if archive already exists** check box.

   - **Folder: Use existing archive location** is the default option, which means you are referencing the archives but leaving the archives in their current location. You can also select the **Copy to project archive location** option.

5 Click **OK**.

Scenario: Creating Projects that Mimic the Existing Workfile Structure

Now Jacob must create two new projects, Checkers and Solitaire. Workfile structures exist for both of these projects and Jacob wants the project structures to mimic their organization.

Jacob selects the Games project database and selects the Add Workfiles option. He browses to the top project directory, `k:\checkers`, where he sees the client and server subdirectories, and clicks **OK**. He then completes the following Add Workfiles options:

- Enters "Original source files for CheckersMaster 2.0" as a brief description for the set of related files. This information is stored in the archive associated with each file. Jacob can later review this information by selecting a versioned file and viewing its properties.
- Maintains the **Use existing workfile location** default to indicate that he wants to store workfiles in the same directory as the original project files—for example, the workfile locations would be *`k:\checkers`* for the main project files, `k:\checkers\client` for the Client subproject files, etc. Version Manager stores workfile locations in workspace settings for the Games project database. (In later scenarios Jacob and the Project Team Members will define workfile locations that reflect the specific needs of individual projects and users.) Once Jacob completes this task, he can verify the workfile locations by selecting the Checkers project and choosing the Set Workfile Location option. He can repeat this verification step for each of the subprojects as well.
- Maintains the Check In defaults, which, after check in, leaves unlocked versions of the workfiles in the workfile location.

- Verifies that the **Include workfiles in subdirectories** check box is selected so that Version Manager will add each of the subdirectories and their contents under the Checkers project. In Version Manager, these subdirectories translate to subprojects of the Checkers project.

Jacob selects OK to confirm his selections. Version Manager performs the following tasks:

- Adds the Checkers project to the Games project database.
- Creates the Client subproject and its Board and Images subprojects.
- Creates the Server subproject and its Library subproject.
- Adds versioned files to the project structure for every source file in the checkers directory and its subdirectories. Each versioned file now has a single revision, the initial revision.
- Assigns the default initial description, "Initial Revision" to the first revision of each versioned file.
- Creates an archive for each versioned file.
- Assigns the existing locations as the workfile locations for the Checkers project and its subprojects.
- Assigns the description "Original source files for CheckersMaster 2.0" to each archive.

Jacob repeats the above process for the Solitaire project. Once he adds the Solitaire project, Jacob displays the Revision pane to review and verify the basic information about the initial revisions of the newly archived files.

Now that Jacob has successfully added all four projects, he can complete the project setup process by configuring the Images subproject so that it is shared between the Chess and Checkers projects. To do so, Jacob selects the Images subproject from Chess and selects the Copy Project option. He selects Checkers as the destination project and accepts the system defaults for the copy action. The Chess and Checkers projects now share access to a single Images project. For each image, Version Manager will maintain a single archive and versioned file. The project setup task is complete.

Chapter 6

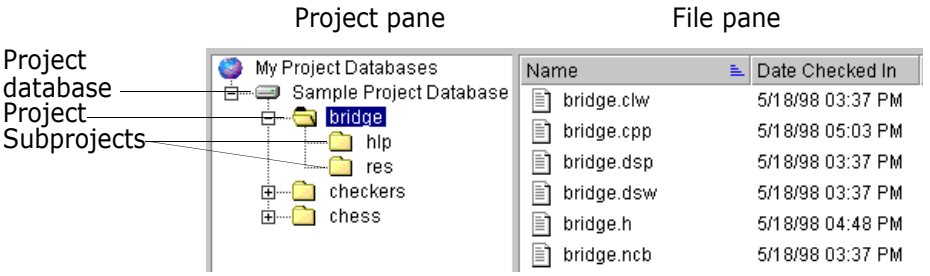
Working with Projects

About Projects

What is a project?

A Version Manager project is like a system directory and its subdirectories. You can create projects in Version Manager to hold sets of interrelated files. Like system directories and subdirectories, projects can also be hierarchical, meaning subprojects can exist under projects and subprojects.

Similar to a system drive, all Version Manager projects must be contained within a Version Manager project database.



Why create projects?

Create projects to help you organize your files. If your workfiles are currently organized in a logical manner, it makes sense to create a Version Manager project that mimics this structure.

However, you can also create a Version Manager project that is different from the existing file structure and makes more sense to your project environment.

Creating Projects

Before you begin...

To create a project, you must open a project database. If you do not have access to a project database, contact your Administrator.

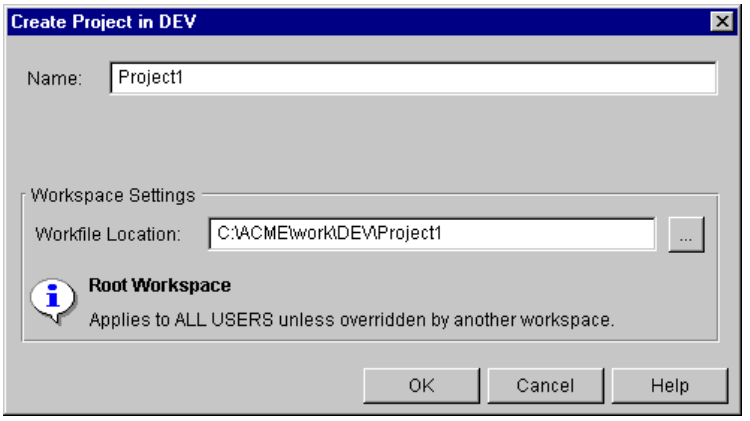
Existing files

If you already have existing files in an organized directory structure, you do not have to create a project. You can simply add the directories and files to the project database, and Version Manager will automatically create the project(s) for you. For information on how to add directories and files directly to Version Manager, see "Adding Workfiles" on page 89.

To create a project:

1 Select the project database you want to use to store the project.

2 Select File | Create Project. The Create Project dialog box appears.



3 Specify a unique name for this project in the **Name** field. No two projects at the same level in the project database can have the same name. The name cannot begin or end with a tab or blank space. Any character can be used in the name except:

- Asterisk (*)
- Colon (:)
- Vertical bar (|)
- Slashes (/ \)
- Question mark (?)
- Angle brackets (< >)
- Tilde (~)
- Percent (%)
- Ampersand (&)
- Double quote (")
- Single quote (')
- Comma (,)

Also note that no project name can be the two character name of .. or the one character name of . or @.

Notice that the project name you enter is also reflected in the **Workfile Location** path.

4 A new workfile location that matches the name of the project you entered is specified in the **Workfile Location** field. If you want to specify a different workfile location, enter it in this field or click the Browse button to select a location.

NOTE When you select a workfile location, we recommend that you specify a local drive to which all authorized users of this project have access. By doing this, you ensure that the default workfile location is one that all Project Team Members can access. Remember that Project Team Members can always change the workfile location by creating a private workspace that specifies a different workfile location.

5 Click OK. If the workfile location you specified does not exist, a Create Directory confirmation message appears. Click Yes to confirm the creation of a new workfile location.

Creating Subprojects

A subproject (also called a nested project) is the same as a project, except that it is stored directly beneath a project within a project database. To create a subproject, follow the same steps as "Creating Projects" on page 104, except select a project instead of a project database in step 1.

Renaming Projects

You can rename a project whenever the project name is no longer appropriate. When you rename a project, you are only changing the project's name in the desktop client; the name of the project directory on the operating system is not changed.

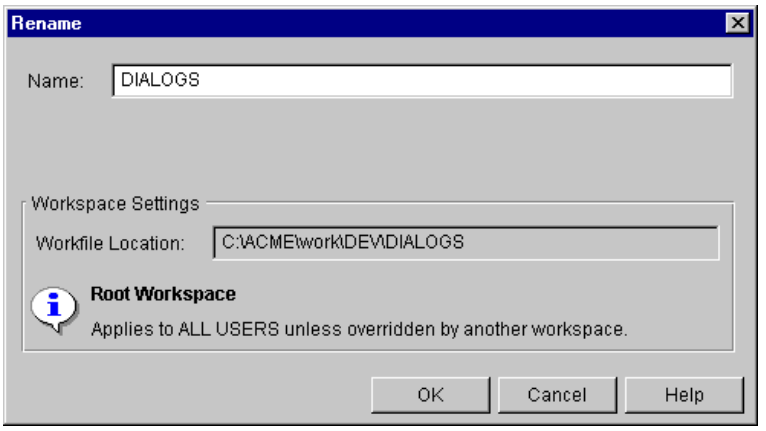
When you rename a project, if the project's original workfile location has not been overridden, a new workfile directory is created to match the new project name. All files are checked in from, and checked out to, the new workfile location. However, if the project's original workfile location has been overridden, the name of the project changes but the workfile location remains the same.

If someone has versioned files checked out while you are changing the name of the project, Version Manager retains the old workfile location. When the file is checked back in, the User will be prompted to select either the old or new workfile location.

To rename a project:

1 Select the project (or subproject) that you want to rename.

2 Select File | Rename. The Rename dialog box appears.



3 Enter the new project name in the **Name** field. Notice that the new project name you enter is also reflected in the noneditable **Workfile Location** path.

4 Click OK. If the workfile location does not exist, a **Create Directory** confirmation message appears. Click Yes to confirm the creation of a new workfile location.

Deleting Projects

You can delete a project whenever the project is no longer necessary. When you delete a project, the project and any subprojects are deleted from the Project pane, and the versioned files are deleted from the File pane. The archives remain in the archive directory unless you have privileges to delete archives.

To delete a project:

1. Select the project (or subproject) you want to delete.
2. Select File | Delete. The Confirm Item Deletion message appears.
3. Click Yes to delete the project.

NOTE You cannot restore a deleted project, but you can recreate a deleted project. To do this, create a new project and import the deleted project's archives.

Scenario: Creating a Subproject and Adding Workfiles

While reviewing the project database setup, Jacob notices that the Solitaire project is missing code written by a new contractor. He needs to incorporate the game_rules code written by the contractor into the Solitaire project.

Jacob wants to store the contractor's work in a separate subproject named Rules. To do so, he selects the Solitaire project and chooses the Create Project option. Jacob names the project Rules and sets the workfile location to `c:\work`. By modifying the workfile location, he allows the contractor to easily move files from the network to his local drive. If the contractor does not currently have a work directory, Version Manager will create it the first time he checks out files. Jacob selects OK. The new subproject displays beneath the Solitaire project. Jacob selects the Add Workfiles option and navigates to the network location of the game_rules files. He selects each of the files, and then selects OK. The files are now incorporated into the Solitaire project.

Chapter 7

Using Workspaces

About Workspaces

What is a workspace?

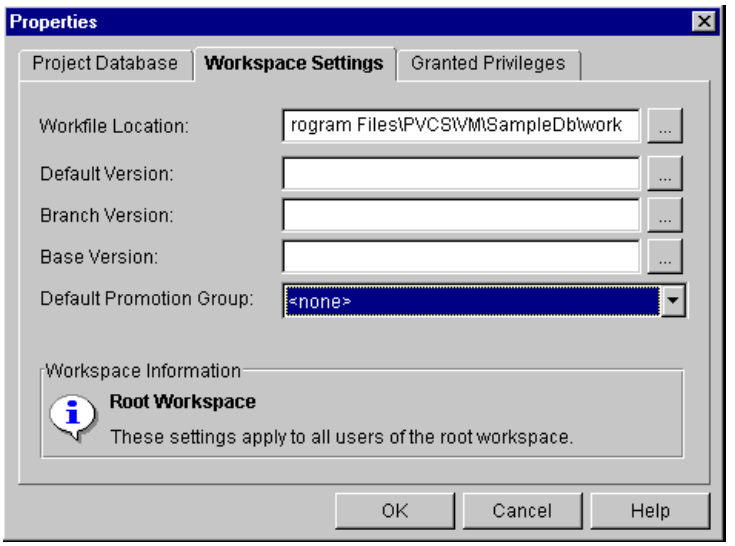
A workspace is a collection of work settings defined for a project database and all of the projects and versioned files contained within the project database. Although you can create and choose from multiple workspaces, you can set only one workspace on a project database at any given time.

A workspace stores these work settings:

- Workfile Location—the directory from which you check files in and out of Version Manager. A workfile location is required, and it is defined when a project database and project are created or when workfiles are added.
- Default Version—the version label that defines the revision that will be acted on if no other version label or revision number is defined. If blank, the latest revision is the default version.
- Base Version—used for automatic branching, this is the version label that you assigned to mark the revision from which you want to start a branch. If blank, automatic branching is not enabled.
- Branch Version—used for automatic branching, this is the version label that you assigned to the tip of the branch. If blank, automatic branching is not enabled.
- Default Promotion Group—the default promotion group, which is valid only if a promotion model is in effect. The default promotion group is a lowest-level promotion group of a promotion model.

  By default, Version Manager associates the default promotion group you specify with revisions when checking out revisions, locking revisions, and adding workfiles. By defining a default promotion group, you eliminate the need for a user to specify which lowest-level promotion group to use for these actions.

You can access these work settings by selecting a project database, project, or versioned file and then selecting File | Properties | Workspace Settings tab.



Why use workspaces?

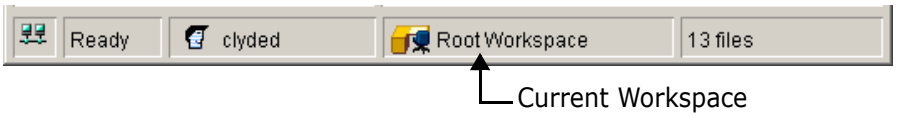
Workspaces allow you to easily set and modify work settings at a project database, project, or versioned file level. Workspaces also allow you to fine tune the work settings of a project to fit individual working styles. For example, if multiple Project Team Members want to work on project files locally, each team member can set up a private workspace

that has a workfile location set for their own workstation. You can also set different default, base, and branch versions for each project to allow for different branching configurations.

Although you can create and choose from multiple workspaces, only one workspace can be active on a project database at any given time.

Workfile locations

The most common work setting stored in a workspace is the *workfile location*. A workfile location is the directory from which you check out files and to which you check in files. Workfile locations are defined when you create a project database, create a project, or add a file. When you modify a workfile location, the change is saved to the *current* workspace. Your current workspace is displayed in the status bar at the bottom of the Version Manager desktop client window.



To quickly access the workfile location work setting, select a project database, project, or versioned file and then select File | Set Workfile Location.

Public vs. Private Workspaces

There are two types of workspaces: public and private.

- *Public workspaces* are typically created by the Administrator when a project database is created. A Project Leader may want to modify or create new public workspaces, depending on the needs of a project. The workfile locations for public workspaces are typically set for locations on network servers that are accessible to all Project Team Members.

  Your ability to change the settings of a public workspace is dependent upon the privileges assigned to you by your Administrator. However, changes made to a public workspace affect everyone who accesses the project database using that workspace. For this reason, using a combination of public and private workspaces is highly recommended when multiple users are accessing the same project database.

- *Private workspaces* are typically created by individual Project Team Members so that they can customize the project work settings without affecting other Project Team Members. Private workspaces can only be seen and accessed by the person who created them. The workfile locations for private workspaces are typically set for locations on an individual's local drive.

About Root Workspaces

Default public workspace

The Root Workspace is the default public workspace. A Root Workspace is created automatically every time your Administrator creates a project database. You cannot delete or rename the Root Workspace. Your ability to change the settings of the Root Workspace is dependent upon the privileges assigned to you by your Administrator.

If you do not create or set any other workspaces, the Root Workspace is set as the active workspace.

The settings that are initially defined in the Root Workspace include:

- The workfile location your Administrator set for the project database when the database was created.
- The default revision to use for actions if the master configuration file associated with the project database has the default revision defined. The initial setting is taken from the master configuration file, which in turn defaults to the tip revision.
- Automatic branching if the master configuration file associated with the project database has automatic branching set up. The initial setting is taken from the master configuration file.

One setting that is not initially defined is the default promotion group, which is valid only if a promotion model is in effect. The default promotion group is used if your Administrator has defined more than one lowest-level promotion group.

Workspace Hierarchies

Derived from the Root Workspace

All workspaces, whether public or private, are derived from the Root Workspace, which is at the top of the hierarchy. Every workspace you create inherits the work settings defined in the workspace from which it was derived. However, any changes you make to a work setting override any inherited work settings.

For example, in the hierarchy above, the Administrator created separate DEV, DOC, and QA public workspaces. The Project Leader needed a separate area for the Development Team to build the product, so a new public workspace was created called Build Area. A member of the Project Team wanted to check files out from the new Build Area, but wanted to work locally, so a private workspace called My DEV Area was created.

You can create public workspaces from other public workspaces. You can create private workspaces from either public or private workspaces. You cannot, however, create a public workspace from a private workspace.

Inheriting Workspace Settings

Workspaces are inherited

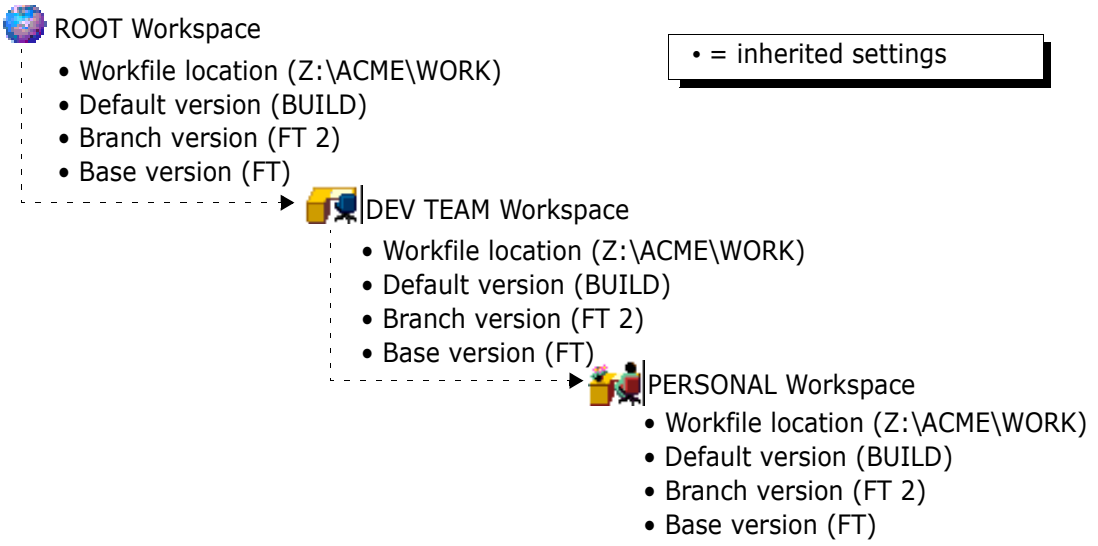
There are five shared settings between a parent and inherited workspace: workfile location, default version, branch version, base version, and default promotion group.

If you change a workspace setting in a parent workspace, it will also change in the inherited workspace unless the workspace setting in the inherited workspace has been modified. To re-inherit the setting of a parent workspace, delete the value in the inherited workspace setting field.

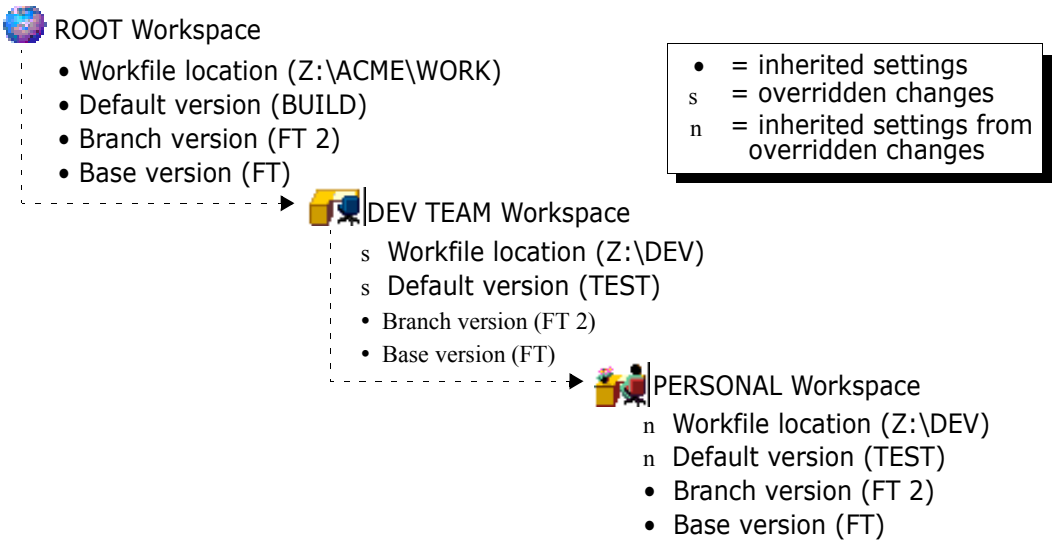
Example 1

The Administrator assigns values to the ROOT Workspace. The Project Leader creates a workspace from the root workspace called DEV TEAM. A Project Team Member creates a

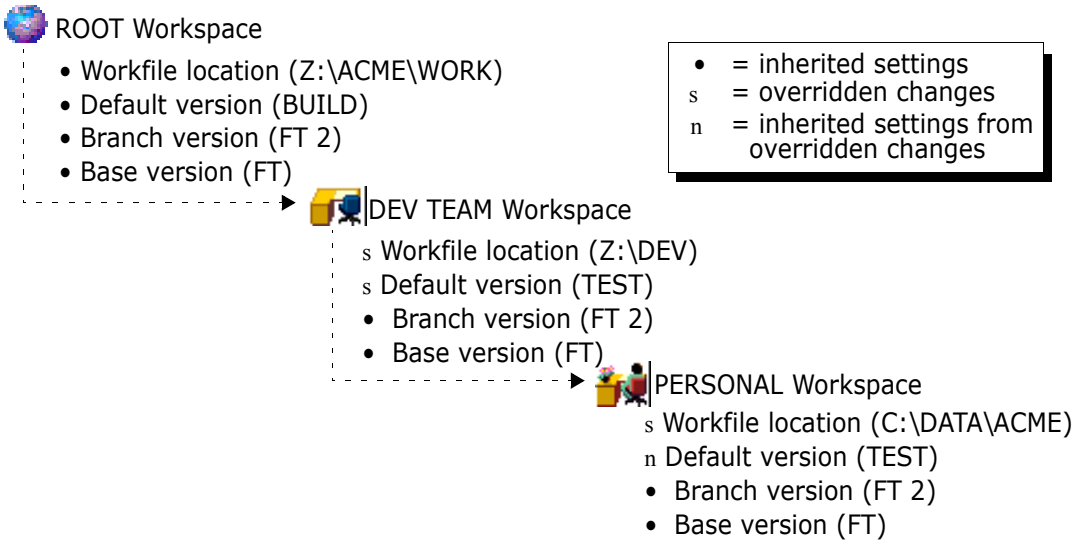
workspace from the DEV workspace called PERSONAL. All three workspaces have identical workspace settings.



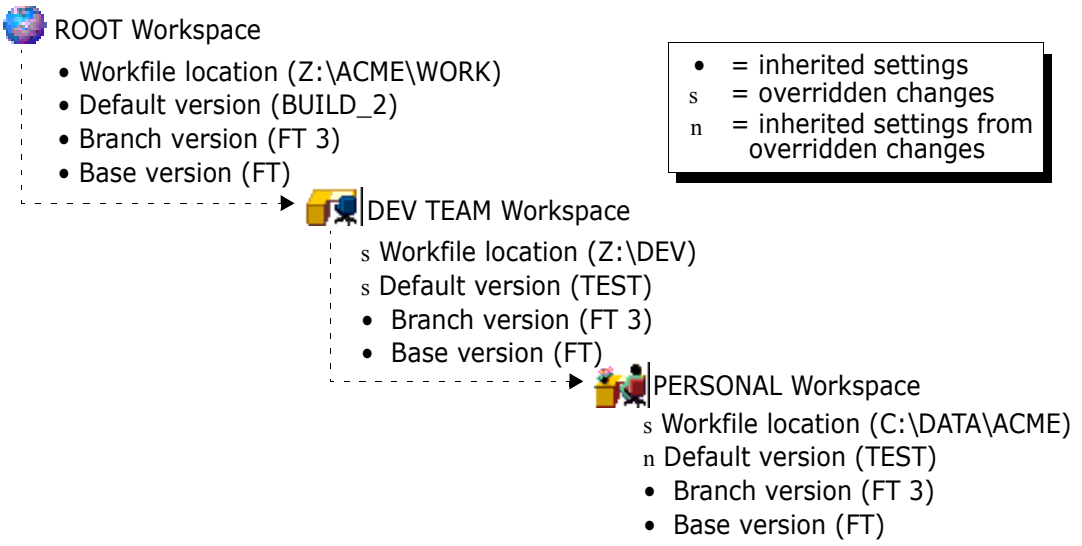
Example 2 The Project Leader changes two workspace settings in the DEV TEAM Workspace. Because the PERSONAL Workspace inherited its settings from the DEV TEAM workspace, the same two workspace settings change in the PERSONAL workspace.



Example 3 The Project Team Member changes the workfile location for the PERSONAL workspace to work locally.

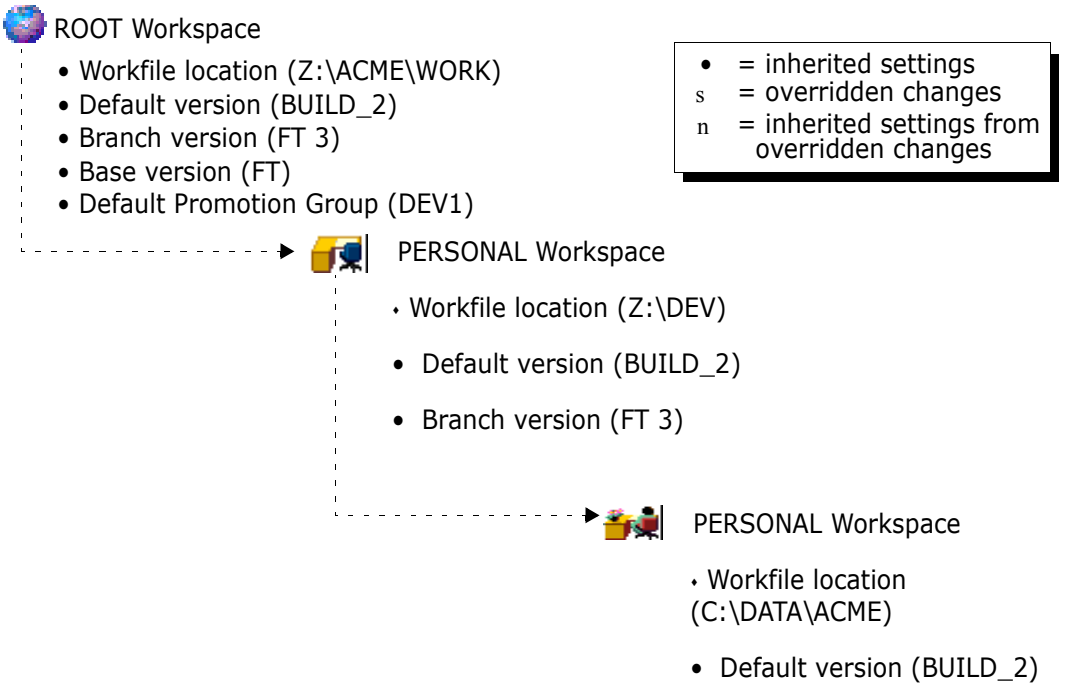


Example 4 The Administrator changes the default version and branch version workspace settings in the ROOT workspace. The branch version changes throughout both the DEV TEAM and PERSONAL workspaces; however, because the default version was overridden in the DEV TEAM workspace, the change in the ROOT workspace does not affect the DEV TEAM workspace or the inherited settings in the PERSONAL workspace.



Example 5 The Project Leader wants to re-inherit the default version setting between the DEV TEAM and ROOT workspace. To do this, the Project Leader deletes the default version value in the DEV TEAM workspace, causing the DEV TEAM workspace to re-inherit the value from its parent (ROOT) workspace. Because the default version value in the PERSONAL workspace setting was not overridden, the default version value in the ROOT workspace is

also picked up in the PERSONAL workspace. In addition, the Project Leader changes the workspace settings in the ROOT Workspace so that a default promotion group is defined.



Calculating Workfile Locations

When you create a project database, Version Manager requires you to enter an absolute path (a path with a drive specification) for its workfile location.

Project Database	Workfile Location
ACME (Z:\ACME)	Z:\ACME\WORK

By default, when you create a project, Version Manager creates a workfile location for the project by using the new project's name and appending it to the end of its parent's workfile location.

Project Hierarchy	Workfile Location
ACME (Z:\ACME)	Z:\ACME\WORK
DEV	Z:\ACME\WORK\DEV
DIALOGS	Z:\ACME\WORK\DEV\DIALOGS

How Version Manager Stores Workfile Locations

Version Manager only stores the part of the workfile location that is different from its parent; however, when you view the properties of a project/project database, the workfile location always appears as an absolute path because Version Manager calculates the complete workfile location for you. Version Manager stores one workfile location for each project and project database in every workspace.

NOTE If you delete the workfile location for a project while working in the root workspace, the workfile location for that project is set to the root project database directory.

Project Hierarchy	Workfile Location: What you see...	Workfile Location: What is stored...
ACME (Z:\ACME)	Z:\ACME\WORK	Z:\ACME\WORK
DEV	Z:\ACME\WORK\DEV	DEV
DIALOGS	Z:\ACME\WORK\DEV\DIALOGS	DIALOGS

Example 1

In the following example, the Administrator created the ACME project database, the DEV project, and the DIALOGS subproject and accepted the default workfile locations. The Root Workspace contains three workfile locations: one workfile location is absolute (`Z:\ACME\WORK`) and the remaining two workfile locations are relative (DEV and DIALOGS).

	Workfile Locations		
	Project Database	**Project**	**Subproject**
Workspaces	ACME (Z:\ACME)	DEV	DIALOGS
Root Workspace	Z:\ACME\WORK	DEV	DIALOGS

Example 2

Using the previous project hierarchy, this example shows how Version Manager calculates workfile locations with two workspaces.

The DEV TEAM workspace was created by the Project Leader to give the Developers a different workfile location for the DEV project to test their workfiles. By only specifying a workfile location for the DEV project, Version Manager must calculate the workfile locations for the ACME project database and DIALOGS subproject in the DEV TEAM workspace.

	Workfile Locations		
	Project Database	**Project**	**Subproject**
Workspaces	ACME (Z:\ACME)	DEV	DIALOGS
Root Workspace	Z:\ACME\WORK	DEV	DIALOGS
DEV TEAM	**1**	Y:\TEST	**2**

1 The DEV TEAM workspace workfile location for the ACME project database is `Z:\ACME\WORK`.

Because no workfile location was specified in the DEV TEAM workspace for the ACME project database, Version Manager looks at its parent workspace (Root Workspace) for a workfile location value (`Z:\ACME\WORK`). Since this is an absolute value, Version Manager is finished calculating the workfile location.

2 The DEV TEAM workspace workfile location for the DIALOGS subproject is `Y:\TEST\DIALOGS`.

Because no workfile location was specified in the DEV TEAM workspace for the DIALOGS subproject, Version Manager looks at its parent workspace (Root Workspace) for a workfile location value (DIALOGS). Since the value is relative, Version Manager appends the DIALOGS value to the DEV TEAM workfile location for DIALOG's parent project, DEV (`Y:\TEST`), resulting in the workfile location of `Y:\TEST\DIALOGS`. Since this is an absolute value, Version Manager is finished calculating the workfile location.

Example 3

Using the previous project hierarchy, this example shows how Version Manager calculates workfile locations with three workspaces.

The PERSONAL (private) workspace was created by a Project Team Member to mimic the ACME project hierarchy locally. By only specifying a workfile location for the ACME project database, Version Manager must calculate the workfile locations for the DEV and DIALOGS project in the DEV TEAM workspace.

	Workfile Locations		
	Project Database	**Project**	**Subproject**
Workspaces	ACME (Z:\ACME)	DEV	DIALOGS
Root Workspace	Z:\ACME\WORK	DEV	DIALOGS
DEV TEAM	—	Y:\TEST	—
PERSONAL	C:\ACME\WORK	**3**	**4**

3 The PERSONAL workspace workfile location for the DEV project is `Y:\TEST`.

Because no workfile location was specified in the PERSONAL workspace for the DEV project, Version Manager looks at its parent workspace (DEV TEAM) for a workfile location value (`Y:\TEST`). Since this is an absolute value, Version Manager is finished calculating the workfile location.

4 The PERSONAL workspace workfile location for the DIALOGS subproject is `Y:\TEST\DIALOGS`.

Because no workfile location was specified in the PERSONAL workspace for the DIALOGS subproject, Version Manager looks at its parent workspace (DEV TEAM) for a workfile location value. Since there is no value, Version Manager looks at DEV TEAM's parent workspace (Root Workspace) and finds a value (DIALOGS).

Since the value is relative, Version Manager appends the DIALOGS value to the PERSONAL workspace workfile location for DIALOG's parent project, DEV. However, since there is no value, Version Manager looks at PERSONAL's parent workspace (DEV) and finds a value (`Y:\TEST`), resulting in the workfile location of `Y:\TEST\DIALOGS`. Since this is an absolute value, Version Manager is finished calculating the workfile location.

Did the PERSONAL workspace accomplish the Project Team Member's goal of working locally? Only if the versioned files were located at the project database level. The reason the workspace did not mimic the ACME project hierarchy locally was because the PERSONAL workspace was derived from the DEV TEAM workspace, which was customized with an absolute value in the middle of the project hierarchy (`Y:\TEST`).

To mimic the entire ACME project hierarchy locally, the Project Team Member must either:

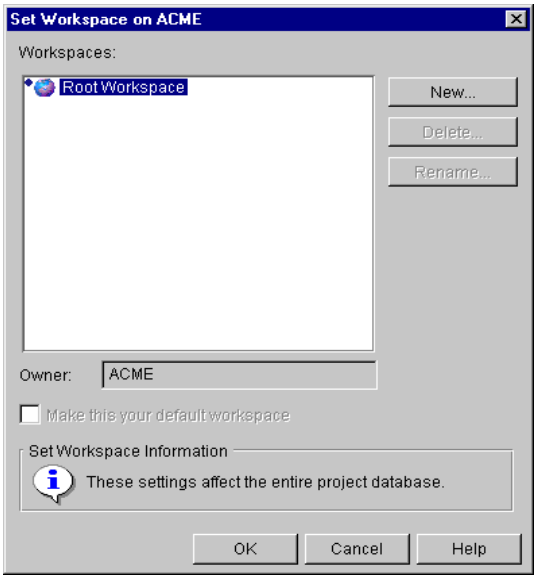
- Enter an absolute path in the PERSONAL workspace workfile location for the DEV project (`C:\ACME\WORK\DEV`), or
- Create a new private workspace that is derived directly from the Root Workspace with a local, absolute workfile location for the ACME project database.

Creating Workspaces

Remember! Only one workspace can be active on a project database at any given time, but each project in the project database can have its own work settings (workfile location, default version, base version, branch version, and default promotion group).

To create a workspace:

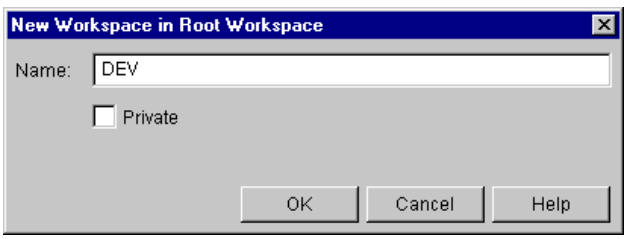
1. Select the project database for which you want to create a workspace.
2. Select File | Set Workspace. The Set Workspace dialog box appears.



3. Select the workspace from which the new workspace will inherit its initial settings. If this is the first workspace that you are creating, you must select the Root Workspace.

NOTE Remember that public workspaces cannot inherit from private workspaces. Private workspaces can inherit from public workspaces.

4. Click **New**. The New Workspace dialog box appears.



5 Enter the name of the new workspace in the **Name** field.

6 Select the **Private** check box if you want to make this a private workspace. If you do not select this check box, the workspace will be a public workspace.

7 Click **OK**. Version Manager creates the workspace and assigns it the work settings of the workspace from which it was derived. To edit these settings, refer to "Changing Workspace Settings" on page 120. The new workspace is also set as your current workspace.

Setting a Workspace

Your active workspace is always displayed in the workspace area of the status bar. Setting a workspace allows you to:

- Change from one workspace to another workspace
- Set a default workspace

What is a default workspace?

A default workspace is the workspace that Version Manager selects as your active workspace when you open a project database or restart Version Manager.

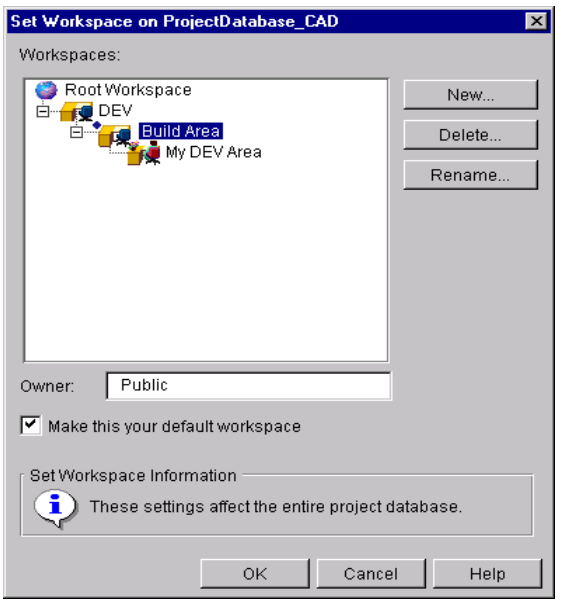
Every project database has one default workspace for each user. Until you set a default workspace, Version Manager uses the Root Workspace as your default workspace. The default workspace is illustrated by a blue diamond next to the workspace icon ( ).

To set a workspace:

1 Select the project database for which you want to set a different workspace.

2 Select File | Set Workspace. The Set Workspace dialog box appears.

TIP You can click the workspace area of the status bar to open the Set Workspace dialog box.



3 Select the workspace that you want to make active for this project database.

4 To make this workspace the default workspace, select the **Make this your default workspace** check box.

5 Click **OK**. The work settings on the project database switch to those defined in the workspace you selected.

Changing Workspace Settings

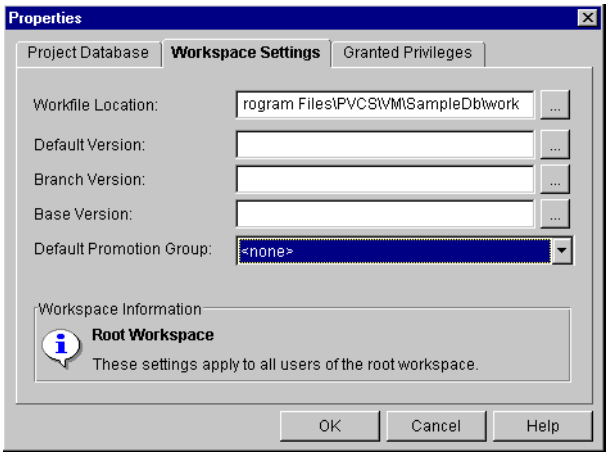
When you first create a workspace, the workspace inherits its work settings from the workspace from which it was derived. Any changes you make to the work settings override the settings that the workspace inherited.

To change workspace settings:

1 Set the workspace you want to edit. For more information, see "Setting a Workspace" on page 119.

2 Select a project database, project, or versioned file.

3 Select File | Properties and select the Workspace Settings tab.



These fields contain the values it inherited from its parent workspace (unless they have already been overridden by you or another user).

4 Enter or select the appropriate information in the following fields.

In this field...	Enter the...
Workfile Location	Location of your workfile directory. This is the directory from which you check in files and to which you check out files. If you delete the settings in this field, it will inherit settings from its parent workspace. If you delete the workfile location for a project while working in the root workspace, the workfile location for that project is set to the root project database directory. Workfile locations can contain `$HOME` at the root of their paths (for example, on UNIX, `$HOME/work` could expand to `/usrs/cherylc/work`). Version Manager substitutes the value of the HOME environment variable to compute the workfile location. The use of `$HOME` allows you to define a path that is automatically individualized for your users according to the value of their HOME environment variable.
Default Version	Version label you want to use as the default version when you perform actions such as check out, get, check in, lock, unlock, or assign version labels and promotion groups. If you delete the settings in this field, it will inherit settings from its parent workspace.
Base Version	Version label that you assigned to mark the revision from which you want to start a branch. This field is used to enable automatic branching. If you delete the settings in this field, it will inherit settings from its parent workspace. If the field is blank, automatic branching is not enabled.
Branch Version	Version label that you assigned to the tip of the branch. This field is used to enable automatic branching. If you delete the settings in this field, it will inherit settings from its parent workspace. If the field is blank, automatic branching is not enabled.
Default Promotion Group	The Default Promotion Group is valid only if a promotion model is in effect. The default promotion group is a lowest-level promotion group of a promotion model. By default, Version Manager associates the default promotion group you specify with revisions when checking out revisions, locking revisions, and adding workfiles. By defining a default promotion group, you eliminate the need for a user to specify which lowest-level promotion group to use for these actions.

5 Click **OK**.

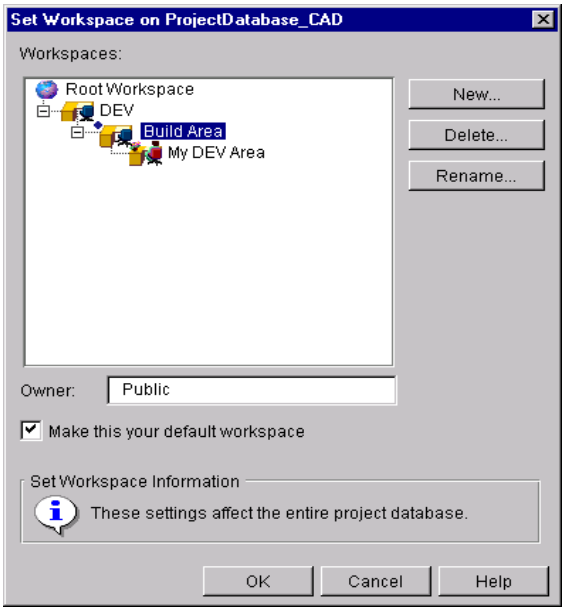
Renaming Workspaces

You can rename a public workspace if you have SuperUser or Unlimited privileges; however, if you rename a public workspace, the change will affect all users who access the workspace. If you created a private workspace, you may rename it.

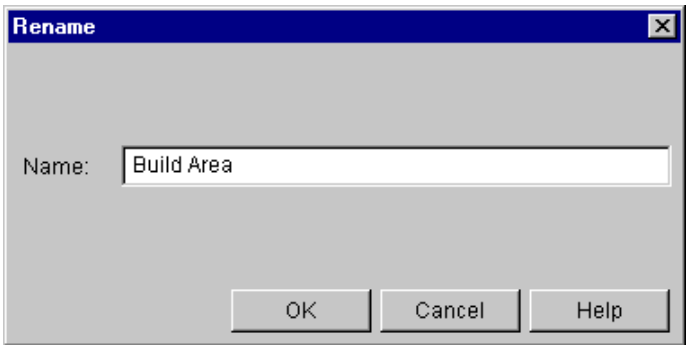
NOTE If you rename a workspace that is set to be a default workspace, the workspace is reset to the Root Workspace. This change affects all users who have the workspace that is renamed as the default workspace.

To rename a workspace:

1. Select the project database for which you want to rename a workspace.
2. Select File | Set Workspace. The Set Workspace dialog box appears.



3. Select the workspace that you want to rename and click **Rename**. The Rename dialog box appears.

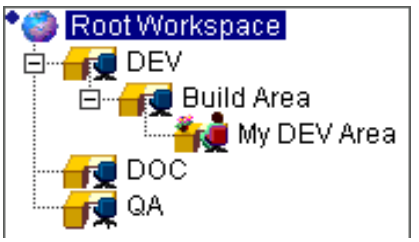


4. Change the name of the workspace and click **OK**. The new name is reflected in the Set Workspace dialog box.
5. Click **OK** to return to the Version Manager main window.

Deleting Workspaces

If you created a private workspace, and it is not your current workspace, you may delete it. If you have SuperUser or Unlimited privileges, you can also delete a public workspace. When you delete a workspace, all of the workspaces that were derived from it are also deleted. You cannot delete the Root Workspace, your current workspace, or a parent of your current workspace.

For example, in the hierarchy below, if you delete the DEV public workspace, you will also delete the Build Area public workspace and the My DEV Area private workspace.

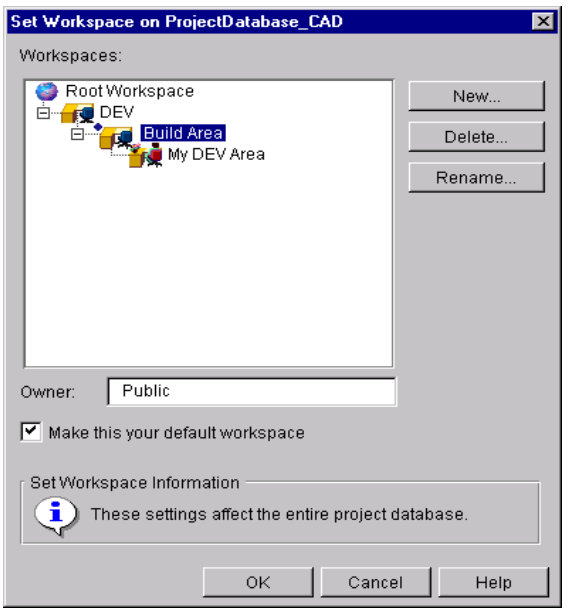


If another user deletes a public workspace that was set as your default workspace, the Root Workspace is set as your default workspace.

IMPORTANT! Deleting a public workspace affects everyone that uses that workspace.

To delete a workspace:

1. Select the project database for which you want to delete a workspace.
2. Select File | Set Workspace. The Set Workspace dialog box appears.



3. Select the workspace that you want to delete and click **Delete**. The Confirm Workspace Delete message appears.

4 Click **Yes** to delete the workspace. The workspace is removed from the workspace hierarchy.

5 Click **OK** to return to the Version Manager main window.

Scenario: Defining Customized Workspaces Without Affecting the Default Work Area

The Bridge project is close to complete. At this time, Quality Assurance Engineers are testing the code, Technical Writers are documenting the features, and Developers are completing their programming tasks. All team members work with the same set of archives on the same network drive but the different groups work in different locations on that drive. Also, the build team compiles the code on a completely different network drive. Jacob wants to set up a separate workspace for each group so that they do not need to redefine the workfile location each time they work with the files.

Jacob first reviews the workspace settings defined by the Administrator. To do so, he right-clicks the Games project database, selects the Properties option and clicks the Workspace Settings tab. The Administrator has configured the Root Workspace, which cannot be deleted or renamed, with the following default settings:

- For Workfile Location, the default setting is `h:\vm\games\work`. When you work with the Root Workspace, versioned files are checked out to a directory that is relative to the project database location. For example, while you are working in the Root workspace, the Checkers versioned files are checked out to `h:\vm\games\work\checkers`.
- For Default Version, the default setting is blank. Version Manager automatically operates on the latest revision if no default version is specified. Ruby has left this field blank. Later, if Jacob decides to set up automatic branching and merging, he will specify a default version that matches that of the branch version.
- For Branch Version, the default setting is blank. This version label identifies the latest, or tip, revision of a development branch. Because branching is not yet implemented in this database, Ruby has retained the blank default.
- For Base Version, the default setting is blank. This version label identifies the revision from which Version Manager begins an alternate, or branch, version of a file. Because branching is not yet implemented in this database, Ruby has retained the blank default.

Because all of the development groups store their workfiles in subdirectories of the workfile location defined for the project database, Jacob wants to specify workfile locations that are specific to the different development groups but relative to that of the project database. To do so, he first creates four new workspaces so that they inherit the Root Workspace default settings specified by Ruby. To accomplish this task, Jacob selects the Games project and selects the Set Workspace option. He selects the Root Workspace and then clicks the **New** button. He names the new workspace Quality Assurance. Because he wants the workspace to be *public*, or accessible by all users, he verifies that the Private check box is not selected before clicking **OK**.

Making sure to reselect the Root Workspace each time, Jacob repeats this process for the other three new workspaces—Writers, Developers, and Build. Once Jacob adds the last workspace, he reviews the workspaces hierarchy to make sure that all of the workspaces he created were displayed as children of the parent Root Workspace. This hierarchical

structure indicates that the four new workspaces inherit settings directly from the Root Workspace. However, he now needs to specify unique workfile locations for the new public workspaces.

To customize the workfile location for the Quality Assurance workspace, Jacob clicks the current workspace in the third quadrant of the status bar and sets the Quality Assurance workspace as the current workspace. He then selects the Games project database and selects the File | Set Workfile Location option. The workfile location for Quality Assurance is currently set as `h:\vm\games\work`, which the workspace inherited from the Root Workspace. Jacob appends `\QA` to the end of the path to define the department-specific workfile location. Note that the new workfile location now displays in the header above the File pane. Jacob repeats the above process for the Development and Writer workspaces. For the Build workspace, because team members do not build the code on the h: drive, Jacob must completely override the workfile location. Therefore, he customizes the Build workspace by entering `z:\build` as the workfile location.

Jacob is now ready to distribute Version Manager to the development team.

Scenario: Defining Personal Workspaces

Lydia is a software developer working on the Chess and Checkers projects. She uses the information Jacob gave her to install the product on her machine and log in to the Games project database. Before Lydia starts using the product, she wants to first customize her work environment. Jacob has given her a checklist of user options she can customize.

She starts by reviewing the workfile locations for the Chess and Checkers projects. As she suspected, both of these locations are on a network drive. However, Lydia works at home three days a week and would prefer to check out files to a location on her laptop. Reviewing Jacob's checklist she sees that the best way to modify these settings is to define a workspace that stores her private workfile settings. By creating a private workspace, Lydia can modify her own workfile locations without affecting the settings of other users.

To create the private workspace, Lydia selects the Games project database, and chooses the Set Workspace option. She highlights the Root Workspace and clicks the **New** button. She names the workspace after herself, LydiaP, and selects the Private check box. This option verifies that the changes will only affect her work location. Lydia selects the **Make this your default workspace** check box; Version Manager stores this setting with Lydia's user information. From now on, when Lydia opens the Games project database, Version Manager automatically sets LydiaP as the current workspace.

With LydiaP as the current workspace, she selects the Chess project, chooses the Set Workfile Location option, and changes the workfile location to `c:\work`. She repeats this step for the Checkers project.

Chapter 8

Specifying User Settings

Setting a Workfile Location

The workfile location is the location where you check in and check out workfiles. The user who creates projects (usually the Project Leader) determines the workfile location when the project is created. The workfile location is stored in the workspace assigned to the project database, which typically is a public workspace.

Public vs. Private Workspaces

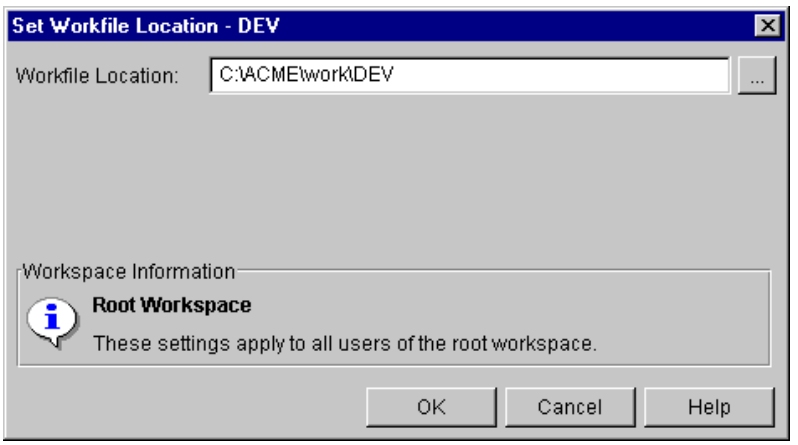
If you change the workfile location while in a public workspace, then you will change the workfile location for all users accessing the project using that public workspace. If you change the workfile location while in a private workspace, then the workfile location will not affect other users.

If you delete the workfile location for a project while working in the root workspace, the workfile location for that project is set to the root project database directory.

For more information regarding workspaces, see Chapter 7, "Using Workspaces" on page 109.

To set the workfile location for a project database, project, subproject, or versioned file:

1. Select the project database, project, subproject, or versioned file.
2. Select File | Set Workfile Location or File | Properties | Workspace Settings tab.



3. Enter the new workfile location in the **Workfile Location** field, or click the Browse button to select a new location.
4. Click **OK**.

Enabling an Application Log

You have the option of creating a `pvcsapp.log` file that records warning and error messages within Version Manager. Only errors that result in a command failure are written to this file. In addition, unexpected conditions, such as OutOfMemoryError, result in a stack trace in this file. Errors that are written to the command results log do not cause the command to stop processing and are not written to the `pvcsapp.log` file.

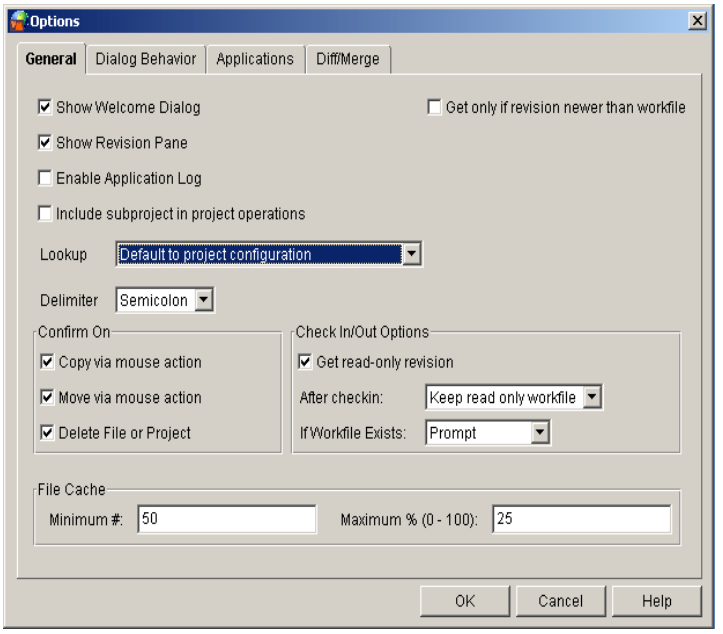
The Application Log option is specific to each workstation that runs Version Manager and can be enabled or disabled by each user. When you enable the Application Log, Version

Manager creates a file called `pvcsapp.log` in your *home* directory. To view the log, open the file with your default editor.

Windows Users: Your *home* directory is determined by the *home* environment variable set in your system properties. If you do not have a *home* environment variable set, Version Manager will place the log file in *drive*`:\users\default` where *drive* is the location of your Windows system files.

To enable the Application Log:

1 Select View | Options. The Options dialog appears with the General tab active.



2 Select the **Enable Application Log** check box and click **OK**.

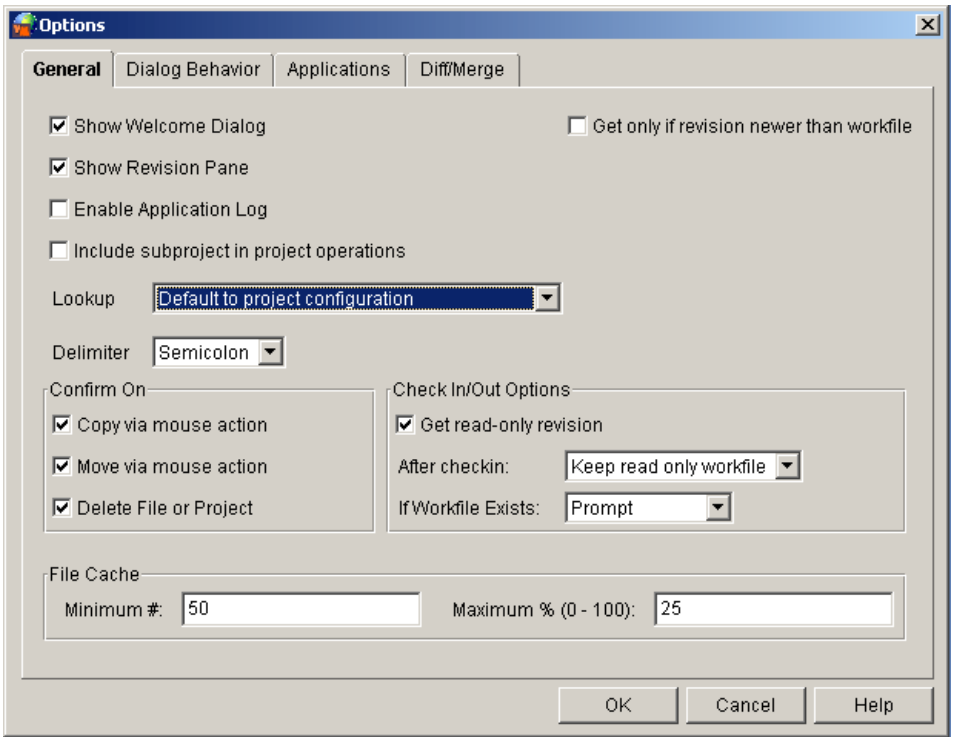
Disabling the Welcome Dialog Box

The Welcome to Serena Version Manager dialog box is designed to help beginning users with creating new project databases, opening the sample project database, and opening existing project databases.

The Welcome to Serena Version Manager dialog box appears each time you start Version Manager unless you select the **Don't show this dialog again** check box or disable the welcome dialog box option in your user settings.

To disable the Welcome to Serena Version Manager dialog box in your user settings:

1 Select View | Options. The Options dialog box appears with the General tab active.



2 Clear the **Show Welcome Dialog** check box.

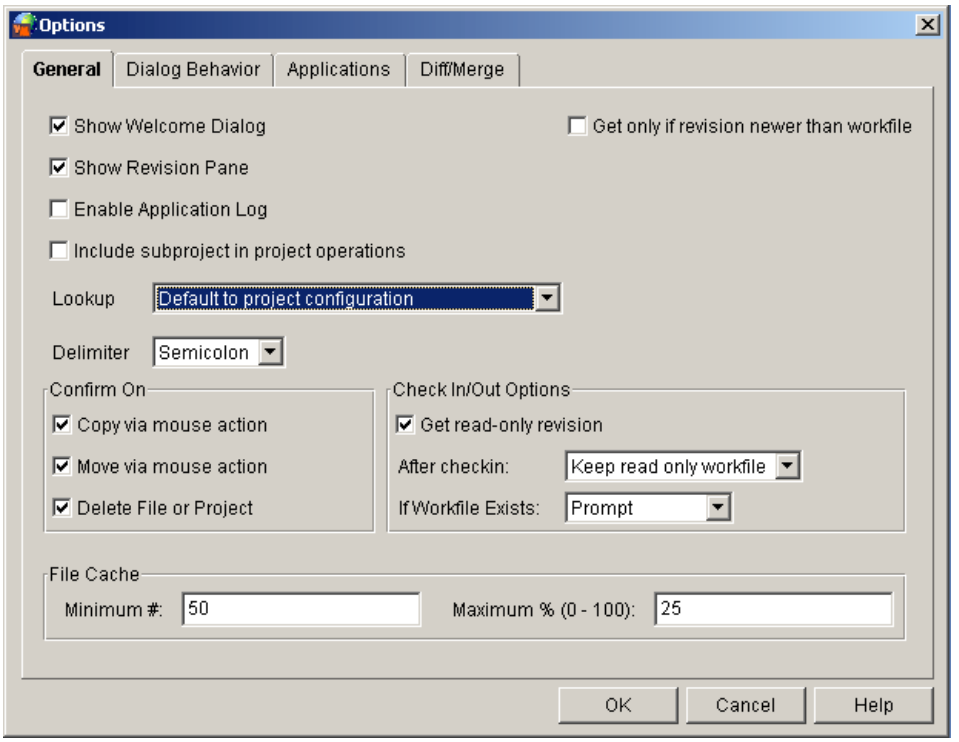
3 Click **OK**.

Disabling Confirmation Dialog Boxes

By default, a confirmation dialog box appears whenever you copy, move, or delete items from within Version Manager. All of the confirmation dialog boxes are turned on by default; however, you can turn many of the dialog boxes off (or back on) depending on the type of task you are doing.

To enable/disable any or all of the confirmation dialog boxes:

1 Select View | Options. The Options dialog box appears with the General tab active.



2 Select any of the options in the Confirm On group:

- To keep a confirmation dialog box from appearing when you copy items using drag and drop, clear the **Copy via mouse action** check box.
- To keep a confirmation dialog box from appearing when you move items using drag and drop, clear the **Move via mouse action** check box.
- To keep a confirmation dialog box from appearing when you delete versioned files or projects, clear the **Delete File or Project** check box.

3 Click **OK**.

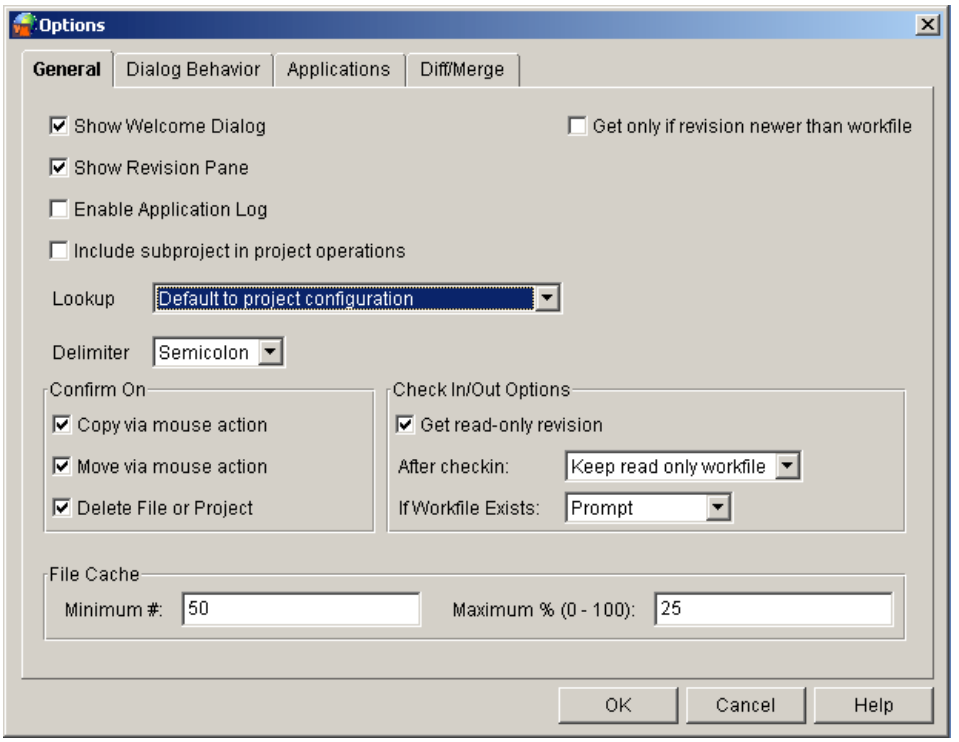
Including Subprojects in Project Operations

When you perform actions on project databases and projects, by default, Version Manager only performs actions on the versioned items contained directly at the project database/project level. Version Manager does not include versioned files in subprojects as part of its default operation.

If you find that you usually include subprojects when you perform actions on project databases and projects, Version Manager enables you to change the default from not including, to including subproject options during project operations.

To change the include subprojects default options when performing project database/project operations:

1 Select View | Options. The Options dialog box appears with the General tab active.



2 Select the **Include subproject in project operations** check box.

3 Click **OK**.

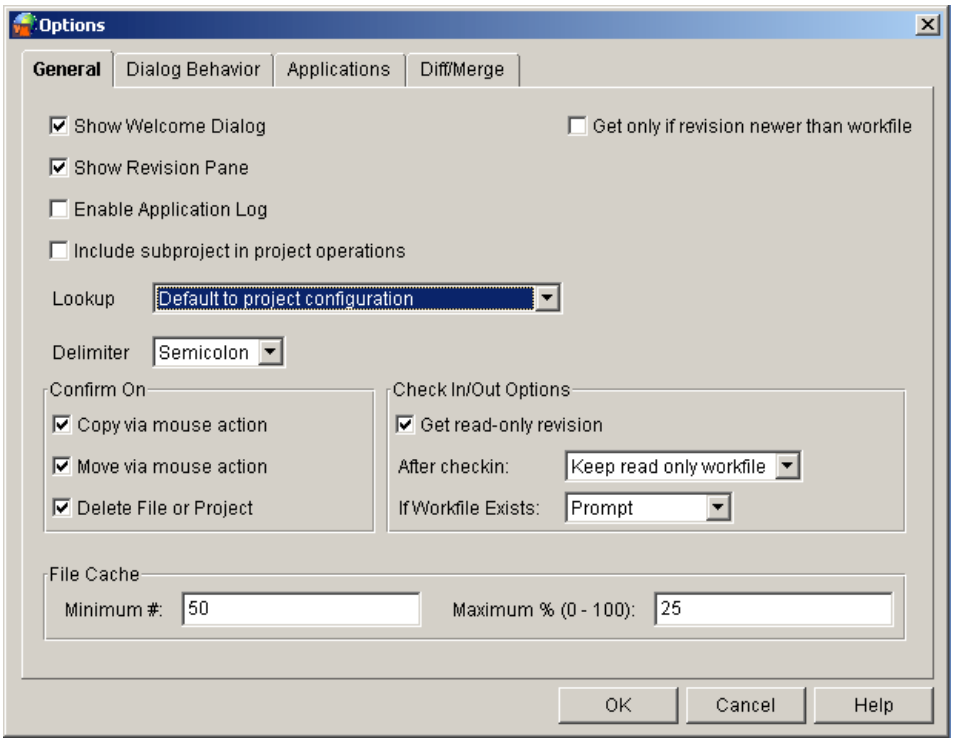
NOTE Selecting the **Include subproject in project operations** check box does not mean that you must include subprojects during project operations. It simply pre-selects the include subproject check box for those dialog boxes that allow project-level operations.

Defining Revision Lookup Behavior

Since Get operations ignore promotion groups unless you specify one in the **Promotion Group** field, a Get operation and a Checkout operation may not return the same revision. By defining the revision lookup behavior, you can cause a Checkout operation to behave like a Get. These settings are relevant only when a promotion model is in effect.

To configure the default revision lookup behavior for Get and Checkout operations:

1 Select View | Options. The Options dialog box appears with the General tab active.



2 Select one of the following from the **Lookup** drop-down list:

- **Classic lookup:**
  For **Get** operations, if the user does not specify a revision, version label, or promotion group to act on, the default revision, if one is defined in the configuration file, will be retrieved. Else the Tip revision on the Trunk will be retrieved.

  For **Checkout** operations, the revision defined by the lowest-level promotion group will be acted on. If such a revision is not found, the operation will climb the promotion model.

- **Default to project configuration:** The revision is determined based on the lookup option specified in the configuration file for each project or project database.

- **Lookup revision based on Revision:** The revision specified by the user will be acted on. If the user selects **[Default Revision]** in the **Revision** field, the revision specified by the workspace settings or configuration file will be acted on; if no default value is found, the Tip of the Trunk will be acted on.

- **Lookup revision based on Promotion group:**
  For **Get** operations, the revision assigned to the promotion group in the **Promotion Group** field of the Get dialog will be retrieved. If such a revision is not found, the operation will climb the promotion model and act on the revision assigned to the lowest currently assigned group in the promotion model.

  For **Checkout** operations, the revision assigned to the promotion group specified in the **Lowest-level promotion group** field will be acted on.

If the user selects **[Default Promotion Group]** in the **Lowest-level promotion group** field, the revision currently assigned to the promotion group specified by the workspace settings or configuration file will be acted on. If a default is not defined, the lowest-level group in the promotion model will be used. If there are multiple lowest-level groups, the user will be prompted to select one. If such a revision is not found, the operation will climb the promotion model and act on the revision assigned to the lowest currently assigned group in the promotion model.

3 Click **OK**.

Specifying a Delimiter for Items Entered in Fields

By default, semicolons (;) are used to separate multiple entries in dialog box fields. The delimiter character is used in all fields that accept multiple entries.

You need to change the delimiter character if you work with items that use it in their names. You can set the delimiter to be a comma (,), colon (:), or semicolon (;).

To specify a delimiter character:

1 Select View | Options. The Options dialog box appears with the General tab active.

2 From the **Delimiter** list, select one of the following:

- **Comma**
- **Colon**
- **Semicolon** (default)

IMPORTANT! Choose a delimiter character that is *not* present in the items you wish to enter into dialog box fields.

3 Click **OK**.

Defining Check In/Check Out Options

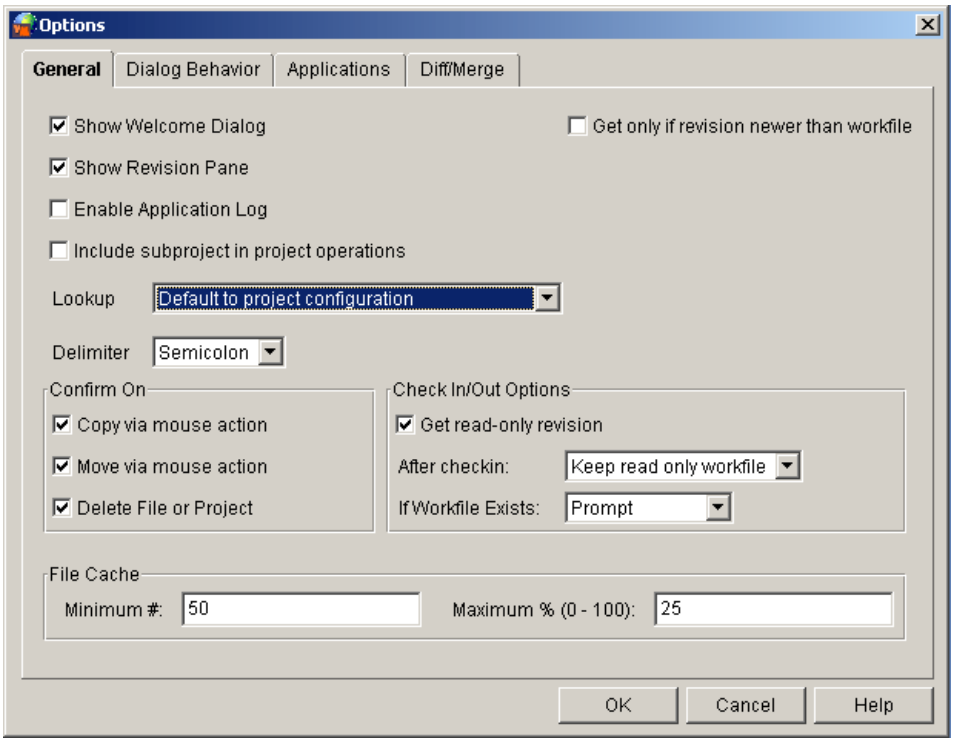
When you perform check in/check out/get actions, Version Manager defaults to the following options:

- Prompts you if a workfile already exists during a check out
- Leaves a read-only copy of the workfile in the workfile location after a check in
- Copies out a read-only workfile when you perform a get

If you find that you usually change these default options when you perform check in/check out/get actions, Version Manager enables you to change these default settings to fit your working style.

To change check in/check out/get options:

1 Select View | Options. The Options dialog box appears with the General tab active.



2 Select options from the Check In/Out Options group:

- **Get read-only revision:** To make workfiles writable when you perform a get, clear the this check box.
- **After checkin:** To determnine what to do with the workfile after checkin, select one of the following from the drop-down menu:
  - **Keep read only workfile**
  - **Delete writable workfile**
  - **Keep writable workfile**
- **If Workfile Exists:** To determine what to do when a workfile already exists, select an option from the drop-down menu:
  - **Prompt:** Asks you what to do.
  - **Overwrite:** Adds the workfile even if a duplicate workfile exists.
  - **Don't Overwrite:** Does not add the workfile.
- **Get only if revision is newer than workfile:** To get or check out a revision only if it is newer than the workfile, select the this check box. This saves time by not getting revisions that have not changed.

3 Select options from the File Cache group:

- Enter the minimum number of files that Version Manager will load into the File pane at one time in the **Minimum # field**. This is the minimum number loaded regardless of the percentage setting. If the number of files exceeds this number, the next minimum number of files will be loaded as required.

- Enter the maximum number of files to load at a time as a percentage of the total number of files in the **Maximum%** field.

  Version Manager compares the values in the two fields and uses the one that will display the largest number of files. For example, if you select 100 in the minimum field and a maximum percentage of 25% for a project that contains 2000 files, Version Manager will load 500 files (or 25% of 2000 files). Selecting a small minimum will load files for initial display faster, while a large maximum percentage will allow for faster scrolling.

  In general, the minimum number should be set to the average number of files in each project. The percentage can then be set for instances when the average is exceeded.

4 Click **OK**.

Defining Dialog Box Behavior

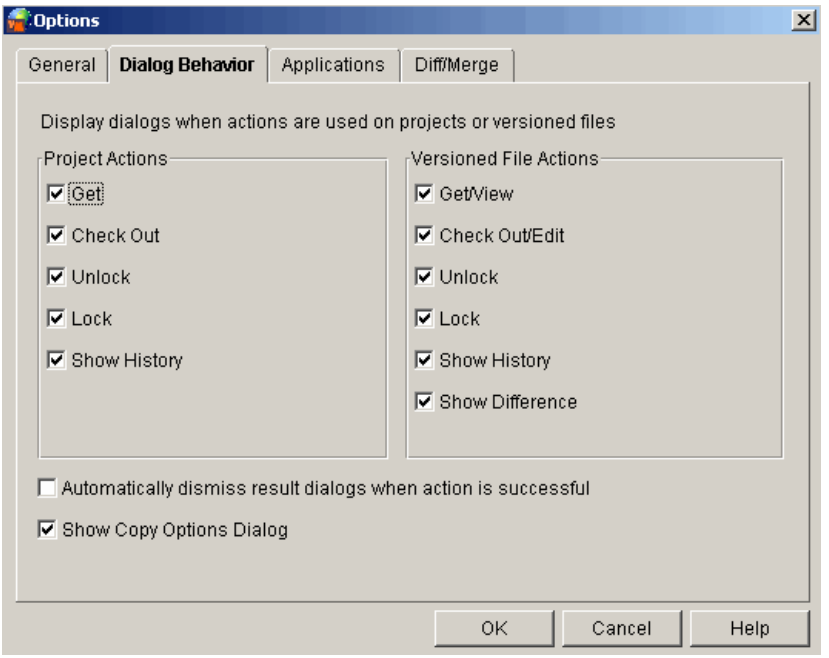
Version Manager allows you the flexibility of controlling dialog box behavior. By default, Version Manager displays a dialog box for all of the following project and versioned file actions:

- Get/View
- Check Out/Edit
- Unlock
- Lock
- Show History
- Show Difference (versioned files only)

NOTE The check in action for both projects and versioned files always invokes a dialog box.

To enable/disable any or all of these dialog boxes:

1 Select View | Options | Dialog Behavior tab.



2 In the Project Actions and Versioned File Actions groups, select any of the dialog boxes you want to disable.

3 If you want to dismiss the results dialog boxes when an action is performed successfully without having to click **OK**, select the **Automatically dismiss result dialogs when action is successful** check box.

4 If you want to hide the copy options dialog box when you copy items, clear the **Show Copy Options Dialog** check box.

5 Click **OK**.

Setting Your Default Editor

Setting your default editor specifies which editor Version Manager launches when you double-click a versioned file in Version Manager.

Windows On Windows, Version Manager opens the file in either the editor defined by the Windows file type associations or a specific editor you have selected. By default, Version Manager uses the Windows file type associations. We recommend that you use the Windows file type associations.

NOTE If the wrong application is launched when you double-click a file, correct the Windows file type associations using Windows Explorer (Tools | Folder Options | File Types tab).

UNIX On UNIX, you must define the location of:

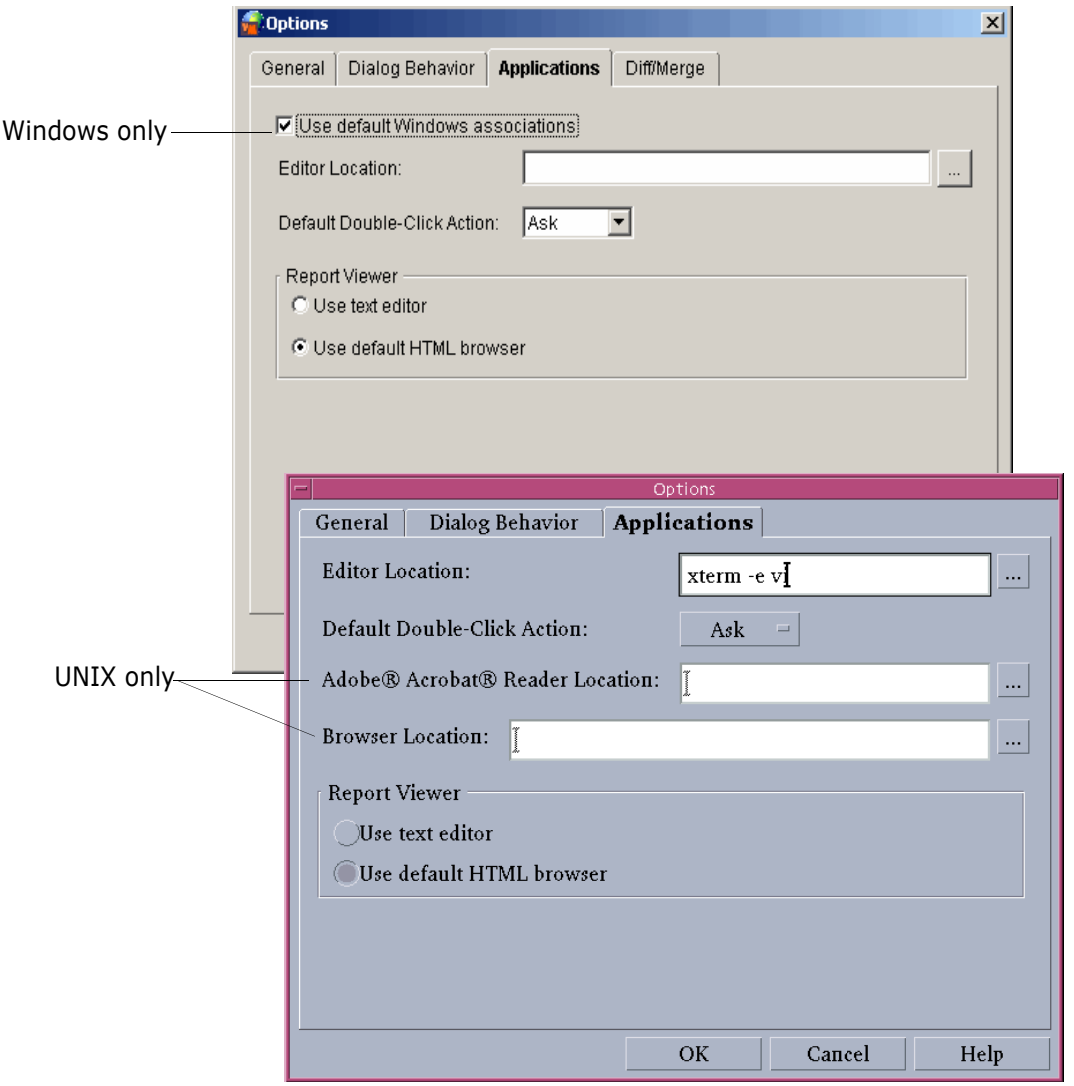
- An editor so you can launch files for editing or viewing from within Version Manager.

- Adobe Acrobat Reader so you can launch the online manuals by selecting Help | Online Manuals.
- An HTML browser so you can view the online help.

NOTE The **Use Default Windows Associations** option is not available to UNIX users.

To set your default editor:

1 Select View | Options | Applications tab.



2 (Windows only) If you want to use the Windows file type associations to launch an editor, select the **Use default Windows associations** check box.

3 In the **Editor Location** field, enter the location of the editor program executable, or click the Browse button to select it.

- (UNIX only) If you specify a non-GUI editor that launches its own window (such as vi), you will need to launch the editor in an xterm or other type of window. For example, in the Editor Location field, you would enter:

```
xterm editor path
```

where `editor path` is the location of your editor.

- (Windows only) If you specify an editor *and* select the **Use default Windows associations** check box, the editor you specified will be used only if Windows cannot find an association.

4 To select the action that you want Version Manager to perform when you double-click a file, select an option from the **Default Double-Click Action** drop-down menu. The options are:

- **Ask**
- **View file**
- **Edit file**

The default is **Ask**, meaning a dialog box appears asking you which action to take—either view the file or edit the file. If you select **View file**, Version Manager gets the file for viewing only. If you select **Edit file**, Version Manager checks out the file for editing.

5 (UNIX only) In the **Adobe® Acrobat ® Reader Location** field, enter the location of the Adobe Acrobat executable used for viewing the Version Manager online manuals, or click the Browse button to select it.

6 (UNIX only) In the **Browser Location** field, enter the location of the HTML browser executable, or click the Browse button to select it. The browser you specify will be used to display the online help; it may also be used to display reports.

7 In the Report Viewer group, select the viewer you want Version Manager to use to display reports:

- Select **Use text editor** if you want to display Version Manager reports in a text editor.
- Select **Use default HTML browser** if you want to display Version Manager reports in HTML format.

NOTE When you generate large reports (over 500 files), we recommend that you display the report in a text editor. Displaying large reports in an HTML editor can take up to a few minutes to display. For more report information, see Chapter 19, "Using Reports" on page 239.

8 Click **OK**.

Specifying a Difference or Merge Tool

By default, Version Manager uses the Serena PVCS merge tool when differencing or merging. However, you can configure Version Manager to use third-party difference and merge tools.

Users have the option to set the default tool in the **islv.ini** file (Windows) or the **.islvrc** file (Unix) instead of selecting it through the Options | Diff/Merge tab.

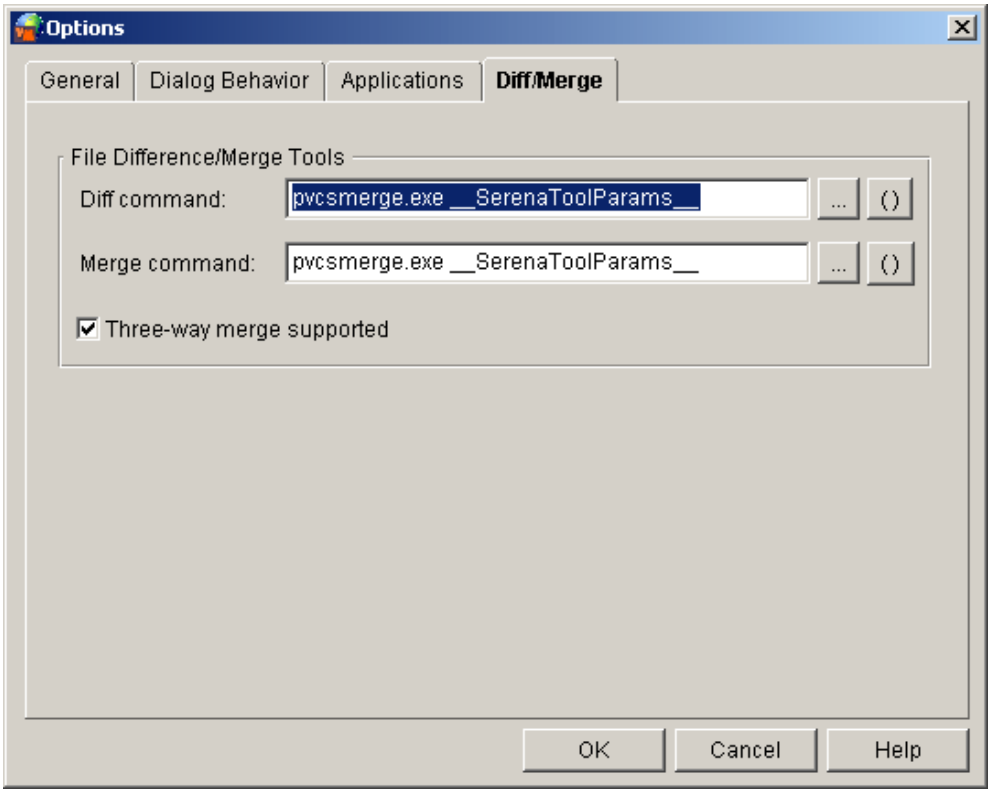
CAUTION! Users must be adept at making changes to the islv.ini file (Windows) or the .islvrc file (Unix) as any errors could severely affect Version Manager.

Tool Name	Sample islv.ini Configuration Information
Serena PVCS merge tool	`pvcs.diff.tool.cmd.win32=pvcsmerge.exe`
WinMerge	`pvcs.diff.tool.cmd.win32=D:\VM\Setup Files\WinMerge-2.8.0-exe\winmerge.exe /dl __AncestorLabel__ /dr __Derivative1Label__ __AncestorFile__ __Derivative1File__`
TkDiff	`pvcs.diff.tool.cmd.win32=c:/progra~1/tkdiff/tkdiff.exe __AncestorFile__ __Derivative1File__ -L __AncestorLabel__ -L __Derivative1Label__`

For the list of arguments, see Chapter 8, "Argument" on page 141 in this user guide.

To specify a difference or merge tool:

1 Select View | Options | Diff/Merge tab.



2 In the **Diff command** field and in the **Merge command** field, enter the location of the third party executable, or click the Browse button to select it.

The argument selection button () facilitates two types of selections:

- Predefined Diff / Merge tools

Above the separator bar are names of predefined Diff/Merge tools like WinMerge, TkDiff and Version Manager PVCS Diff Tool. Select these options to replace the value of the Diff/Merge command with a set of arguments that work with these tools.

NOTE Except for the PVCS Diff Tool, the arguments are provided as an example only and are not guaranteed to work on all versions of those tools.

NOTE An administrator can add, modify or delete the predefined Diff/Merge tools by changing the `diffmerge.properties` files. This file is stored in the directory `vm/common/pvcsprop/pvcs/vm`.

- Arguments

If you want to manually construct a Diff/Merge command, you can select values (arguments) from below the separator bar. This will provide access to values in the Diff/Merge dialog as selected at run time.

The following table is a list of difference and merge arguments that can be applied. The result column lists sample result values for each parameter:

Argument	Usage	Result
AncestorFile	Use this parameter to select the base file (the file from which the other files were derived).	`C:\DOCUME~1\Clyde~1\LOCALS~1\Temp\unuz781.tmp`
AncestorLabel	Use this parameter to pass a label that describes the ancestor file.	`Base: /Project1/islvtk.ini 1.1`
Derivative1/ _Derivative2_	Use these parameters to access the files associated with the Branch1 / Branch2 selections. **NOTE** Derivative2 is applicable only for the **Merge command** field and is used if the tool supports three way merge functionality.	`C:\DOCUME~1\Clyde~1\LOCALS~1\Temp\v6dvuw0.tmp`
Derivative1Label/ _Derivative2Label_	Use these parameters to pass labels that describe the Branch 1 / Branch 2 files (to be used as captions). **NOTE** Derivative2 Label is applicable only for the **Merge command** field.	`Branch 1: / Project1/ islvtk.ini 1.0`
IsMerge	This parameter will be replaced by *true* if the command is executed as part of a Merge operation, and *false* if it is executed as part of a Diff operation.	
TargetFile	Use this parameter to pass the output file path specified by the user in the **Show Merge** dialog.	`C:\DOCUME~1\Clyde~1\LOCALS~1\Temp`

3 Select the **Three-way merge supported** checkbox if the selected third party tool supports three way merging.

4 Click **Ok**.

For more information on using the diff/merge tool, see Chapter 16, "Comparing Files" on page 207 and Chapter 17, "Merging Files" on page 213 in this manual.

Passing Options Based on Ignore White Space

Version Manager has an **Ignore white space** checkbox on the **Show Differences** dialog. It is possible to create an optional argument for the Diff tool based on the state of this checkbox.

This can be done using the parameter __IfIgnoreWhitespace__.

The syntax for this parameter is:

```
__IfIgnoreWhitespace__DiffArgumentIfBoxIsSelected|DiffArgumentIfBoxIsNotSelected|
```

For example, if your Diff tool expects the argument `-b` to ignore white space, you would include:

```
__IfIgnoreWhitespace__-b||
```

in your list of arguments.

NOTE In the above example when a user selects **Ignore white space**, the option -b will get passed to the specified Diff command. When it is not selected, no argument is passed.

If your Diff tool enables white space handling using arguments like `--ignore-white-space=true` and `--ignore-white-space=false`, you can use:

```
__IfIgnoreWhitespace__-ignore-white-space=true|-ignore-white-space=false|
```

Scenario: Specifying Personal Work Settings

To finish defining her personal settings, Lydia selects View | Options. Version Manager stores these options separately for each user. Lydia specifies her personal settings as follows:

- **On the General tab:**
  - Lydia clears Show Welcome Dialog so the welcome dialog box does not appear each time she launches Version Manager.
  - Given that she typically works with the latest version of a file, she chooses to maintain the default of not viewing the Revision pane. She knows that she can display the Revision pane if she needs to review the history of a particular versioned file.
  - She selects the option for enabling the Application Log. The Application Log records warning and error messages. It is stored as a pvcsapp.log file and is viewable in any text editor. Hopefully, she will not need to view this information, but she wants to make sure that she can when necessary.
  - She selects the Include subproject in project operations option. Given that both the Chess and Checkers projects contain subprojects, she wants to ensure that all actions performed on these projects include the versioned files within the selected project as well as the versioned files stored within their subprojects.

  - She chooses to maintain the option of confirming both move and delete operations, but chooses not to confirm any of her copy operations. She does not modify any of the Check In/Out options at this time. She may choose to modify these settings after she has worked with the product for a short time.

- **On the Dialog Behavior tab**, because Lydia wants to familiarize herself with the behavior of the Version Manager environment, she retains the option of viewing the dialogs for all of the actions she will perform on projects and versioned files. She also retains the default of viewing the results dialog box, regardless of whether the action was successful. Once she is familiar with the product, she plans on setting the appropriate Get, Check Out, and Check In defaults by turning off these dialog boxes. When she does so, Version Manager will perform the action with the selected defaults without prompting her with an intermediate dialog box. At that point, she will also select the option for dismissing the results dialog when the action is successful.
- **On the Applications tab**, Lydia can set a default editor to open when she double-clicks the versioned file. The Applications tab allows Lydia to select an editor she can use when editing workfiles. She enters the path name of the editing program executable file in the Editor Location field. Now it is set up so when she double-clicks the versioned file, Version Manager opens it through the chosen editor.
- **On the Diff/Merge tab**:
  - Lydia can set a default difference and merge tool from the available list of diff/merge tools including the default PVCS difference and merge tool.
  - She customizes the display parameters like ancestor file (the base file from which the other files are derived), target file etc. and clicks the **Three-way merge supported** checkbox if the external tools support three-way merge. Based on this setup, she can view the difference and merge when she clicks the **Show difference** or **Show merge** submenu from the Actions menu.

Lydia stores her personal settings and is now ready to perform version control tasks.

Chapter 9

Checking Out Revisions

About Check Out

Why check out revisions?

Check out revisions when you want to make changes to them. You can check out revisions by selecting a versioned file, multiple versioned files, a project, or an entire project database. When you check out a project or project database, Serena PVCS Version Manager checks out all of the versioned files contained within the project or project database.

If you do not want to change a revision, use the Get option (see "Getting Revisions" on page 157).

What happens during check out?

When you check out a revision, Version Manager locks the revision and creates a writable workfile in the specified workfile location.

Specifying a revision

When you check out a revision, you can select the revision by revision number, version label, or promotion group.

Associating issues

If you have installed TrackerLink or SourceBridge, you can associate issues with revisions from the Check Out dialog box by clicking the **Associate Issues** button.

NOTE TrackerLink or SourceBridge is invoked automatically if you or your administrator have set it up to require that you associate issues when checking out revisions.

NOTE You can set which issue management integration will be used by the next invocation of the Version Manager Desktop Client. Any currently open client sessions will not be affected.

1. Launch the Serena Issue Management Integration utility (from the Serena folder of the Windows Start menu, select Version Manager | Issue Management Integration).
2. Select **TeamTrack SourceBridge** or **Tracker TrackerLink**.
3. Click the **OK** or **Launch Version Manager** button.

Default Check Out Options

Version Manager uses the following initial defaults when you check out a revision. These defaults can be overridden when you check out the revision; and in some cases, the defaults can be redefined in the Options dialog box (View | Options | General tab). By default, Version Manager:

- Checks out the default revision to the location currently set as the workfile location

NOTE To define the default revision for your project or project database, see "Defining the Default Version" on page 162.

- Checks out only the revisions of the selected project and does not include the revisions of the project's subprojects

- Prompts for confirmation before overwriting the workfile if a writable workfile exists in the current workfile location

NOTE Version Manager does not prompt for confirmation if a read-only workfile exists in the current workfile location.

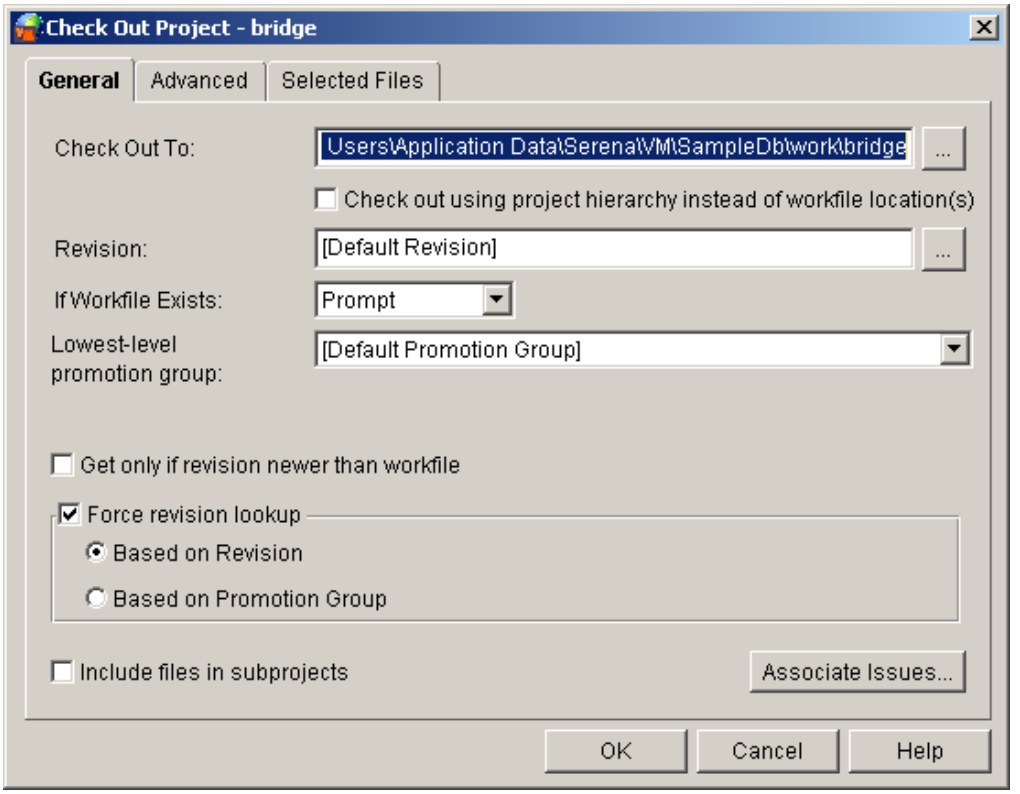
- Checks out a revision whether or not it is newer than the workfile.
- Retains the promotion group currently assigned to the revision, if a promotion model is in effect
- Does not automatically invoke TrackerLink or SourceBridge for associating issues when checking out files

Checking Out Revisions

To check out a revision:

1 Select the versioned file(s), project, or project database associated with the revision that you want to check out.

2 Select Actions | Check Out. The Check Out dialog box appears.



TIP If you typically check out the default revision, you can bypass the Check Out dialog box by clearing the **Check Out/Edit** check box under View | Options | Dialog Behavior tab. Note that if you are required to associate issues with checked out revisions, bypassing the Check Out dialog box does not affect the automatic display of the TrackerLink or SourceBridge association dialog box.

3 Do one of the following:

- To override the default check out options, continue to the next section.
- To accept the default check out options, click **OK**. Version Manager checks out the default revisions to their specified workfile locations and places a lock icon on the versioned file.

Overriding Default Check Out Options

1 Under the General tab of the Check Out dialog box, do any of the following:

- To change the workfile location of the revisions being checked out, edit the location in the **Check Out To** field, or click the Browse button to select the location.

  Editing this path changes the workfile location for this check out only. It does not permanently change the workfile location.

NOTE When you check out a single file, the **Check Out To** field displays the location and name of the file. You can change the name of the file, which creates a new workfile, and check it back in to create a new archive.

- (For checking out multiple versioned files, a project database, or project only). Select **Check out using project hierarchy instead of workfile location(s)** to override the default workfile locations and check out files to directories (locations) that mirror the project structure relative to the workfile location specified in the **Check Out To** field. If the directories do not exist, Version Manager creates them. Otherwise, Version Manager checks out files to the default workfile locations defined for each project/ subproject.

  For example, if the value in the **Check Out To** field is `c:\projdb\work` and this option is selected, Version Manager places the files as indicated in the following table:

If the project structure is:	Version Manager places the files in:
/proj1/subproj1	c:\projdb\work\proj1\subproj1
/proj1/subproj2	c:\projdb\work\proj1\subproj2
/proj2/subproj1	c:\projdb\work\proj2\subproj1

- To check out a revision other than the default revision, enter the revision number, version label, or promotion group assigned to the revision you want to check out in the **Revision** field, or click the Browse button to select it.

  If Version Manager can't find a revision associated with the promotion group you specify, it will check out the revision associated with the next highest group in the promotion model.

  To specify a value that begins with a number, you must precede it with a backslash (\). For example, `\1.2` or `\1abc`.

  To define the default revision for your project or project database, see "Defining the Default Version" on page 162.

- In the **If Workfile Exists** drop-down menu, select what you want Version Manager to do if the workfile you are checking out already exists in the selected workfile location. **Prompt** is the default option, which asks what you want to do when a duplicate workfile exists.

  The other options include **Overwrite**, which adds the workfile if a duplicate workfile exists, or **Don't Overwrite**, which does not add the workfile to the workfile location.

- To check out a revision only if it is newer than the workfile, select the **Get only if revision is newer than workfile** check box. This saves time by not checking out revisions that have not changed.

- **Lowest-level promotion group:** (Applies only if a promotion model is in effect.) If a promotion model is in effect, the lowest-level promotion group will be assigned to the revision you check out. If there are multiple lowest-level promotion groups, you can select one to apply from this drop-down list.

  If you select **[Default Promotion Group]** in the **Lowest-level promotion group** field, the promotion group specified by the workspace settings or configuration file will be used. If a default is not defined, the lowest-level group in

the promotion model will be used. If there are multiple lowest-level groups, you will be prompted to select one. l

NOTE If the **Include files in subprojects** check box is selected, the behavior described above applies to the promotion model in effect for each subproject.

- **Force revision lookup:** (Applies only if a promotion model is in effect.) If this check box is **not** selected and you do not specify a revision in the **Revision** field, the revision lookup behavior will be as described for the **Based on Promotion Group** option below.

NOTE To mirror the classic lookup behavoir of the Get dialog (**Force revision lookup** disabled on the Get dialog), select the **Based on Revision** option below.

To force different revision lookup logic, select one of the following options:

- **Based on Revision:** The revision specified in the **Revision** field will be acted on. You can enter/select a revision number, version label, or promotion group to specify the desired revision.

  If you select **[Default Revision]** in the **Revision** field, the revision specified by the workspace settings or configuration file will be acted on; if no default value is found, the Tip of the Trunk will be acted on.

  The promotion group specified in the **Lowest-level promotion group** field will be assigned to the revision.

- **Based on Promotion group:** The revision assigned to the promotion group specified in the **Lowest-level promotion group** field will be acted on.

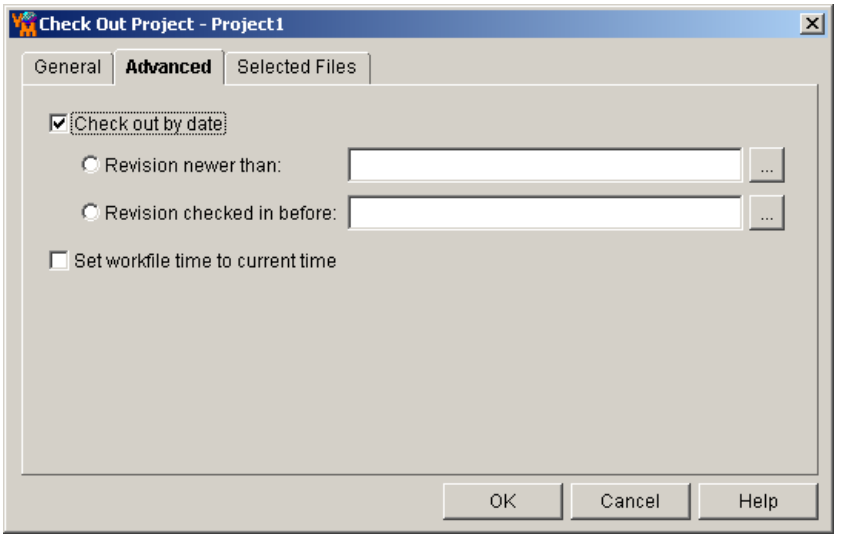
  If you select **[Default Promotion Group]** in the **Lowest-level promotion group** field, the revision currently assigned to the promotion group specified by the workspace settings or configuration file will be acted on. If a default is not defined, the lowest-level group in the promotion model will be used. If there are multiple lowest-level groups, you will be prompted to select one. If such a revision is not found, the operation will climb the promotion model and act on the revision assigned to the lowest currently assigned group in the promotion model.

- (For projects and project databases only). To check out revisions for versioned files located in subprojects, select the **Include files in subprojects** check box.
- (For Serena TrackerLink and SourceBridge users only). Click the **Associate Issues** button if you want to associate the workfiles you are checking out with issues. This displays the association dialog box.

  Note that TrackerLink or SourceBrdige is invoked automatically if you or your administrator have set it up to require that you associate issues when checking out files.

NOTE If you disable the Check Out dialog box to prevent it from displaying when performing a check-out operation, but you enable the association dialog box and require issue associations, Version Manager displays the association dialog box when you check out revisions.

2 Under the Advanced tab, do any of the following:

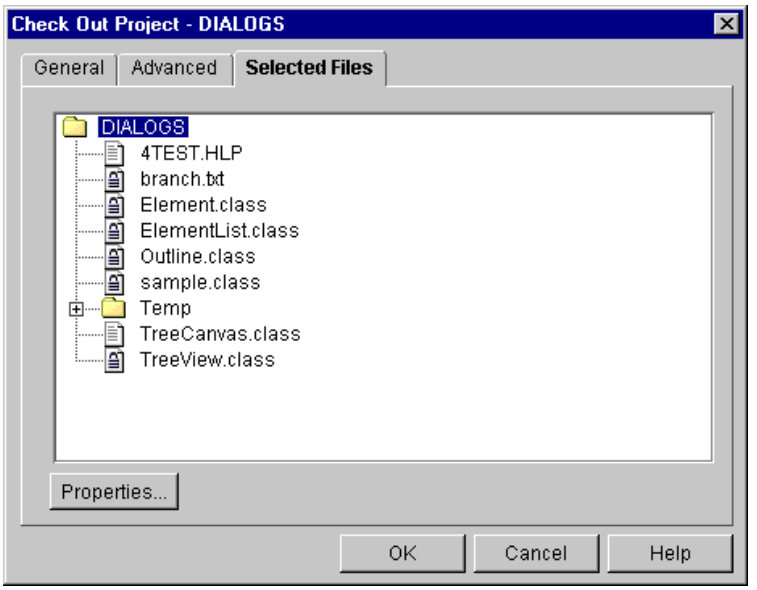


- To check out a revision by a specific date and time, select the **Check out by date** check box. By default, this option is not checked.

  The options include **Revision newer than**, which allows you to check out a revision based on a specific revision date, and **Revision checked in before**, which allows you to check out a revision based on a specific check in date.

- If you want to update the timestamp of the workfile to the current date and time, select the **Set workfile time to current time** check box.

3 Under the Selected Files tab, do any the following:



- To verify that you are checking out the correct files, review the files listed under this tab.

- To review project and workspace setting information of the selected items, click the **Properties** button. Click **OK** to return to the Selected Files tab.

4 Click **OK**. Writable copies of the selected revisions are checked out to their respective workfile locations and the revisions are locked.

Scenario: Checking Out and Editing Project Files

Lydia is now ready to start working on the set of Checkers files stored in the Server subproject. She selects the Server subproject and notices that the last quadrant on the status bar displays the total number of files in this subproject. She selects the Check Out option and reviews the default check out settings.

Because she wants to check out all of the files in the subproject as well as those in embedded subprojects, she selects the Include files in subproject check box. She verifies that her personal workfile location, `c:\work`, displays in the Check Out To field. (This location corresponds to the workfile location Lydia specified for her private workspace in an earlier scenario.) Lydia has been working on this project with three other Developers for several months. She is not sure if her working directory has existing workfiles in it or not, so she leaves the If Workfile Exists option set to Prompt. Version Manager will warn her if she checks out a file that corresponds to a writable workfile in her working directory.

Lydia then reviews the options on the Advanced tab. She does not want to check out files that are older than existing workfiles in her working directory, so she selects the Check out by date option for checking out only those revisions that are newer than the workfiles. She selects OK. Version Manager retrieves a writable copy of all the files contained in the Server project and its subproject, Library, and copies them to `c:\work`. Version Manager places a lock icon on the versioned files and the selected revisions.

Lydia has reviewed the Check Out options and set her personal defaults, such as to Prompt when overwriting, and she does not want to interact with the dialog the next time she checks out files. Therefore, she selects View | Options | Dialog Behavior tab and clears the Check Out option for both projects and versioned files. She also decides to modify the Default Double-Click Action from the Applications tab to Edit. From now on, when Lydia selects Check Out, Version Manager will automatically check out the files using the parameters as currently specified in the Check Out dialog. Also, when she double-clicks a versioned file, Version Manager will open a writable version in the appropriate editor. If she needs to review a read-only version of the file, Lydia will select the file and then Edit | View File.

Lydia double-clicks the first file. Because Lydia left the Use default Windows associations checked, Version Manager displays the file in the appropriate editor. She modifies, saves, and closes the file. She then realizes that her changes affect work being done in the Chess project and wants to add a note to the readme file reminding Chess developers to review her related work. She double-clicks the readme file in the Chess project; the file is checked out and automatically displayed in Notepad. Lydia adds the note to the readme file, saves it, and closes the editor.

At this time, Patrick, another developer, selects the Checkers subproject, Library, and then Actions | Check Out. Previously, Ruby has configured the database so that it allows multiple locks on a revision. She has also defined and enabled a promotion model for the database. To allow for multiple locks with a promotion model, Ruby has defined a model with three lowest-level promotion groups, Development, Bug_Fix, and Temp. Lydia has the primary lock on the library files. Therefore, Version Manager displays a message box warning Patrick that the library files are locked by Lydia and asks if he wants to start branches for these files. He would then own secondary locks on these files. He selects Yes.

When a promotion model is in effect for a project database, every checked out revision must be associated with a lowest-level promotion group. Therefore, Version Manager also prompts Patrick to select a different lowest-level promotion group. (Only one revision per archive can be associated with each lowest-level promotion group.) Lydia associated the Development promotion group with the trunk revisions. Patrick is not fixing a bug so he

assigns Temp to the Library branch and selects OK. Later, Patrick will need to merge his changes back into the main development trunk.

Chapter 10

Getting Revisions

Get vs. Check Out

Why get revisions?	Get revisions when you require a workfile but do not need to update it. You can get revisions by selecting a versioned file, multiple versioned files, a project, or an entire project database. If you want to make changes to a revision, use the Check Out option (see "Checking Out Revisions" on page 147).
What happens during a get?	When you get a revision, Serena PVCS Version Manager leaves the revision in the state that it is in (locked or unlocked) and creates a read-only workfile in the specified workfile location.
Specifying a revision	When you get a revision, you can specify the revision by specifying a revision number, version label, or promotion group.
Get projects and project databases	When you choose to get revisions by selecting a project or project database, Version Manager gets the specified revision associated with every versioned file within the selected project or project database.

Default Get Options

Unless your Administrator has changed the defaults for the project's configuration, when you get a revision, Version Manager:

- Creates a read-only copy of the default revision in the temporary directory specified by your system. On Windows, the temporary directory is defined by the TEMP environmental variable. Version Manager creates a `\pvcs` directory in your temporary directory (for example, `\temp\pvcs`). On UNIX, it is defined by the pvcsvmux script, which sets it to `/tmp/pvcs`.
- Gets only the revisions of the selected project and does not include the revisions of the project's subprojects.
- Prompts for confirmation before overwriting the workfile if a writable workfile exists in the current workfile location.

NOTE Version Manager does not prompt for confirmation if a read-only workfile exists in the current workfile location.

- Gets the revision whether or not it is newer than the workfile.
- Retains the promotion group currently assigned to the revision, if a promotion model is in effect.

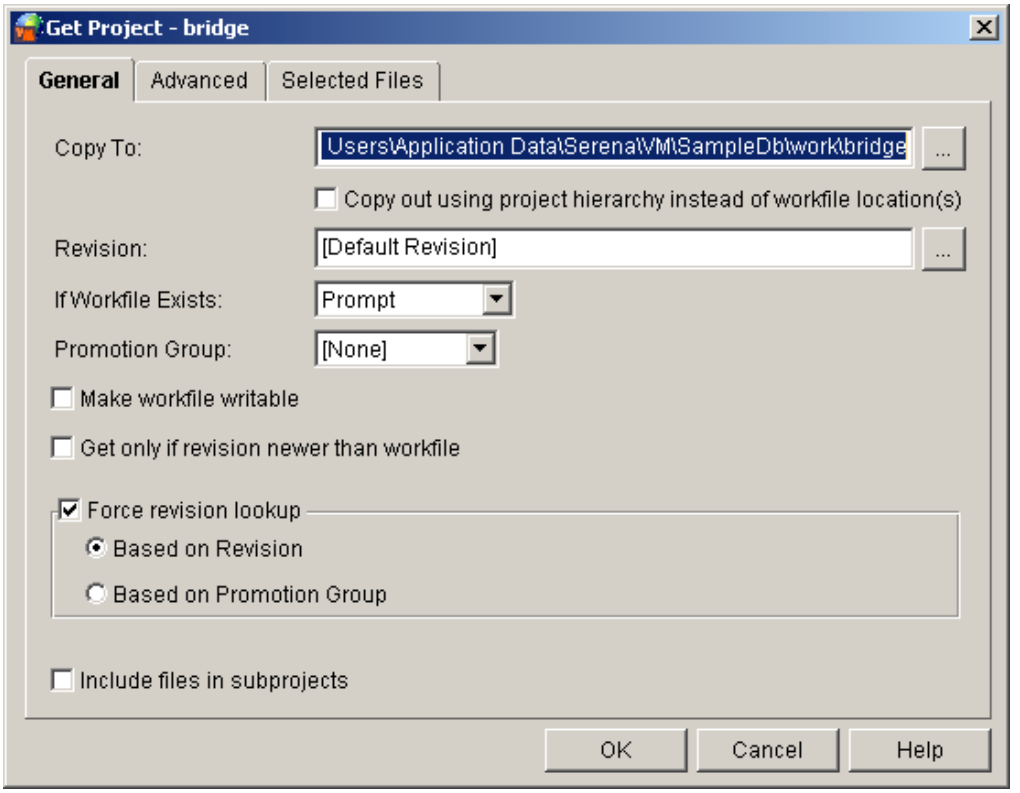
Getting Revisions

To get a revision:

1 Select the versioned file(s), project, or project database associated with the revision that you want to get.

NOTE When you perform a Get operation on multiple projects or folders, the workfile location displayed in the dialog box is the workfile location of the common parent. For more information, refer to "Workfile Locations" on page 35.

2 Select Actions | Get. The Get dialog box appears.



NOTE If you typically get the default revision, you can bypass the Get dialog box by clearing the Get/View check box under View | Options | Dialog Behavior tab.

3 Do one of the following:

- To override the default get options, continue to the next section.
- To accept the default get options, click **OK**. Version Manager copies the selected revisions and places read-only workfiles in their specified workfile locations.

Overriding Default Get Options

1 Under the General tab of the Get dialog box, do any of the following:

- To change the workfile location, edit the location in the **Copy To** field, or click the Browse button to select the location.

  Editing this path changes the workfile location for this get only. It does not permanently change the workfile location.

NOTE When you get a single file, the **Copy To** field displays the file's location and name. You can change the name of the file, which creates a new workfile, and check it back in to create a new archive.

- (For getting multiple versioned files, projects, and project databases only). Select the **Copy out using project hierarchy instead of workfile location(s)** check box to overwrite the workfile location for versioned files with absolute paths and versioned files in subdirectories. This option creates paths and subdirectories that mimic the project structure relative to the path in the **Copy To** field.
- To get a revision other than the default revision, enter the revision number, version label, or promotion group assigned to the revision you want to get in the **Revision** field, or click the Browse button to select it.

  To specify a version label that begins with a number, you must precede it with a backslash (\). For example, `\1.2` or `\1abc`.

  To define the default revision for your project or project database, see "Defining the Default Version" on page 162.
- In the **If Workfile Exists** drop-down menu, select what you want Version Manager to do if the workfile you are copying already exists in the selected workfile location. **Prompt** is the default option, which asks what you want to do when a duplicate workfile exists.

  The other options include **Overwrite**, which adds the workfile even if a duplicate workfile exists, or **Don't Overwrite**, which does not add the workfile to the workfile location.
- If the selected files are associated with a project database that has a promotion model assigned to it, you can get a revision by selecting a promotion group. To get a revision associated with a promotion group, select the promotion group from the **Promotion Group** drop-down menu. If Version Manager cannot find a revision associated with the promotion group you specify, it will check out the revision associated with the next highest group in the promotion model.
- To make the revision writable, select the **Make workfile writable** check box. This is useful when you want to make a copy of the revision for testing purposes. It allows you to edit the revision without locking it, leaving it accessible for other users.
- To get a revision only if it is newer than the workfile, select the **Get only if revision is newer than workfile** check box. This saves time by not getting revisions that have not changed.
- **Force revision lookup:** (Applies only if a promotion model is in effect.) If this check box is **not** selected and you do not specify a promotion group in the **Promotion Group** field, the revision lookup behavior will be as described for the **Based on Revision** option below.

To force different revision lookup logic, select one of the following options:

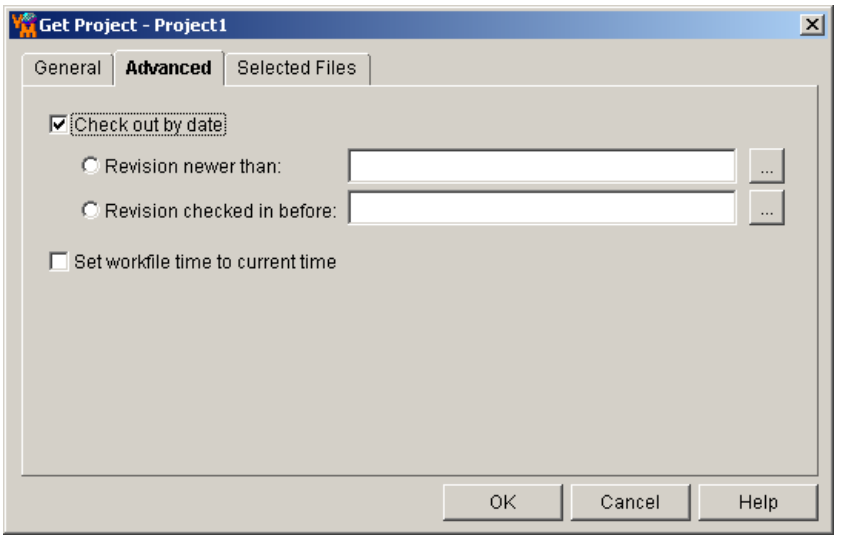
- **Based on Revision:** The revision specified in the **Revision** field will be acted on. You can enter/select a revision number, version label, or promotion group to specify the desired revision.

  If you select **[Default Revision]** in the **Revision** field, the revision specified by the workspace settings or configuration file will be acted on; if no default value is found, the Tip of the Trunk will be acted on.

- **Based on Promotion Group:** The revision assigned to the promotion group in the **Promotion Group** field will be retrieved. If such a revision is not found, the operation will climb the promotion model and act on the revision assigned to the lowest currently assigned group in the promotion model.

- (For projects and project databases only). To get revisions for versioned files located in subprojects, select the **Include files in subprojects** check box.

2 Under the Advanced tab, do any of the following:

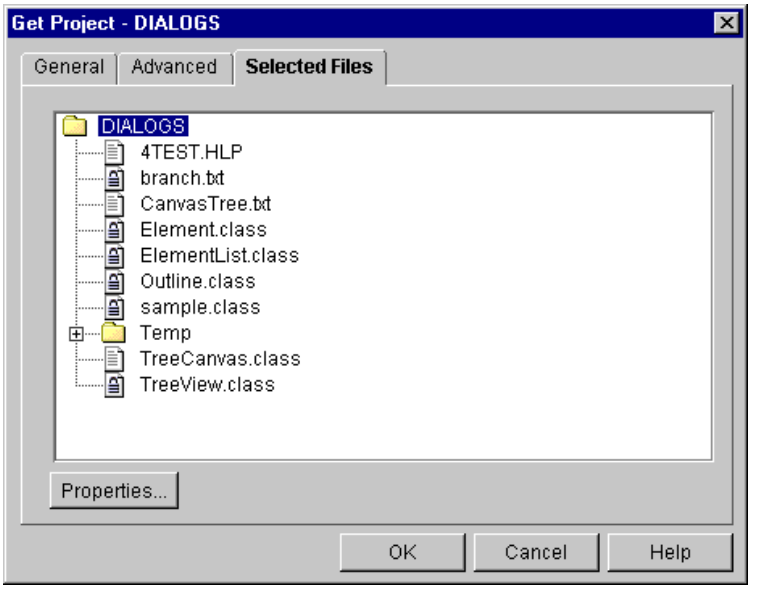


- To get a revision by a specific date and time, select the **Check out by date** check box. By default, this option is not checked.

  The options include **Revision newer than**, which allows you to get a revision based on a specific revision date, and **Revision checked in before**, which allows you to get a revision based on a specific check in date.

- If you want to update the timestamp of the workfile to the current date and time, select the **Set workfile time to current time** check box.

3 Under the Selected Files tab, do any of the following:



- To verify that you are getting the correct files, review the files listed under this tab.
- To review project and workspace setting information of the selected items, click the **Properties** button. Click **OK** to return to the Selected Files tab.

4 Click **OK**. Copies of the selected revisions are copied to their respective workfile locations.

Scenario: Checking Out Read-Only Copies of Project Files

Terri is a new member of the Chess project. Her first task is to fix several bugs in the layout of the chessboard in the November 10 build of the application. However, before starting her work, she wants to review all of the code for the game to familiarize herself with the components and identify the problematic modules. She does not need to edit these files and, hence, does not want to lock other users out of them.

Terri selects the Chess project and chooses the Get option. Because she wants to store the copies of the files in a single local directory, she overrides the default workfile location in the Copy To field with a new location of `c:\review`. She selects the Browse button beside the Revision field and chooses the Version 4.3 Chess Nov_11 version label. The label identifies all of the revisions associated with a November 11 build of the Chess application. She maintains the default for accessing read-only versions of the files. She selects the option for including files in subprojects and then selects OK. Version Manager copies all of the project files associated with the Version 4.3 Nov_11 version label, and she can begin reviewing the existing work.

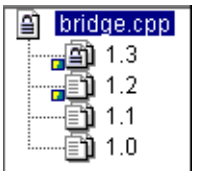
Chapter 11

Working with Revisions

About Revisions

A revision is an instance of a versioned file that can be re-created by checking it out of Serena PVCS Version Manager. When you check out a revision, it becomes a workfile and is stored in the workfile location specified by the versioned file. When you check the workfile back in, it becomes a new revision of the versioned file and is stored in the archive associated with the versioned file.

When you add a workfile for the first time, it is stored in an archive as the initial revision (1.0) of the versioned file. For example, the versioned file "bridge.cpp" below has four revisions: 1.0, 1.1, 1.2, and 1.3. The first revision, 1.0, is the original copy of the workfile.



Defining the Default Version

When you check out or get a revision, the default version is checked out unless you specify a different revision. By default, the default version is the latest revision (also known as the tip) of a versioned file. However, you can define the default version to be a version label.

Project workspace settings

The default version is stored in the Default Version field of your project or project database's workspace settings (File | Properties | Workspace Settings tab). The default version can be specified in the project configuration options by your Administrator or Project Leader. When a default version is specified for your project database, all of the projects within the project database inherit the default version setting. For more information, see "Workspace Hierarchies" on page 112.

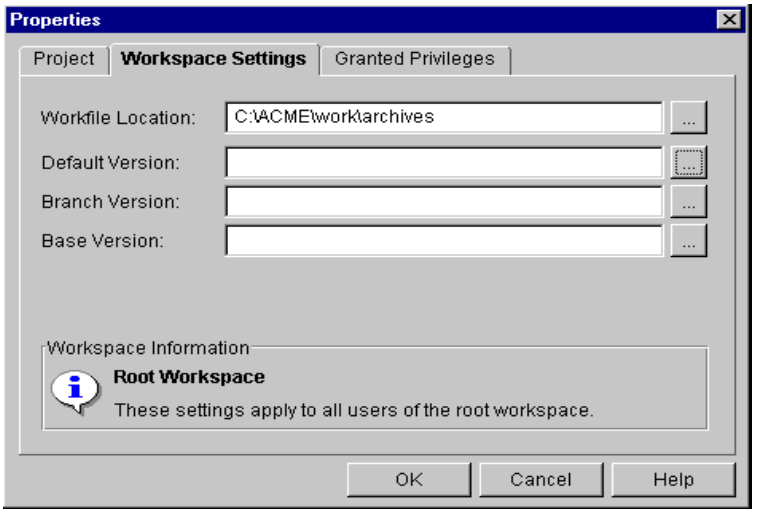
You can define a default version for a project or project database. When you define a default version, Version Manager automatically checks out or gets the revision associated with the specified default version.

NOTE Defining a default version is required when your project database is set up for automatic branching and merging. For information on branching and merging, see the "Branching and Merging Revisions" chapter in the *Serena PVCS Version Manager Administrator's Guide*.

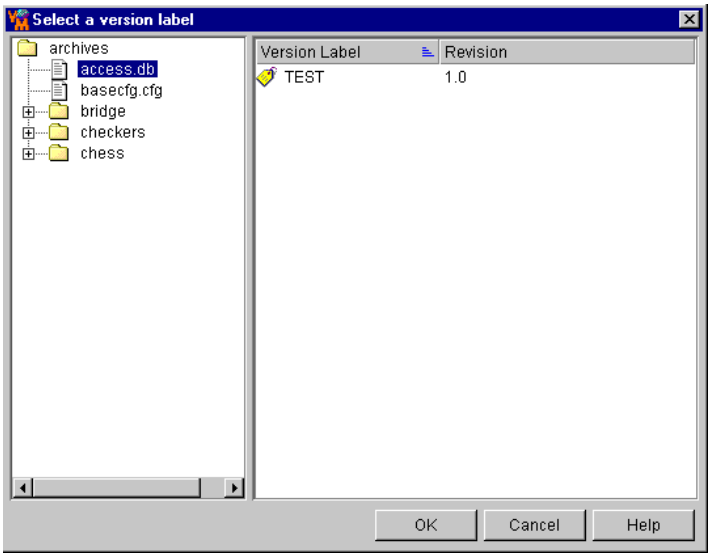
To define the default version:

1 Select the project database or project for which you want to define a default version.

2 Select File | Properties and click the Workspace Settings tab.



3 Enter a version label in the **Default Version** field, or click the Browse button to select one. The Select a version label dialog box appears.



Enter the version label using the correct case. The **Default Version** field is case-sensitive.

4 Click **OK**.

Viewing a Revision

Viewing a revision is basically the same as getting a revision using the Get function. Version Manager creates a read-only copy of the default revision. However, when you view a revision, Version Manager:

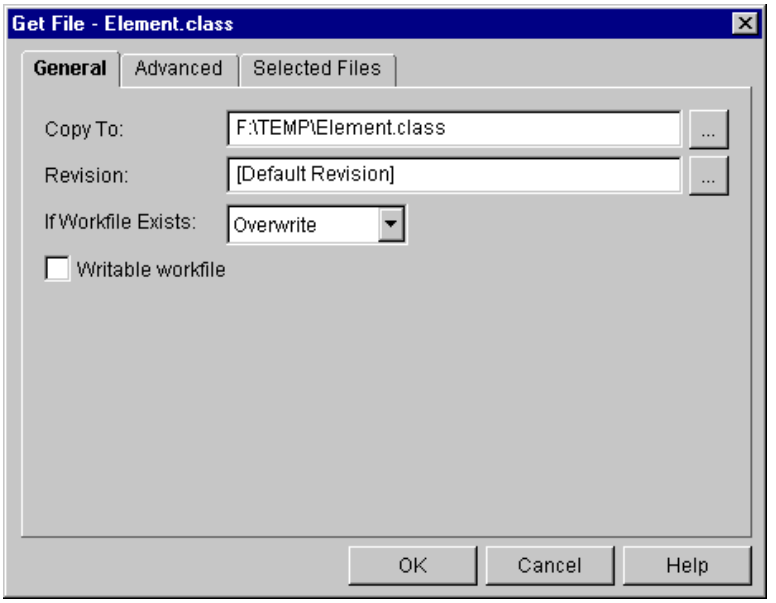
- Copies the workfile into a temporary directory (on Windows: `/temp/pvcs; on UNIX: /tmp/pvcs`). On Windows, the temporary directory is defined by the TEMP environmental directory. On UNIX, it is defined by the pvcsvmux script.

- Launches the application associated with the workfile and opens the file. The editor Version Manager uses to display the file is either defined by the default Windows associations or by entering an editor location in the View | Options dialog box.

NOTE For more information on how Version Manager associates your revisions with a specific application, see "Setting Your Default Editor" on page 137.

To view a revision:

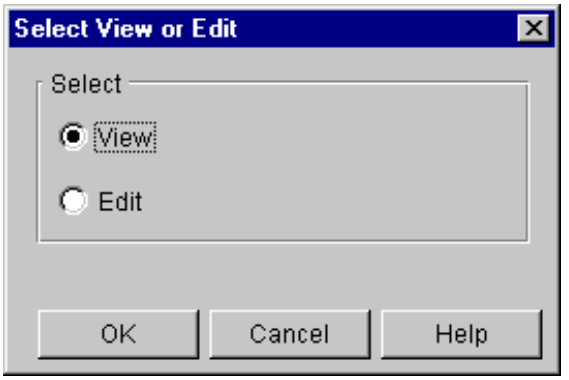
1 Select the versioned file you want to view in the File pane and select Edit | View File. The Get dialog box appears.



NOTE If you typically view the default revision, you can bypass the Get dialog box by clearing the Get/View check box under View | Options | Dialog Behavior tab.

2 Click **OK**. Version Manager opens a read-only copy of the workfile in the associated application.

NOTE If the **Default Double-Click Action** setting (View | Options | Applications tab) is set to Ask or View, double-click the versioned file in the File pane. The Select View or Edit dialog box appears. Select View and click **OK**. The Get dialog box appears with file selected. Click **OK**.



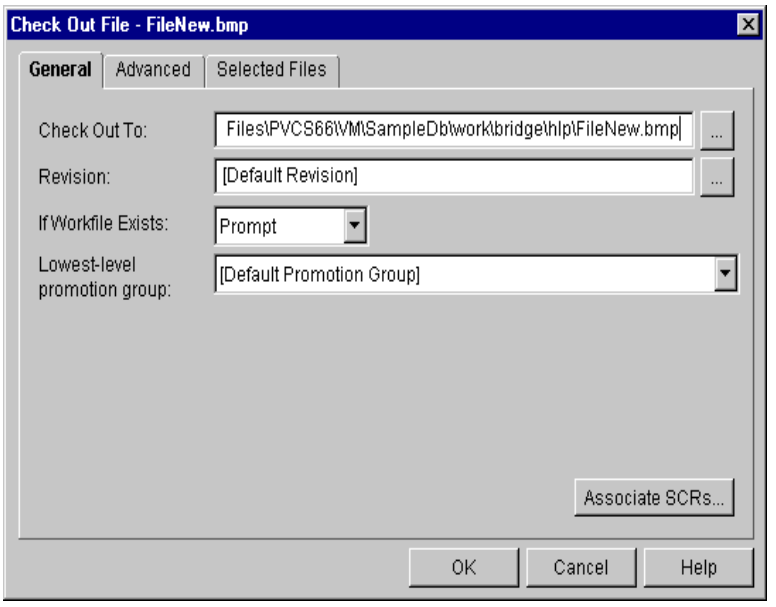
Editing a Revision

Editing a revision provides the same basic features as the Check Out function: it checks out the default revision from the location currently set as the workfile location. However, when you edit a revision, the application associated with the workfile is launched and a writable copy of the revision is opened.

This revision is copied to the temporary directory defined for your system instead of the location currently set as the workfile location. On Windows, the temporary directory is set by the TEMP environmental variable. Version Manager creates a `/pvcs` directory in your temporary directory (for example, `\temp\pvcs`). On UNIX, this temporary directory is defined by the pvcsvmux script, which sets it to `/tmp/pvcs`.

To edit a revision:

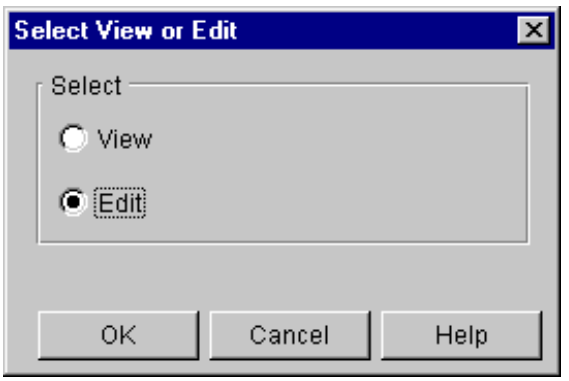
1 Select the versioned file you want to edit in the File pane and select Edit | Edit File. The Check Out dialog box appears.



NOTE If you typically edit the default revision, you can bypass the Check Out dialog box by clearing the Check Out/Edit check box under View | Options | Dialog Behavior tab.

2 Click **OK**. Version Manager locks the revision and opens the workfile in the associated application.

NOTE If the **Default Double-Click Action** setting (View | Options | Applications tab) is set for Ask or View, double-click the versioned file in the File pane. The Select View or Edit dialog box appears. Select Edit and the Check Out dialog box appears with the file selected. Click **OK**. To skip the Select View or Edit dialog boxes, set the Default Double-Click Action to Edit.

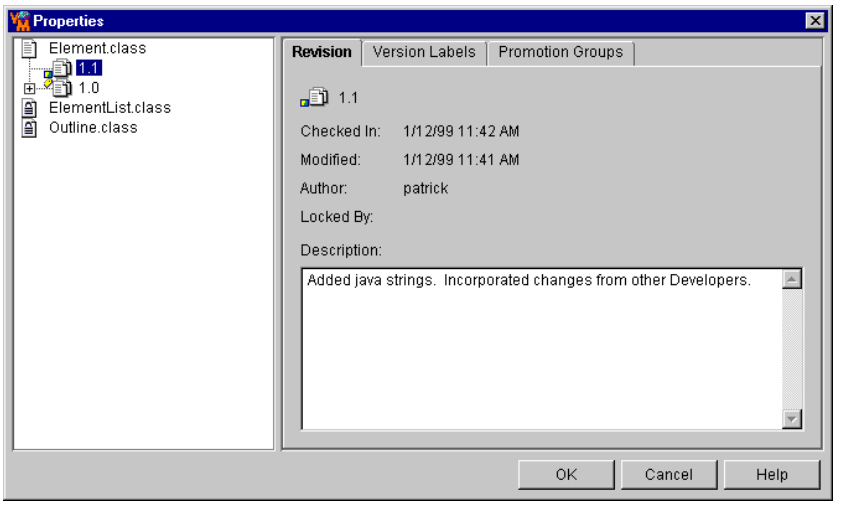


Adding/Modifying Change Descriptions

You can add or modify the change description of a versioned file or revision at any time. Change descriptions are valuable for identifying the changes that were made to a specific revision.

To add or modify a change description to a revision or versioned file:

1. Select the versioned files or revisions (if the Revision pane is displayed) associated with the change descriptions that you want to change.
2. Select File | Properties. The Properties dialog box appears. If you selected a versioned file, the Versioned File tab appears. If you selected a revision, the Revision tab appears.



3. To make changes to either the versioned file or revision, select the item in the left pane. Notice the tabs change depending on the item you selected.
4. Enter your changes in the **Description** field.
5. Repeat steps 3 - 4 if you selected multiple versioned files or revisions in step 1.
6. Click **OK**.

Deleting Revisions

You can delete specific revisions from an archive.

CAUTION! When you delete a revision, the action is *permanent* and the revision is *not recoverable*.

To delete a revision:

1. Select the versioned file associated with the revision you want to delete. If the Revision pane is displayed, select the revision that you want to delete.
2. Select File | Delete or click **Delete**. The Confirm Item Deletion message appears.

3 Click **Yes**. Clicking Yes permanently deletes the revision from the archive and the action cannot be undone.

Chapter 12

Checking In Workfiles

About Checking In

When do you check in workfiles?

After you make changes to a workfile, check in the workfile when you want to:

- Preserve the changes made to the workfile
- Make your changes available to other Project Team Members

Each time you check in a workfile, it becomes a new revision in the versioned file.

What happens during check in?

By default, when you check in a workfile, Serena PVCS Version Manager creates a new revision, assigns it the next number in sequence, and keeps a read-only workfile in the workfile location.

Checking in projects and project databases

When you choose to check in workfiles by selecting a project or project database, Version Manager checks in the workfiles associated with every versioned file within the selected project or project database.

Associating issues

If you have installed TrackerLink or SourceBridge, you can associate issues from the Check In dialog box by clicking the **Associate Issues** button.

NOTE TrackerLink or SourceBridge is invoked automatically if you or your administrator have set it up to require that you associate issues when checking in workfiles.

NOTE You can set which issue management integration will be used by the next invocation of the Version Manager Desktop Client. Any currently open client sessions will not be affected.

1. Launch the Serena Issue Management Integration utility (from the Serena folder of the Windows Start menu, select Version Manager | Issue Management Integration).
2. Select **TeamTrack SourceBridge** or **Tracker TrackerLink**.
3. Click the **OK** or **Launch Version Manager** button.

Default Check In Options

Version Manager uses the following initial defaults when you check in a workfile. These defaults can be overridden when you check in the workfile; and in some cases, the defaults can be redefined in the Options dialog box (View | Options | General tab). If you accept the defaults, Version Manager:

- Checks in files from the location to which they were checked out. If a file has no check out location (as when a file is locked, but not checked out), it will be checked in from its default workfile location as defined in the current workspace, unless you specify an alternate location.
- Stores the workfile as the next revision in the archive, assigns it the next revision number in sequence, and leaves the revision unlocked.
- Leaves a read-only copy of the workfile in the location currently set as the workfile location.

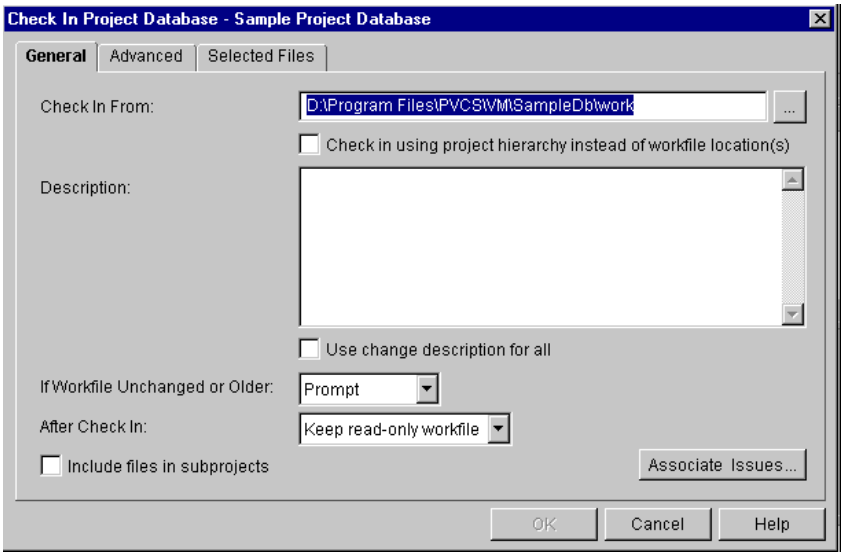
- Checks in only the workfiles of the selected project and does not include the workfiles of the project's subprojects.
- Prompts for confirmation before checking in the workfile if the workfile is unchanged or older than the previous revision in sequence.
- Retains the promotion group currently assigned to the revision, if a promotion model is in effect.
- Does not assign a version label.
- Does not automatically invoke TrackerLink or SourceBridge for associating issues when checking in workfiles.
- Does not create a branch (unless the workfile checked out was not the latest revision).

Checking In Workfiles

To check in workfiles:

1 Select the workfile or workfiles, the project, or the project database that contains the workfiles that you want to check in.

2 Select Actions | Check In. The Check In dialog box appears.



3 Enter a description of the changes that you made to the workfile(s) in the **Description** field. Be descriptive so that other Project Team Members will be able to tell the difference between revisions.

4 Do one of the following:

- To override the default check in options, continue to the next section.
- To accept the default check in options, click **OK**. The selected workfiles are checked in, and read-only workfiles are placed in their specified workfile locations.

Overriding Default Check In Options

1 Under the General tab of the Check In dialog box, do any of the following:

- To change the check in location of the workfile being checked in, edit the location in the **Check In From** field, or click the Browse button to select the location.

  By default, the **Check In From** location is the location to which the files were checked out. Editing this path changes the check in location for this check in only.

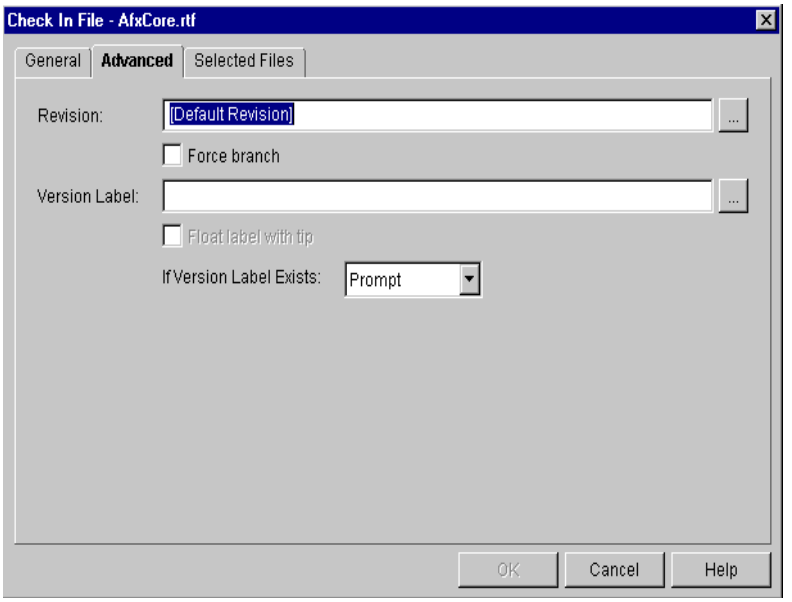
NOTE If the check out location of the workfile does not match the check in location, a warning appears.

- (For checking in multiple versioned files, projects, and project databases only). To overwrite the check in location for workfiles with absolute paths and workfiles in subdirectories, select the **Check in using project hierarchy instead of workfile location(s)** check box. This option creates paths and subdirectories that mimic the project structure relative to the path in the **Check In From** field.
- (For adding multiple workfiles only.) If you want Version Manager to prompt for a unique description for each workfile you are adding, deselect the **Use change description for all** check box. Otherwise, the same description will be used for every file.
- In the **If Workfile Unchanged or Older** drop-down menu, select what you want Version Manager to do if the workfile you are checking in is unchanged or older than the previous revision. **Prompt** is the default option, which asks what you want to do if the workfile is unchanged.

  The other options include **Check In**, which checks in the workfile even if it is unchanged, or **Don't Check In**, which does not check in the workfile to the archive location.
- In the **After Check In** drop-down menu, select what you want Version Manager to do with the workfile after the workfile is checked in. If you want to:
  - Delete the workfile from the **Check In From** location, select the **Delete workfile** option.
  - Keep a read-only copy of the workfile in the workfile location, select the **Keep read-only workfile** option.
  - Keep a writable copy of the workfile in the workfile location, select the **Keep writable workfile** option.
  - Lock the revision that will be created once the workfile is checked in, select the **Keep revision locked** option.
- (For projects and project databases only). To check in workfiles located in subprojects, select the **Include files in subprojects** check box.
- (For Serena Tracker or SourceBridge Users only). Click the **Associate Issues** button if you want to associate the workfiles you are checking in with issues. This displays the association dialog box.

NOTE TrackerLink or SourceBridge is invoked automatically if you or your administrator have set it up to require that you associate issues when checking in workfiles.

2 Under the Advanced tab, do any of the following:



- In the **Revision** field, enter a new revision number for the workfile you are checking in if you want it to be a number other than the next number in sequence.

  This field has two purposes. The main purpose is to specify a new revision number for the workfile you are checking in. You must enter the new revision number in the field; you cannot select it.

  The second purpose of this field is to identify the locked revision you want to check in when you have multiple revisions of a versioned file checked out with a lock.

IMPORTANT! A floating label **cannot** float past a change in the revision numbering scheme.

For example, if you assigned a floating label to revision 1.3, it would float to the new revisions (1.4, 1.5, and finally to 1.6) as they were created. However, if you assigned an arbitrary number to any digit to the left of the decimal point (say **2**.0 or **2**.7 instead of 1.7), the label would not move to the new revision. The same logic applies to branch revisions (1.14.1.2, 1.14.1.3, 1.14.1.4) 1.14.**2**.0.

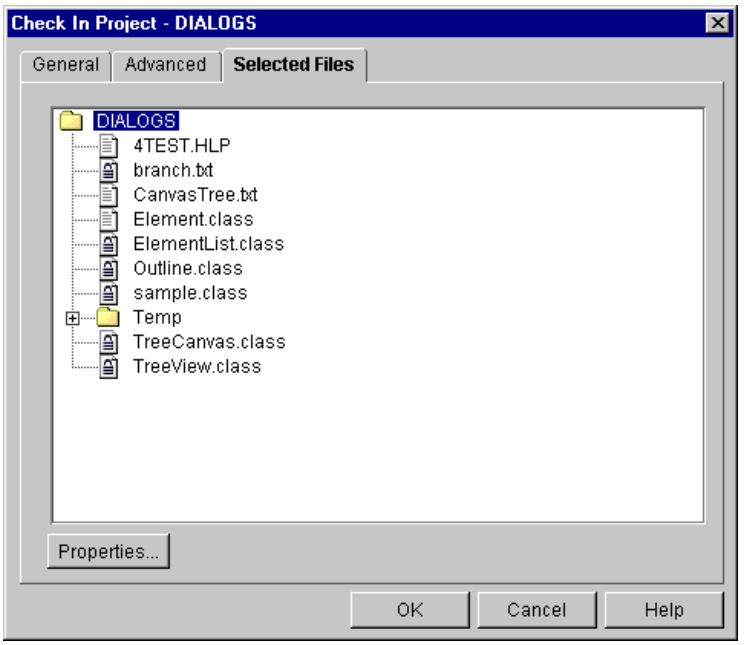
- Select the revision number of one of the locked revisions or select a promotion group or version label associated with one of the locked revisions. If you do not select a revision number, you will be prompted to select one.

  To specify a revision, enter the revision number in the **Revision** field.

- To force a branch using this revision, select the **Force Branch** check box.
- To assign a version label, enter the version label in the **Version Label** field, or click the Browse button to select a version label. If you choose to enter the version label, note that it is case sensitive.
- To keep the version label associated with the latest (tip) revision of the current trunk or branch, select the **Float label with tip** check box.
- In the **If Version Label Exists** drop-down menu, select what you want Version Manager to do if an identical version label is already assigned to a revision within the selected archive. **Prompt** is the default option, which asks what you want to do when an identical version label exists.

The other options include **Reassign**, which reassigns the version label to the revision you are checking in, or **Don't Reassign**, which cancels check in if the label already exists.

3 Under the Selected Files tab, do any of the following:



- To verify that you are checking in the correct files, review the files listed under this tab.
- To review project and workspace setting information of the selected items, click the **Properties** button. Click **OK** to return to the Selected Files tab.

4 Click **OK**. The selected workfiles are checked in to their respective archive locations.

Scenario: Checking In a Set of Project Files

Lydia has completed a set of changes to the Checkers files and now wants to store her changes in Version Manager. She has not made changes to all of the files but, in the interest of time, chooses to check in the entire project regardless. She selects the project and then chooses the Check In option.

She verifies that the Check In From field displays her personal working directory of `c:\work`. She then enters a description of her changes—"Completed definition of drag and drop functionality". Version Manager stores the description as a characteristic of each revision.

Lydia knows that she is checking in several files that have not been changed. She does not want to individually confirm the check in of these files so she selects the **Don't Check In** option in the **If Workfile Unchanged or Older** field. Version Manager checks in just the modified files.

In addition, because Lydia has additional work to perform on the Checkers files, she does not want others to modify these files. She selects the **Keep revision locked** check box in the After Check In group. Lydia will retain the locks on the modified files, prohibiting other

Users from modifying the files without creating a new development branch. Lydia selects the Include files in subprojects option to verify that Version Manager checks in all of the modified files in the Checkers project and its subprojects.

On the Advanced tab, Lydia retains the default for Revision. However, she wants to assign a version label to her ongoing work. This label will allow her to easily identify the set of files that define the drag and drop functionality. Therefore, she assigns a version label of Drag_Drop_November. She does not select the Float label with tip option because she wants the label to maintain its association with these revisions instead of automatically moving to later versions of the files. Her work is part of the trunk development for the Checkers project and, hence, she does not want to force a branch. She clicks **OK** to check in the files. Version Manager performs the following actions:

- Checks in the modified files in the `c:\work` directory
- Does not check in unmodified files
- Retains writable copies of all of the Checkers files in the `c:\work directory`.
- Associates the "Completed definition of drag and drop functionality" description with each modified revision.
- Associates a fixed Drag_Drop_November version label to each modified revision.

Chapter 13

Using Locks

Locking Revisions

When to use locks

Use locks on a revision you are working on when you want to prevent other Project Team Members from editing it. A lock controls access to revisions by warning other users that a file is in use. Other Project Team Members can always view or get the revision, but they cannot check in and overwrite the revision.

When you check out a revision, a lock is automatically placed on the revision. The lock remains on the revision until you either check the workfile back in or unlock it.

What happens during a lock

When you lock a versioned file, technically, you are locking a specific revision of the versioned file. Locking a specific revision denies other Project Team Members access to that revision.

The only time other Project Team Members can edit a locked revision is if multiple locking has been enabled on the project you are working on. For more information on multiple locking, see page 183.

Specifying a lock

You can lock a revision by specifying a revision number, version label, or promotion group. You can lock revisions by selecting a single revision or versioned file, multiple versioned files, a project, or an entire project database.

Associating issues

If you have installed TrackerLink or SourceBridge, you can associate issues with the locked files from the Lock File dialog box by clicking the **Associate Issues** button.

NOTE TrackerLink or SourceBridge is invoked automatically if you or your manager have set it up to require that you associate issues when locking revisions.

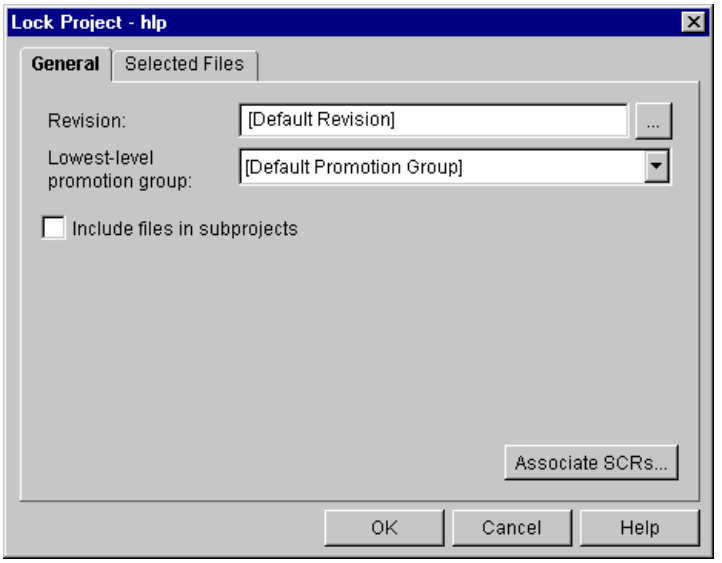
NOTE You can set which issue management integration will be used by the next invocation of the Version Manager Desktop Client. Any currently open client sessions will not be affected.

1. Launch the Serena Issue Management Integration utility (from the Serena folder of the Windows Start menu, select Version Manager | Issue Management Integration).
2. Select **TeamTrack SourceBridge** or **Tracker TrackerLink**.
3. Click the **OK** or **Launch Version Manager** button.

To lock a revision:

1. Select the revision (if the Revision pane is displayed) or versioned file that you want to lock. If you want to lock a group of revisions, select the project or project database that contains the revisions you want to lock.

2 Select Actions | Lock. The Lock dialog box appears.



TIP If you typically lock the default revision, you can bypass the Lock dialog box by clearing the Lock check box under the View | Options | Dialog Behavior tab. Note that if you are required to associate issues with locked revisions, bypassing the Lock dialog box does not affect the automatic display of the association dialog box.

3 Under the General tab, do any of the following:

- In the **Revision** field, specify the revision number or version label associated with the revision that you want to lock. You can also specify a promotion group if the revision you selected is associated with a project database that has a promotion model defined.

NOTE To specify a version label that begins with a number, you must precede it with a backslash (\). For example, `\1.2` or `\1abc`.

If you do not specify a revision, Version Manager will lock the default revision specified for the project or project database that contains your revisions. The default revision is the latest revision unless otherwise specified.

To define the default revision for your project or project database, see "Defining the Default Version" on page 162.

- In the **Lowest-level promotion group** field, select a promotion group from the drop-down menu. This field is only available if a promotion model is defined in the current project database. The promotion group you select will be associated with the locked revision.

  The default value is **[Default Promotion Group]**, which is a workspace setting that defines a lowest-level promotion group to use for this action. If a value for this workspace setting is not defined and you do not select another value in this field, the default behavior is as follows:

  - If more than one lowest-level promotion group exists in the promotion model, Version Manager prompts you to select which lowest-level promotion group to use for this action.

- If only one lowest-level promotion group exists in the promotion model, Version Manager uses this promotion group.

- If the **Include files in subprojects** check box is selected, the above description applies to subprojects with a promotion model defined if the project does not have one defined.

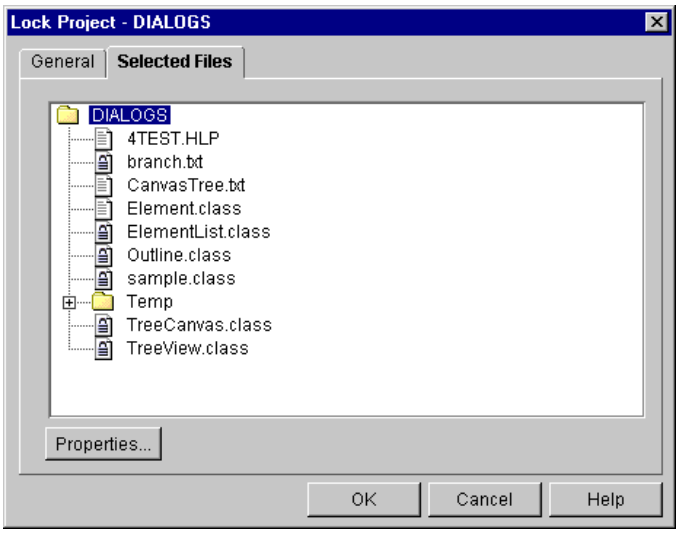
NOTE This field is not used to select the revision to be locked. To lock a revision based on a promotion group, enter a promotion group in the **Revision** field.

- (For Serena Tracker and SourceBridge users only). Click the **Associate Issues** button if you want to associate the workfiles you are locking with issues. This displays the association dialog box.

  Note that TrackerLink or SourceBridge is invoked automatically if you or your manager have set it up to require that you associate issues when locking revisions.

NOTE If you disable the Lock dialog box, to prevent it from displaying when performing a lock operation, but you enable the association dialog box and require issues associations, Version Manager displays the association dialog box when you lock revisions.

4 Under the Selected Files tab, do any of the following:



- To verify that you are locking the correct files, review the files listed under this tab.
- To review project and workspace setting information of the selected items, click the **Properties** button. Click **OK** to return to the Selected Files tab.

5 Click **OK**. The selected revisions are locked.

Unlocking Revisions

When to unlock revisions

When you check in a workfile, Version Manager automatically removes the lock from the default revision. However, you may need to unlock a revision without checking the revision back in.

For example, if you check out an entire project and check in only those files that have changes, any file that did not change will still have a lock on it. Instead of checking in an unchanged workfile just to unlock the revision, you can use the unlock option to remove the lock.

What happens during an unlock

When you unlock a versioned file, you are unlocking a specific revision of the versioned file. By unlocking a revision, you are freeing up the revision and allowing other Project Team Members access to the revision.

If you unlock a file containing an issue association that was locked but not modified, the association is dissolved and a notation is made in the issue indicating that the association was removed.

Unlocking privileges

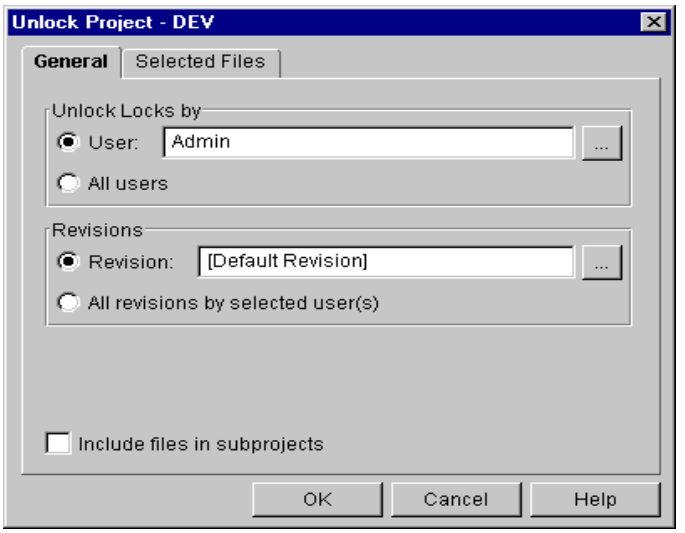
Unlocking privileges are based upon your user ID and access list group privileges established by your Administrator. By default, you can only unlock a lock that you have locked (unless you have SuperUser or Unlimited privileges).

Unlocking specific revisions

You can unlock revisions by specifying a revision number, version label, or promotion group. You can unlock revisions by selecting a single revision or versioned file, multiple versioned files, a project, or an entire project database.

To unlock a revision:

1. Select the revision (if the Revision pane is displayed) or versioned file that you want to unlock. If you want to unlock a group of revisions, select the project or project database that contains the revisions you want to unlock.
2. Select Actions | Unlock. The Unlock dialog box appears.



Under the General tab, do any of the following:

- In the Unlock Locks by group, the user ID you used to log in to the selected project database is displayed in the **User** field.

  If your privileges include the Break Lock privilege, you can remove a lock that was created using a different user ID. Click the Browse button to select a different user ID.

  If you have SuperUser privileges, you can select the **All Users** option to unlock locks created by all users.

- In the Revisions group, specify the revision number or version label associated with the revision that you want to unlock in the **Revision** field. You can also specify a promotion group if a promotion model is defined for the project database you selected.

NOTE To specify a version label that begins with a number, you must precede it with a backslash (\). For example, `\1.2` or `\1abc`.

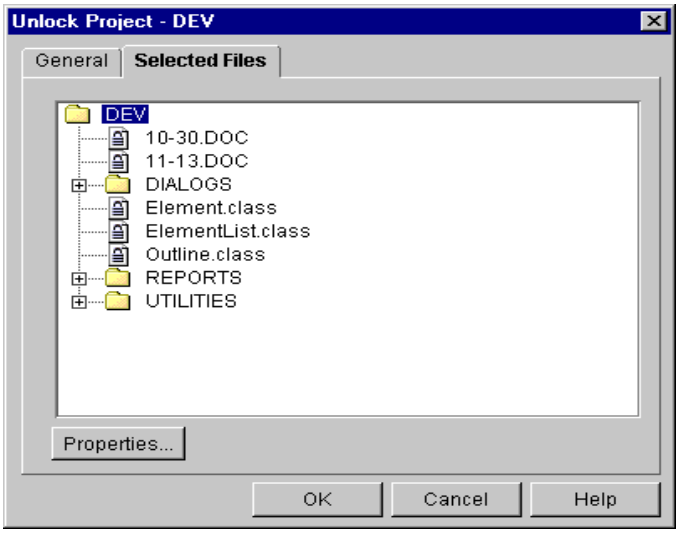
If you do not specify a revision, Version Manager will unlock the default revision specified for the project or project database that contains the revisions. The default revision is the latest revision unless otherwise specified.

To define the default revision for your project or project database, see "Defining the Default Version" on page 162.

You can also select the **All revisions by selected user(s)** option to unlock all revisions associated with the user ID specified in the **User** field.

- (For projects and project databases only). By default, Version Manager does not unlock files within subprojects. If you want to unlock files in subprojects, select the **Include files in subprojects** check box.

3 To review the selected files or projects before you unlock them, click the Selected Files tab.



4 Under the Selected Files tab, do any of the following:

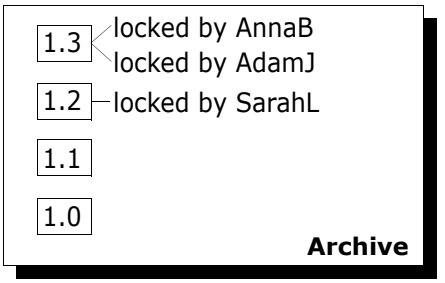
- To verify that you are unlocking the correct files, review the files listed under this tab.
- To review project and workspace setting information of the selected items, click the **Properties** button. Click **OK** to return to the Selected Files tab.

5 Click **OK**. The selected revisions are unlocked.

Multiple Locking

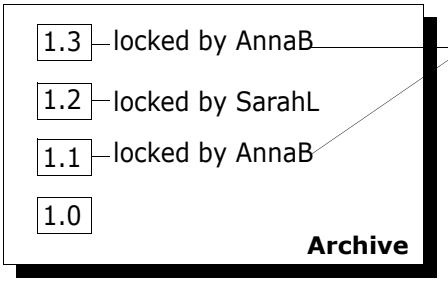
By default, Version Manager creates projects that allow one lock per revision. However, your Administrator can set up projects that allow for multiple locks per revision, multiple locks by the same user within an archive, or multiple locks per revision by the same user, as shown below.

- Multiple locks on a single revision (multiple locks per revision)



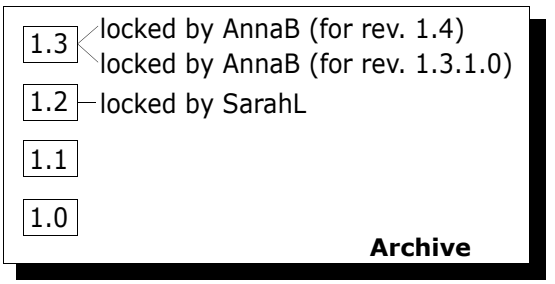
No user can have more than one lock in an archive but multiple locks on a revision are allowed.

- Multiple revisions in an archive locked by a single user (multiple locks per user)



A user can lock more than one revision in an archive but no revision can have more than one lock.

- Multiple locks on a single revision by a single user (multiple locks per revision per user)



A user can have multiple locks on a revision in the same archive.

Multiple locking is designed for Project Team Members who maintain parallel development paths, or branches. For more information on multiple locking, see the "Branching and Merging Revisions" chapter in the *Serena PVCS Version Manager Administrator's Guide*.

Scenario: Prohibiting Other Users from Modifying Files

Mark, the Team Leader, and six additional Writers are documenting the Bridge project. They do not yet have a working version of the product. Instead, they have been using product specifications to document the features. By using specifications completed in August, they have completed versions of all of the dialog help topics. However, Mark

recently discovered that many of the dialog boxes have undergone major changes due to findings from a recent usability test. These changes are not reflected in the August version of the specifications. Mark wants to prohibit the Writers from modifying these files until the specifications are brought up to date. He does not want the Writers to waste their time documenting features with incorrect information.

The development Project Leader assigned the version label, Usability_Chgs, to each of the modified dialog boxes. Mark selects the Bridge project and filters his view using the Usability_Chgs version label. He also clicks the recursive view button to display all of the files within the Bridge project that match the Usability_Chgs version label. The File pane only lists the versioned files that have the version label assigned to it. He uses this list to identify the corresponding documentation topics.

Mark holds down the CTRL key and selects each of the versioned files in the File pane and then selects the Lock option. This will prohibit access to the files. Mark accepts the lock defaults and clicks **OK**. The versioned files display with an associated lock icon in the File pane.

Mark wants to remind the Writers why these files are locked. With the versioned files still highlighted in the File pane, he assigns a version label to the locked revisions. He enters "On Hold" as the name for the non-floating, or *fixed*, version label and then clicks **OK**.

Mark resets his filter to All Files and decides to display the Revisions pane to verify that the appropriate revisions are locked and associated with the On Hold version label. Once the specifications are updated, Mark will unlock the files and remove the version label. These tasks are described in the next scenario.

Chapter 14

Using Version Labels

About Version Labels

What are version labels?

Version labels are tags used to identify revisions. Typically, version labels are used to identify a particular revision for each component of a specific product release, such as "Beta Test 1."

Assign a version label to a specific revision when you want to distinguish the revision from other revisions within the same versioned file or group of versioned files.

You can assign version labels by selecting a revision, one or more versioned files, a project, a 5.3/6.0 folder, an entire project database, an existing version label, or a promotion group.

Once labels are assigned to revisions, you can perform actions by specifying version labels instead of revisions. These actions include, but are not limited to, checking in workfiles, checking out revisions, promoting groups of files, and generating reports.

Multiple version labels

You can assign multiple version labels to a revision; however, version labels must be unique *within* each versioned file.

Case-sensitive

Version labels are case-sensitive. When working with version labels, make sure to specify the exact case of the version label.

Character limitations

Version labels have a limit of 254 characters. You can use alpha, numeric, and special characters except for colons (:), asterisks (*), plus signs (+), minus signs (-), and double-quotation marks (").

NOTE It is best practice to use version labels that do **NOT** look like revision numbers.

Fixed vs. Floating Labels

There are two types of version labels, fixed and floating:

- **Fixed:** Remains assigned to a specific revision of the file until you move it to another specific revision.
- **Floating:** Moves to the tip (newest) revision of the file whenever a new revision is added.

A floating label allows you to refer to the latest revision in an archive without knowing its revision number. This is especially useful when working with projects that contain branches, as you can assign a unique floating label to the tip of each branch and the trunk.

IMPORTANT! A floating label **cannot** float past a change in the revision numbering scheme.

For example, if you assigned a floating label to revision 1.3, it would float to the new revisions (1.4, 1.5, and finally to 1.6) as they were created. However, if you assigned an arbitrary number to any digit to the left of the decimal point (say **2**.0 or **2**.7 instead of 1.7), the label would not move to the new revision. The same logic applies to branch revisions (1.14.1.2, 1.14.1.3, 1.14.1.4) 1.14.**2**.0.

Assigning Version Label Default Options

When you assign a version label, if you do not specify a revision or change any of the values in the Assign Version Label dialog box, Version Manager:

- Assigns the version label to the default revision. If a default revision is not specified, the latest (tip) revision is used.
- Assigns a fixed version label to the default revision.
- Prompts for confirmation before overwriting an existing version label.

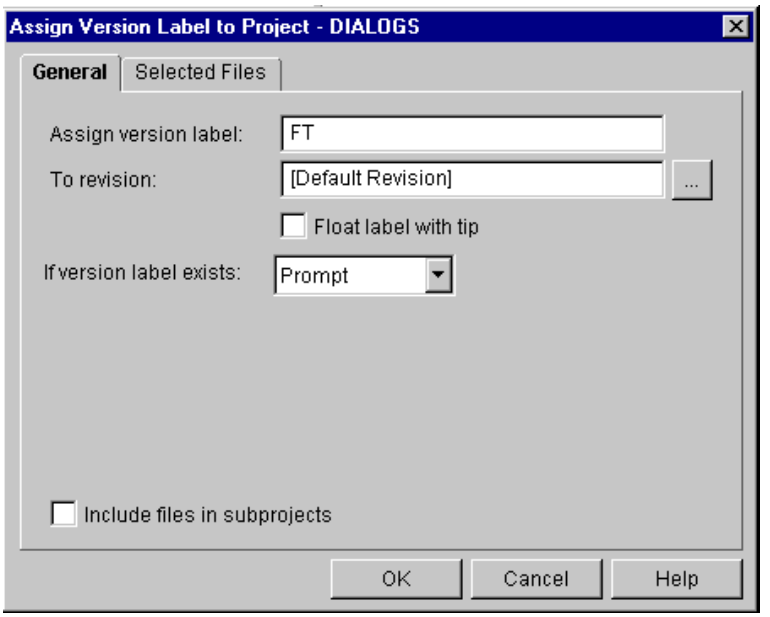
Assigning Version Labels

You can assign a version label by selecting one or more revisions, one or more versioned files, a project, a folder, a project database, an existing version label, or a promotion group.

To assign a version label:

1 Select the item that you want to assign a version label.

2 Select Actions | Version Label | Assign. The Assign Version Label dialog box appears.



3 Enter a version label in the **Assign version label** field. Version labels have a limit of 254 characters. You can use alpha, numeric, and special characters except for colons (:), asterisks (*), plus signs (+), minus signs (-), and double-quotation marks (").

NOTE It is best practice to use version labels that do **NOT** look like revision numbers.

4 Do one of the following:

- To override the assigning version label default options, continue to the next section.
- To accept the assigning version label default options, click **OK**. A fixed version label is assigned to the default revision specified for the item(s) selected.

Overriding Default Version Label Options

1 In the Assign Version Label dialog box, do any of the following:

- To assign the version label to a revision other than the default revision, enter the revision number, version label, or promotion group in the **To revision** field, or click the Browse button to select the revision.

  To define the default revision for your workspace, see "Defining the Default Version" on page 162.
- To move the version label with the latest (tip) revision, select the **Float label with tip** check box.
- In the **If version label exists** drop-down menu, select what you want Version Manager to do if the version label you are assigning already exists in a versioned file. **Prompt** is the default option, which asks what you want to do when a duplicate version label exists.

  The other options include **Reassign**, which moves the existing version label to the specified revision, or **Don't Reassign**, which does not assign the version label to the versioned file.
- (For projects and project databases only). To assign version labels to all of the versioned files located in subprojects, select the **Include files in subprojects** check box.

2 Click **OK**. The version label you specified is assigned to the selected revisions.

Assigning Version Labels When You Check In and Add Workfiles

You can also assign version labels to revisions when you check in and add workfiles. Version label options are available from the Advanced tab under the Check In and Add Workfiles dialog boxes. For information on assigning version labels during check in, see "Overriding Default Check In Options" on page 172. For information on assigning version labels when adding workfiles, see "Adding Workfiles to Project Databases/Projects" on page 90.

Renaming Version Labels

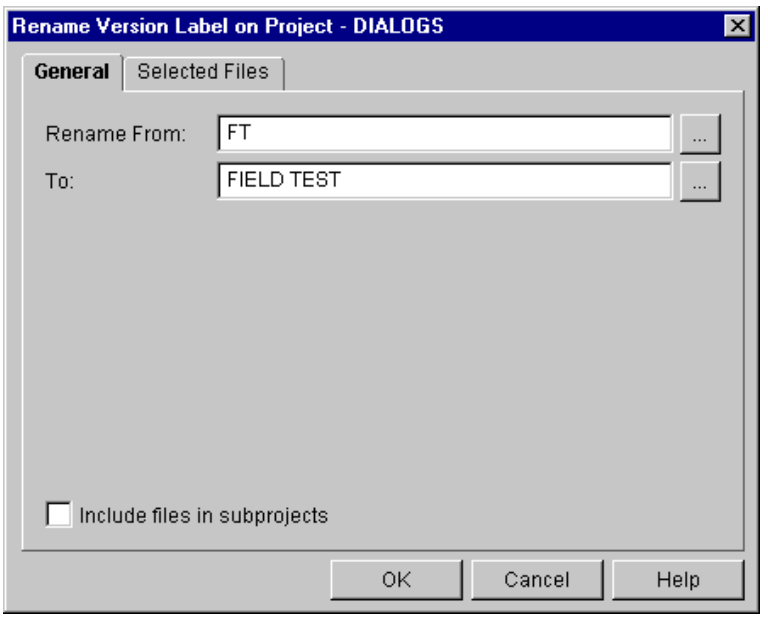
Rename a version label to change an existing version label. You can rename version labels by selecting an existing version label, revision, one or more versioned files, a project, a 5.3/6.0 folder, or an entire project database.

When you rename version labels at the project, folder, or project-database level, all of the version labels that match the version label you are renaming are changed.

To rename a version label:

1 Select the version label or the item that contains the version label that you want to change.

2 Select Actions | Version Label | Rename. The Rename Version Label dialog box appears.



3 Enter the version label that you want to rename in the **Rename From** field, or click the Browse button to select a version label.

4 Enter the new version label in the **To** field, or click the Browse button to select a version label. Version labels have a limit of 254 characters. You can use alpha, numeric, and special characters except for colons (:), asterisks (*), plus signs (+), minus signs (-), and double-quotation marks (").

NOTE It is best practice to use version labels that do **NOT** look like revision numbers.

5 (For projects and project databases only). To rename version labels located in subprojects, select the **Include files in subprojects** check box.

6 Click **OK**.

Reassigning Version Labels

You can reassign a version label when you want to:

- Move an existing version label from one revision to another *within* the same versioned file

- Change the properties of a version label (from a floating version label to a fixed version label, or vice versa)

You can reassign version labels by selecting an existing version label, revision, one or more versioned files, a project, a 5.3/6.0 folder, or an entire project database.

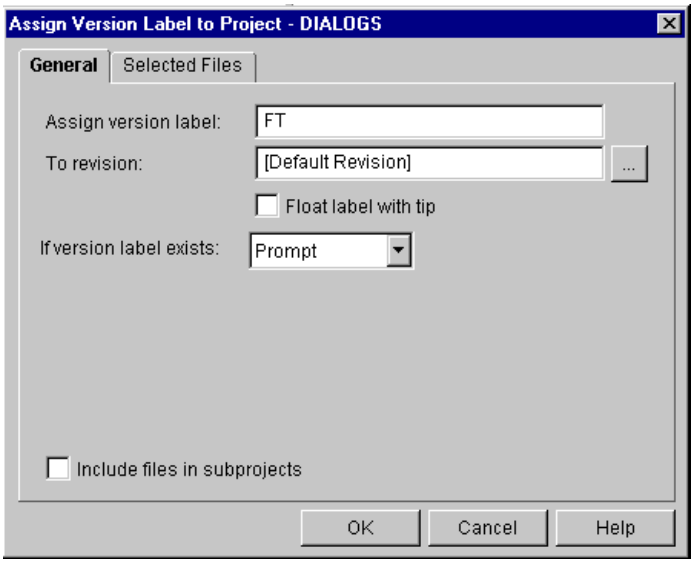
When you reassign version labels at the project, folder, or project-database level, all of the version labels that match the version label you are reassigning are changed.

Moving Existing Version Labels

To move an existing version label within the same versioned file:

1 Select the version label or the item that contains the version label you want to move.

2 Select Actions | Version Label | Assign. The Assign Version Label dialog box appears.



3 Enter the version label you want to move in the **Assign version label** field.

IMPORTANT! Version labels are case-sensitive. Make sure that the version label you entered matches the case of the existing version label that you are reassigning.

4 To move the version label to a revision other than the default revision, enter the revision number, version label, or promotion group in the **To revision** field, or click the Browse button to select the revision.

To define the default revision for your workspace, see "Defining the Default Version" on page 162.

5 If the revision you selected in the **To revision** field is the latest (tip) revision, and you want to keep the version label on the tip as subsequent workfiles are checked in, select the **Float label with tip** check box.

6 Select **Reassign** in the **If version label exists** drop-down menu.

7 (For projects and project databases only). To move version labels for versioned files located in subprojects, select the **Include files in subprojects** check box.

8 Click **OK**.

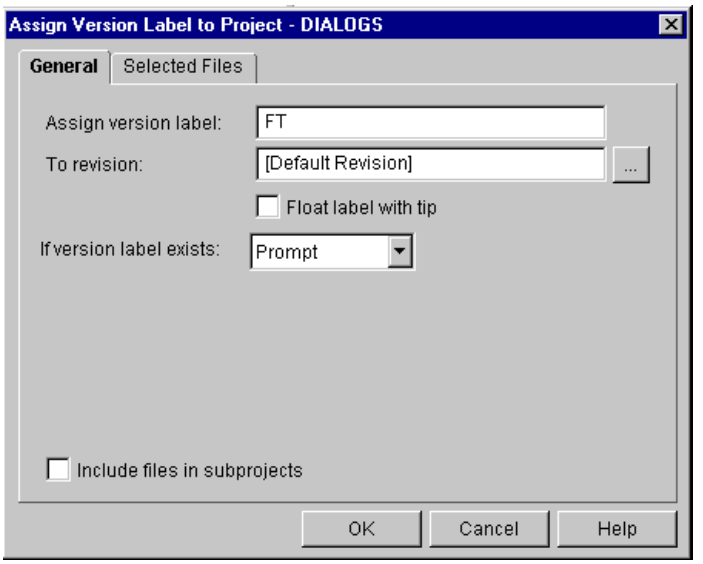
Changing Version Label Properties

You can change a fixed version label to a floating version label (or vice versa) by changing the version label properties.

To change the properties of a version label:

1 Select the version label or the item that contains the version label you want to change.

2 Select Actions | Version Label | Assign. The Assign Version Label dialog box appears.



3 Enter the version label you want to change in the **Assign version label** field.

IMPORTANT! Version labels are case-sensitive. Make sure that the version label you entered matches the case of the existing version label that you are changing.

4 Enter the *same* version label you entered in the **Assign version label** field in the **To revision** field.

5 If you want to change the label to a *floating* label, select the **Float label with tip** check box; otherwise, leave it unchecked to change the label to a *fixed* label.

6 Select **Reassign** in the **If version label exists** drop-down menu.

7 (For projects and project databases only). To change version label properties for versioned files located in subprojects, select the **Include files in subprojects** check box.

8 Click **OK**.

Setting a Version Label as the Default Version

When you check out, get, or promote revisions, the default revision set for your current workspace is used unless you specify a different revision. By default, the default revision is the latest revision (also known as the tip) of a versioned file; however, you can specify a version label as the default revision.

By specifying a version label as the default revision, Version Manager uses the version label as the default value for checking out, getting, and promoting revisions.

To define a version label as the default revision for a workspace, see "Defining the Default Version" on page 162.

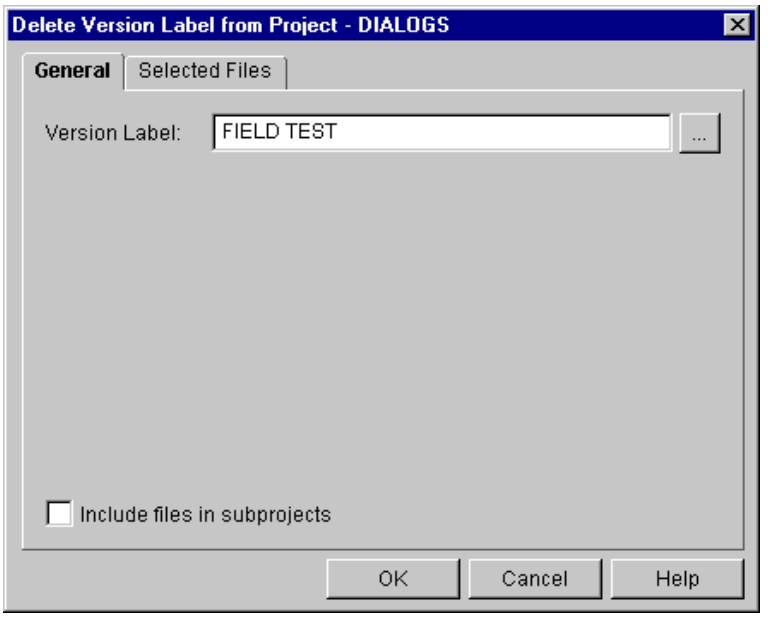
Deleting Version Labels

You can delete version labels when they are no longer needed. You can delete version labels by selecting a version label, a revision, one or more versioned files, a project, a 5.3/6.0 folder, or an entire project database.

When you delete version labels at the project, folder, or project-database level, all of the version labels that match the version label you want to delete will be deleted.

To delete a version label:

1. Select the version label or the item that contains the version label you want to delete.
2. Select Actions | Version Label | Delete. The Delete Version Label dialog box appears.



3 Enter the version label that you want to delete in the **Version Label** field, or click the Browse button to select a version label.

IMPORTANT! Version labels are case-sensitive. Make sure that the version label you entered matches the case of the existing version label that you are deleting.

4 (For projects and project databases only). To delete version labels located in subprojects, select the **Include files in subprojects** check box.

5 Click **OK**.

Scenario: Redefining, Renaming, and Deleting Version Labels

The Bridge project is just about ready for testing. Becky is the Project Leader for this Development team. To prepare these files for promotion to the QA_Testing promotion group, Becky wants to assign a FT, or Field Test, label to the files. She wants this label to stay with the most recent revision of the files until the Developers have completed their work.

Becky clicks the Bridge project and selects Assign | Version Label | Rename. She enters FT as the label name and selects the **Float label with tip** check box to keep the version label with the most recent, or *tip*, revisions, and clicks **OK**.

The Developers continue their work on the Bridge project. When they complete their changes for field test, they communicate this milestone to Becky. Becky displays the Revisions pane and clicks the Version Labels tab. She verifies that the FT version label is associated with the tip revisions.

Justin, another Developer, sets his current workspace to Build. This workspace specifies the custom workfile location for the build process. He checks out the Bridge files based on the FT version label into the Build directory, `z:\build`. To do so, he selects the Check Out option and enters FT in the Revision field. The Check Out To location reflects the workfile location for the build process, `z:\build`. When he clicks **OK**, Version Manager copies writable versions of the files into the build directory.

Becky compiles the code and completes a feature check. She makes some minor modifications to the files and is then ready to pass the code to QA. She wants to rename the version label to Field Test, so she clicks the Bridge project and selects Actions | Version Label | Rename. She enters FT in the Rename From field and enters Field Test in the To field. She checks the Include files in subprojects option and clicks **OK**.

She must now check in the files but also wants to change the Field Test version label from floating to fixed. She selects the Bridge project and selects the Check In option. On the General tab, she enters Ready for Field Test QA in the description field. On the Advanced tab, she enters Field Test in the Version Label field and makes sure that the **Float label with tip** check box is not selected.

Version Manager creates new revisions of the Bridge files. The revisions are now associated with the fixed label, Field Test. The Quality Assurance team can now check out the Bridge files based on the Field Test label and test the code while the Bridge Developers continue with their work for the final release of the product. As the Developers

create new revisions of the files, the Field Test label remains associated with these revisions.

Meanwhile, Mark, the Documentation Team Leader, learns that the product specifications for the Chess project are updated. He needs to unlock the dialog box help files and remove the On Hold version label. To do so, Mark selects the Unlock option and enters MarkP in the Unlock Locks By User field. He selects the Include files in subprojects check box and then clicks **OK**. The files no longer display with a lock icon in the Files pane. Next, Mark selects the Chess project selects Actions | Version Label | Delete. He enters On Hold in the Version Label field and selects the Include files in subprojects check box. When he clicks **OK**, the version label is no longer associated with the dialog box help files. The Writers can now check out the topics and use the updated specifications to continue working.

Part 3

Performing Advanced Tasks with Version Manager Desktop Client

Chapter 15

Branching Revisions

About Branching

What is branching?

Branching is a separate line of development consisting of one or more revisions that diverge from a revision on the trunk (main line) or from another development branch.

Parallel development

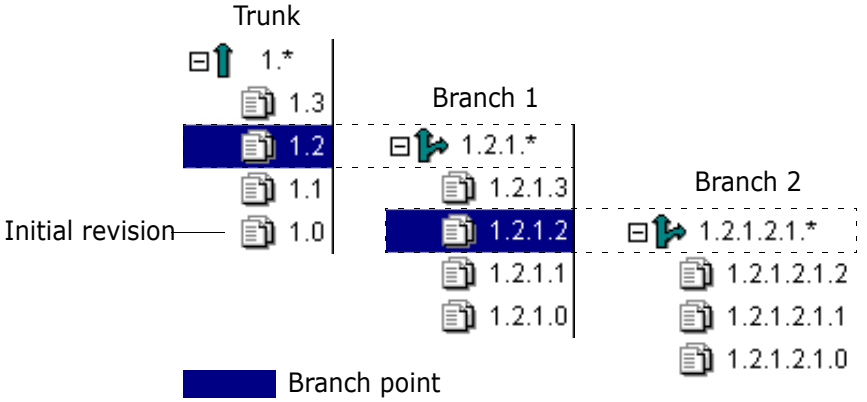
Branching lets you develop alternate versions of a file in *parallel* with other Project Team Members who are working on the trunk or on another branch. Some common reasons for branching include:

- Developing a version of a file for a different platform. For example, if you have several archives that store source code files for a Windows application, you can start branches in each archive to develop an alternate version of the application for UNIX.
- Fixing defects without interrupting development on the trunk. If you discover a defect, but want development to continue in other areas of the source code file, you can create a branch from the revision containing the defect, fix it, and test it without impeding progress on trunk development. You can later merge the fix in with the newest revision on the trunk.
- Creating a base product and then customizing the product for major customers.

Branching lets a number of developers continue parallel development on different revisions of the same file; it is also possible for one developer to work on both trunk development and various branches.

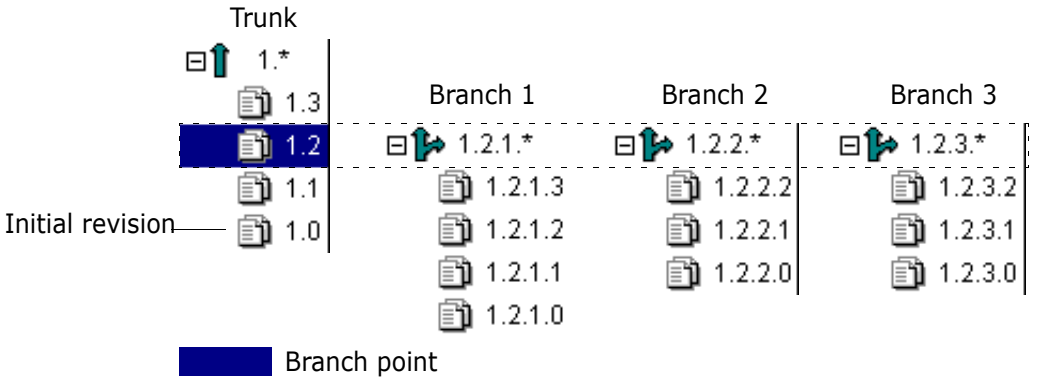
Branch Numbering

Branches diverge from the trunk or from other branches. The revision from which a branch begins is called the branch point (also known as the base version). The first branch revision from the branch point carries the entire branch point revision number followed by 1.0. Version Manager increments each subsequent branch revision by 0.1.



For example, in the figure above, revision 1.2 is the branch point, so the first branch revision is numbered 1.2.1.0. As with trunk revisions, Version Manager increments each new branch revision by 0.1, so subsequent revisions on this branch would be 1.2.1.1, 1.2.1.2, etc. Additionally, revision 1.2.1.2 is another branch point, so the first branch revision is numbered 1.2.1.2.1.0. Subsequent revisions on this branch would be 1.2.1.2.1.1, 1.2.1.2.1.2, etc. Version Manager identifies a branch by its branch point revision number followed by 1.*.

However, if you create multiple branches from the same revision, Version Manager increments each new branch revision by 1.0. For example, in the following example, if you use revision 1.2 to create multiple branches, the first branch will be 1.2.1.*, the second branch will be 1.2.2.*, and the third branch will be 1.2.3.*.



Creating a Branch

Before you begin…

Before you create a branch, you should display the Revision pane (View | Show Revisions). The Revision pane is the only pane from the main window that displays branch information.

When Branches Are Created

Branches are created when you:

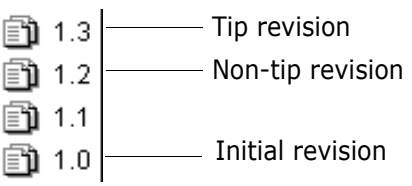
- Check in a non-tip revision
- Force a branch when checking in a tip revision
- Check in a revision with multiple locks
- Configure Version Manager for automatic branching

Checking In a Non-tip Revision

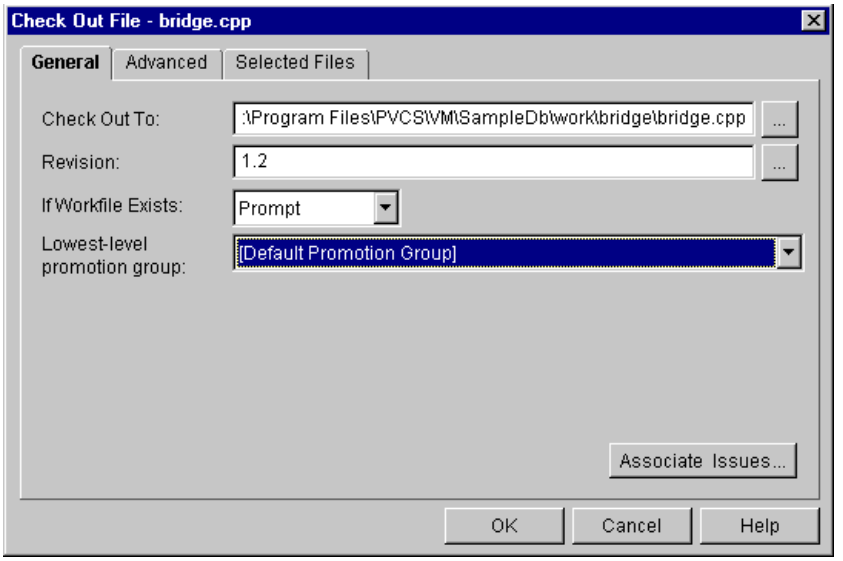
You create a branch when you check in a non-tip revision.

To create a branch by checking in a non-tip revision:

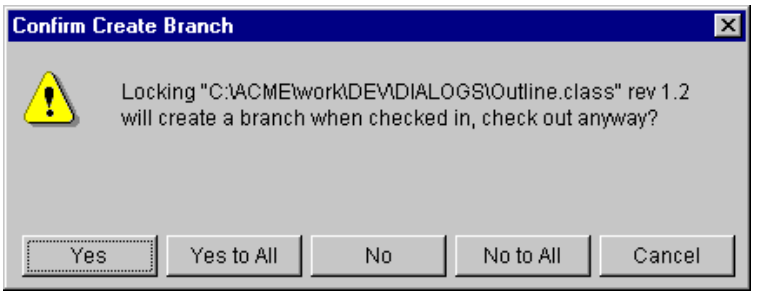
1 From the Revision pane, select a non-tip revision that you want to use as the starting point of the branch.



2 Select Actions | Check Out. The Check Out File dialog box appears.

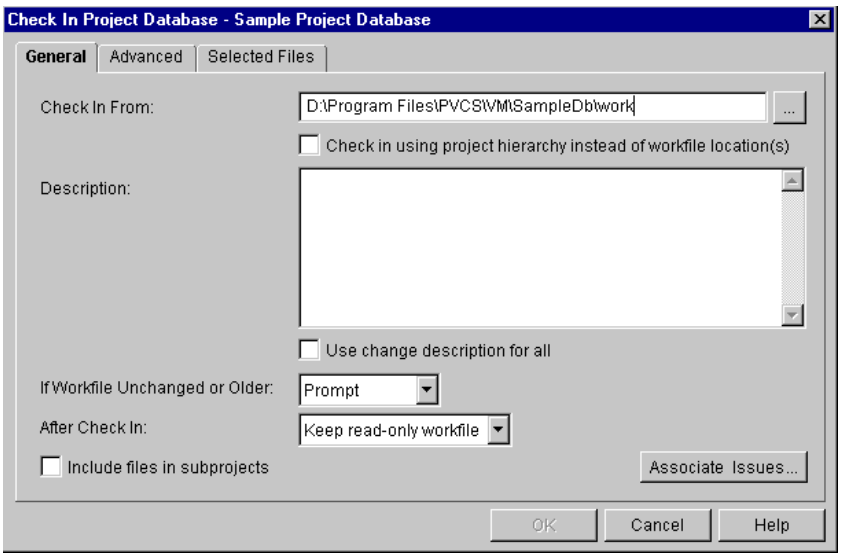


3 Click **OK**. The Confirm Create Branch message appears.

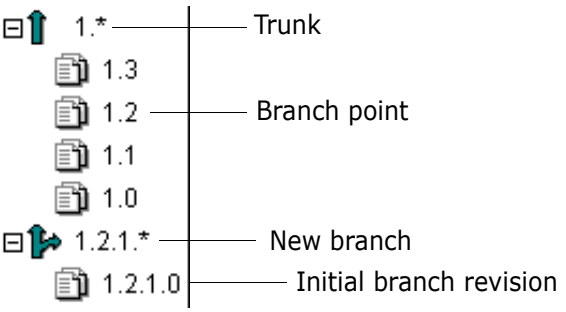


4 Click **Yes**. Version Manager checks out the revision to the location currently set as the workfile location and places a lock on the revision.

5 When you are finished editing the workfile, check the workfile back in to the archive by selecting Actions | Check In. The Check In File dialog box appears.



6 Enter a change description in the **Description** field and click **OK**. Version Manager creates a branch, as illustrated by the branch icon ( ), assigns it the branch point revision number followed by 1.* , and assigns the initial branch revision the branch point revision number followed by 1.0.

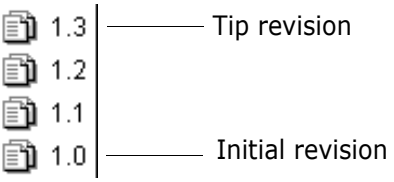


Forcing a Branch

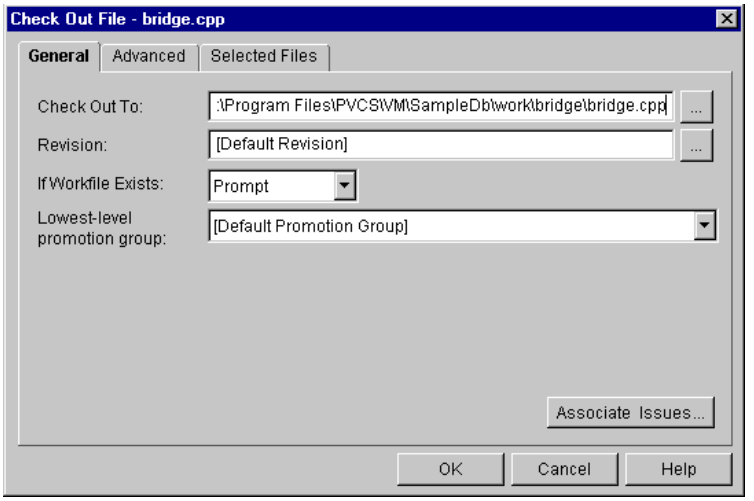
You can create a branch at the latest (tip) revision by forcing a branch. To create a branch from the tip revision, you must select the **Force branch** option when checking the workfile back in.

To force the creation of a branch at the tip revision:

1 From the Revision pane, select a tip revision that you want to use as the starting point of the branch.

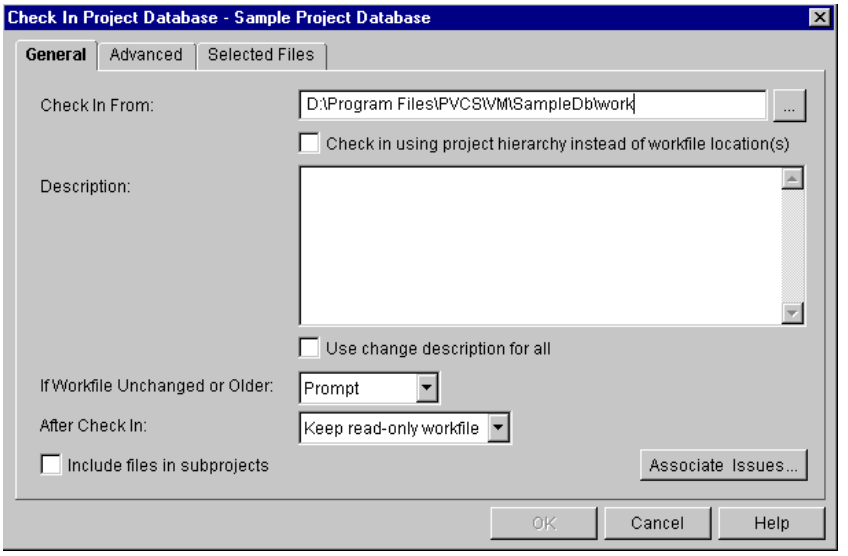


2 Select Actions | Check Out. The Check Out File dialog box appears.



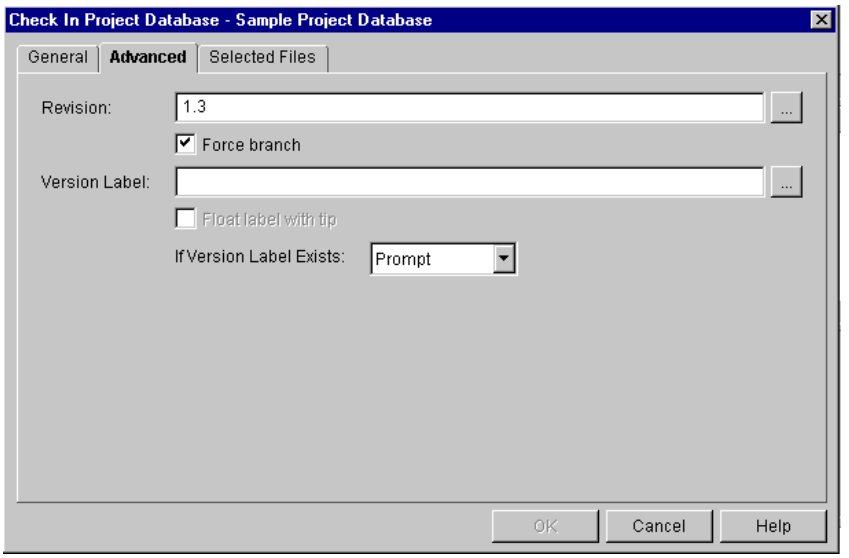
3 Click **OK**. The tip revision is checked out and the revision is locked.

4 When you are finished editing the workfile, check in the workfile by selecting Actions | Check In. The Check In File dialog box appears.

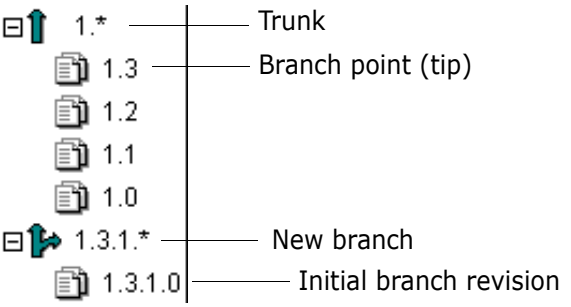


5 Enter a change description in the **Description** field.

6 Under the Advanced tab, select the **Force branch** check box.



7 Click **OK**. Version Manager creates a branch, as illustrated by the branch icon ( ), assigns it the branch point revision number followed by 1.*, and assigns the initial branch revision the branch point revision number followed by 1.0.



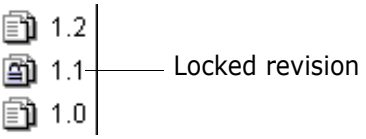
Checking In a Revision with Multiple Locks

Before you begin…

To check in a revision with multiple locks, you must enable your project/project database for multiple locking. See the "Using Multiple Locks for Branching" section in the *Serena PVCS Version Manager Administrator's Guide*.

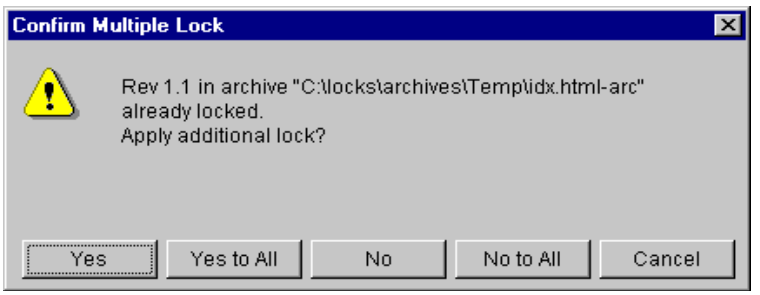
To create a branch when multiple locking is enabled:

1. From the Revision pane, select a locked revision.

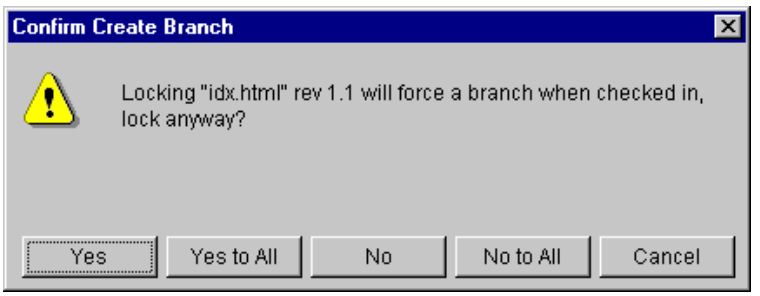


2. To apply a secondary lock, select Actions | Check Out or Actions | Lock. The Check Out File or Lock File dialog box appears.

3. Click **OK**. The Confirm Multiple Lock confirmation dialog box warns that the revision is already locked and asks if you want to apply an additional lock.



4. Click **Yes**. The Confirm Create Branch dialog box warns that locking the selected revision will force a branch when the revision is checked in and asks if you wish to lock or check out the file anyway.

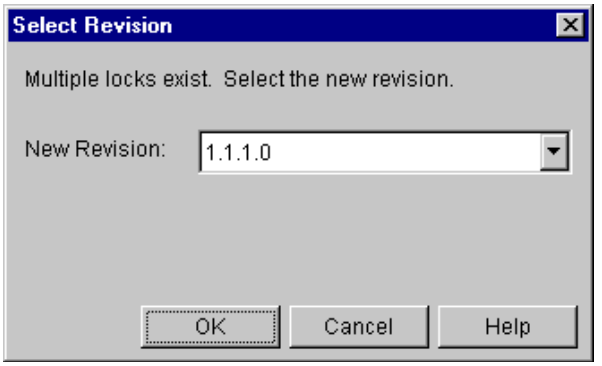


5. Click **Yes**. Version Manager checks out or locks the revision.

6. Modify the workfile as necessary and save your changes.

7. Select Actions | Check In. The Check In File dialog box appears.

8 Enter a change description in the **Description** field and click **OK**. Because you are checking in a workfile with multiple locks, the Select Revision dialog box appears.



9 From the **New Revision** drop-down menu, select the revision number that will create a new branch when the workfile is checked in.

10 Click **OK**. Version Manager creates a branch from the revision you selected and displays the branch, as illustrated by the branch icon ( ), in the Revision pane.

Setting Up Automatic Branching

You can set up Version Manager to allow for automatic branching. Automatic branching lets you create a branch automatically from a trunk revision. Then, by default, you work with the tip revision of that branch whenever you perform an action.

Without automatic branching, you must create a branch manually, either by checking in a non-tip revision, forcing a branch when checking in a tip revision, or checking in a revision with multiple locks.

For information on how to set up Version Manager for automatic branching, see the "Automatic Branching" section in the *Serena PVCS Version Manager Administrator's Guide*.

Scenario: Fixing Bugs Without Disrupting the Main Line of Development

Terri has completed her review of the Chess project files and identified the files that she needs to modify to fix the chessboard bugs. All of these files correspond to the tip, or latest, revisions for the Board subproject. She selects the Board subproject and selects the Check Out option. She maintains the system defaults and clicks **OK**. At the end of each day, she is required to check her files back in to version control; however, she is not ready to incorporate her changes into the main line of Chess development just yet.

Instead, she would like to work with these files for a few more days and make sure that her changes are sound before she makes them part of the main Chess development effort. She wants to create a separate branch of development, but because the revisions she checked out were tip revisions, she must force a branch off of the main development trunk to keep her work separate for the time being.

From the Check In dialog box, Terri enters a change description. She also selects the Advanced tab and chooses the Force branch check box so that new development branches

will be created when she checks in her workfiles. She also associates the floating label, "Chessboard Bug Fix" with the project files. This label will allow Terri to easily identify and work with the revisions. When Terri clicks **OK**, Version Manager checks in the workfiles as branch revisions of the main development trunk. Version Manager also associates the new version label with these files.

To continue work on the chessboard branch, Terri checks out the Board files based on the "Chessboard Bug Fix" version label in the Default Revision field. Each time Terri checks in her files, this version label moves to the newest branch revision of each file.

Chapter 16

Comparing Files

About Comparing Files

Side-by-side comparisons

Serena PVCS Version Manager allows you to select two text files and compare them side-by-side to see the additions, deletions, and changes made to the files. You cannot compare binary files.

Compare revisions and workfiles

The Show Differences option (Actions | Show Differences) allows you to compare:

- A revision and a workfile
- Revisions in a single versioned file
- Revisions in two different versioned files
- Two workfiles

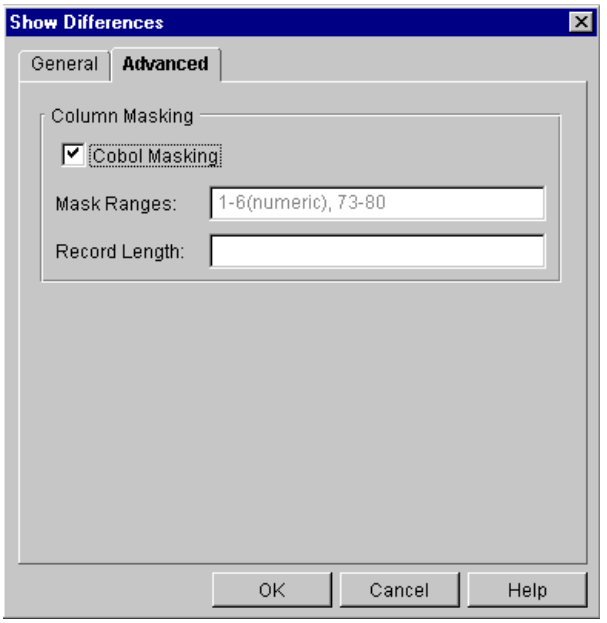
View only

The Show Differences option provides a quick comparison of two files. You cannot print or save the comparison file.

Configuring Column Masking in Windows

To enable column masking:

1 Select the Advanced tab.



NOTE The values set in the Show Differences | Advanced tab apply only to the current session. These values are not stored in the project configuration file nor the archive file.

2 Select the **Cobol Masking** check box to use the default Cobol masking definition for Cobol files.

3 The default value is automatically displayed in the **Mask Ranges** field. The default is mask columns 73-80 and mask columns 1 through 6 only if these columns are numeric. You may override this value by entering a new value.

4 Enter the length of records in fixed-length files so that Version Manager can generate deltas based on logical lines. Do not enter a value in this field for variable length files. Doing so could cause the output to be invalid or incorrect.

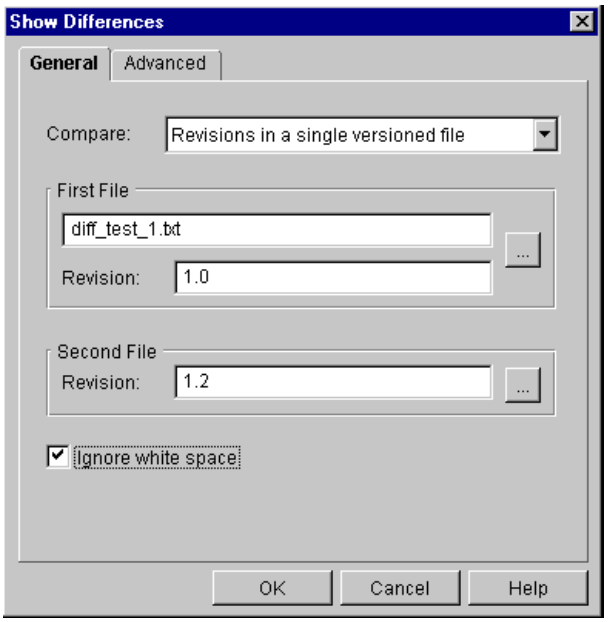
Viewing Differences

To view the differences between files:

1 Select the versioned file(s) or revision(s) that you want to compare. You can select any text files.

NOTE To compare revisions located in different projects (but within the same project database), select the project database. You cannot compare revisions located in different project databases.

2 Select Actions | Show Differences. The Show Differences dialog box appears.



NOTE The appearance of the Show Differences dialog box varies depending on the type of comparison you select.

3 Select the type of comparison that you want to perform from the **Compare** drop-down menu.

To compare...	You must specify...
A revision and a workfile	A versioned file and revision number, and a workfile
Revisions in a single versioned file	A versioned file and revision number, and a second revision number
Revisions in different versioned files	Two versioned files and two revision numbers
Workfiles	Two workfiles

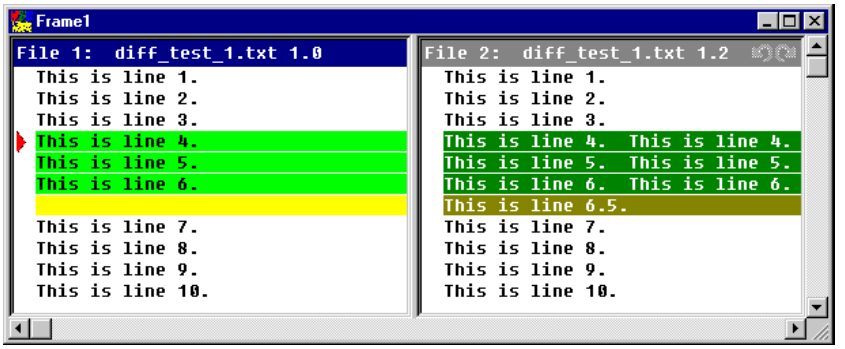
4 To change the revision (or file, if comparing workfiles) selected in the **First File** group, click the Browse button.

NOTE You can select a revision by version label, revision number, or promotion group.

5 To change the revision (or file, if comparing workfiles) selected in the **Second File** group, click the Browse button.

6 Select the **Ignore white space** check box if you want to ignore trailing, intervening, and leading white spaces, tabs, and form feeds.

7 Click **OK**. The Merant Merge Tool is launched with the two files side-by-side in separate panes.



8 To view the differences, scroll through the files and compare the colored text blocks.

Windows users can also click the Next Difference button (≠>) to jump directly to the next difference.

Interpreting Difference Results

Types of differences

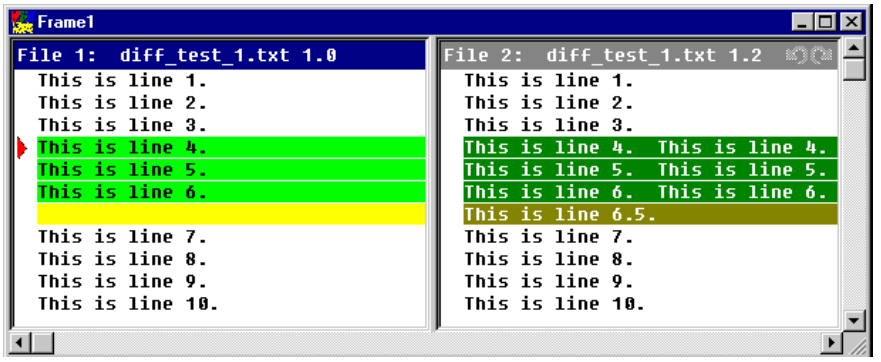
File 1 is used as the base file. File 2 is compared to File 1. Differences between the files are categorized into additions, deletions, and changes:

- **Additions** are the lines of text that were added to File 2. These lines of text are not in File 1.

- **Deletions** are the lines of text that were deleted from File 2. These lines of text are in File 1.
- **Changes** are the lines of text that were modified in one of the files.

Placeholders

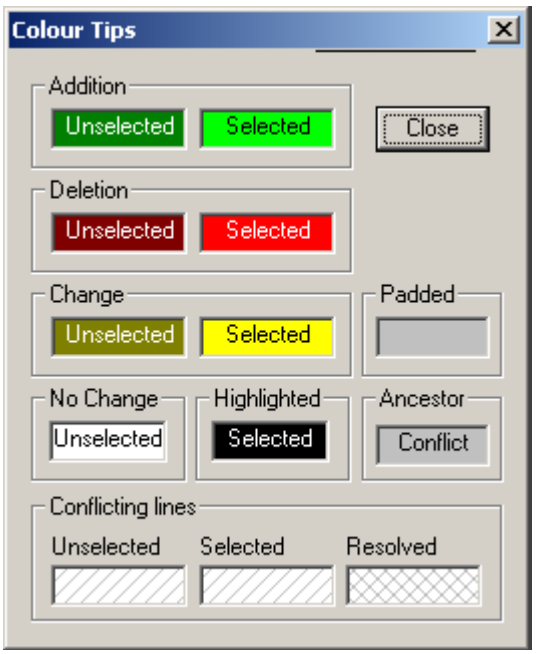
Placeholders are the blocks of colors used to identify the types of differences between the files. Each difference type (additions, deletions, and changes) has two placeholders (or colors) assigned to it: one for selected one for unselected.



Windows users

Windows users have the option of changing the colors of the placeholders. To change the colors, from the Merant Merge Tool, click the Configuration button ( ) and select the Colors tab.

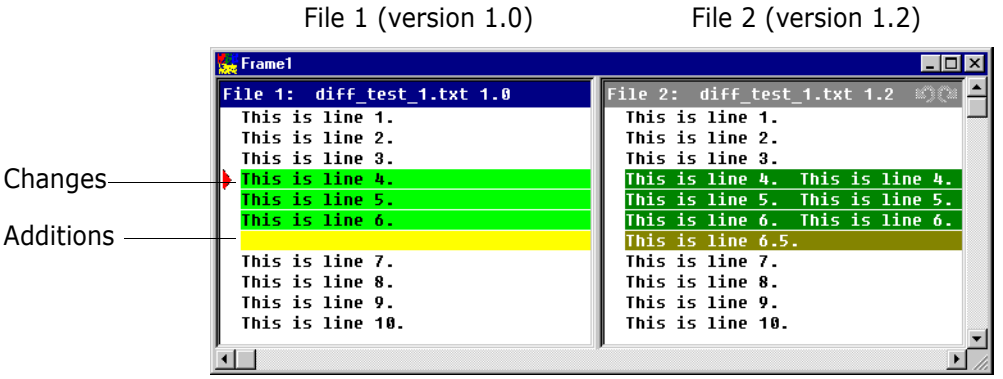
Windows users also have the option of displaying a color legend to help identify what each placeholder stands for. To display the color legend, from the Merant Merge Tool, click the Color Tips button ( ). The Color Tips dialog appears.



Difference Examples

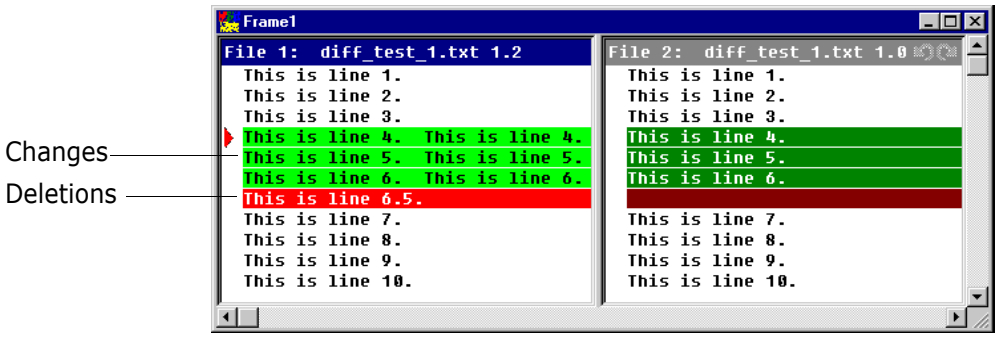
The following examples show how the differences between two files can vary depending on the file you select as File 1 (the base file).

Example 1 This example shows two revisions from the same versioned file. Version 1.0 was selected as File 1 (base file) and version 1.2 was selected as File 2.



Lines 4, 5, and 6 have been changed and are highlighted as one block of differences. These differences are identified by change placeholders. Line 6.5 was added to File 2 and is highlighted as another block of differences. These differences are identified by addition placeholders.

Example 2 Using the same files as above but reversing the order of the files, you get the following results.



Lines 4, 5, and 6 are still highlighted as differences and identified by change placeholders. However, the added line, line 6.5, is now in File 1 (the base file) but no longer in File 2. This difference is identified by deletion placeholders.

Chapter 17
Merging Files

About Merging

What is merging?

Merging is the process of comparing the differences between two (or more) text files, or revisions of text files with a common base file, accepting or rejecting the differences between them, and combining the changes into a new text file.

You cannot compare or merge binary files.

Why merge files?

Merging is useful for parallel development. It enables you to combine the work performed on a branch of development and integrate it back into the main line of development.

Types of merges

Serena PVCS Version Manager allows you to perform a variety of merges. You can merge:

- A revision with a workfile
- Revisions within a single versioned file
- Revisions in different versioned files
- Workfiles

Windows users

Version Manager for Windows can perform N-way (unlimited) merges. Although you can select only three files (one base and two branch files) from within Version Manager, you can perform an N-way merge once the Merant Merge Tool is launched. For information on how to perform N-way merges, please see the Merant Merge Tool online help.

UNIX users

Version Manager for UNIX can perform two-way (a base file and a branch file) merges. N-way merges are not supported on UNIX.

Merging Terms and Definitions

The following table is a list of basic merge terms and definitions:

Term	Definition
Base or Ancestor	The base file from which other files are derived.
Branch or Derivative	The result obtained by making changes in the base file.
Output or Target	The final merged file.
Addition	A line of text added to a branch file. This line of text is not in the base file.
Deletion	A line of text deleted from a branch file. This line of text is in the base file.
Change	A line of text that was modified in one of the branch files. The content of this line of text differs between the branch file and the base file.
Conflict (Windows only)	A line of text that was modified in multiple branch files. The content of this line of text differs between the branch files and the base file.

NOTE In UNIX, *changes* are treated as *conflicts* and must be resolved by the user.

The Merge Process

The merge process consists of:

- Locking the tip of the branch or trunk where you want to check in the output (target) file
- Selecting a base file (ancestor) as a reference point
- Selecting a revision branch file (derivative)
- Entering an output (target) file name
- Resolving conflicts between the base and branch files by either accepting or rejecting the changes made
- Saving the changes made to the output (target) file
- Checking in the output file

Selecting a Base File

A base file is the file that you want to use as your point of reference. It is usually the branch point—the revision from which a branch begins (typically off the main trunk of development).

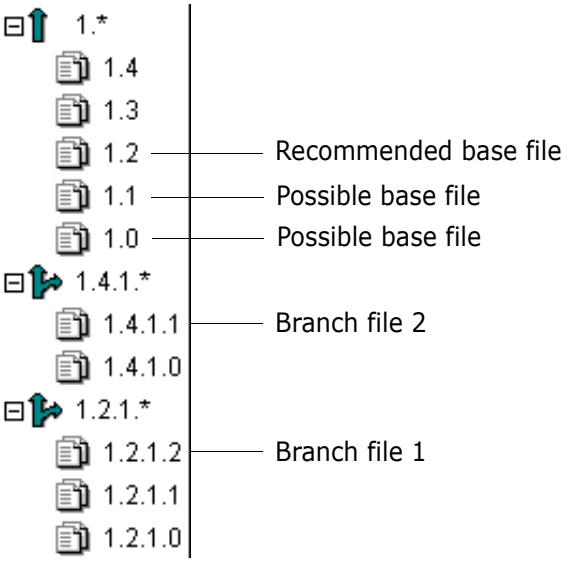
If you are merging revisions that are located in different branches or versioned files, the base file you select should be the revision or file that is the latest file to be created which is a common ancestor to *all* of the revisions that you want to merge.

IMPORTANT! UNIX Only:

If you are merging or comparing revisions on a branch to revisions on a trunk on UNIX, you should set the base file to the latest revision on the trunk, then select the latest revision on the branch as the branch file.

In the following example, if you want to merge the changes from revision 1.2.1.2 with revision 1.4.1.1, you would use revision 1.2 as the base file. You could use revision 1.1 or 1.0 as the base file, but we recommend using the revision that is closest to the files you

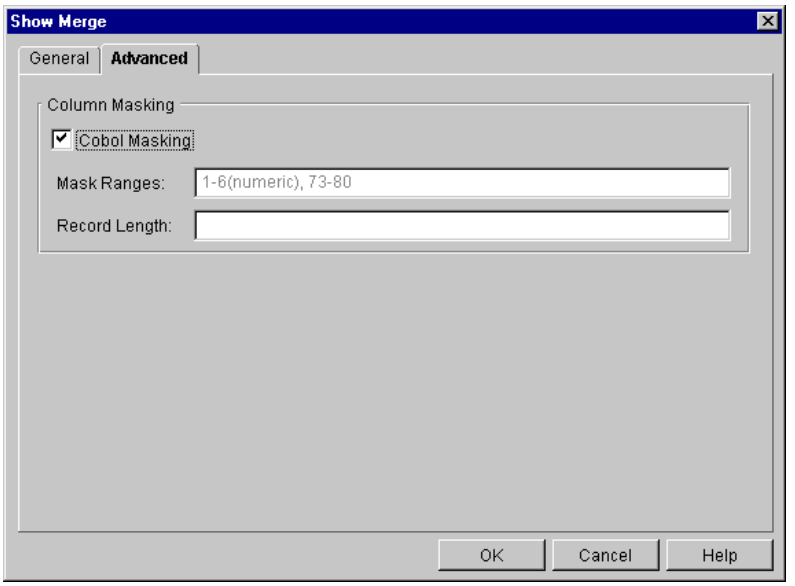
are merging. The closer the content of the base file is to that of the files you are merging, the fewer conflicts you will have to resolve.



Configuring Column Masking in Windows

To enable column masking:

1 Select the Advanced tab.



NOTE The values set in the Show Merge | Advanced tab apply only to the current merge session. These values are not stored in the project configuration file nor the archive file.

2 Select the **Cobol Masking** check box to use the default Cobol masking definition for Cobol files.

3 The default value is automatically displayed in the Mask Ranges field. The default is mask columns 73-80 and mask columns 1 through 6 only if these columns are numeric. You may override this value by entering a new value.

4 Enter the length of records in fixed-length files so that Version Manager can generate deltas based on logical lines. Do not enter a value in this field for variable length files. Doing so could cause the output of the Merge Tool to be invalid or incorrect.

Merging Files in Windows or UNIX

To merge files:

1 Lock the tip of the branch or trunk where you want to add the results of the merge.

2 Select the versioned file(s) or revision(s) that you want to merge.

When selecting files to merge, consider the following:

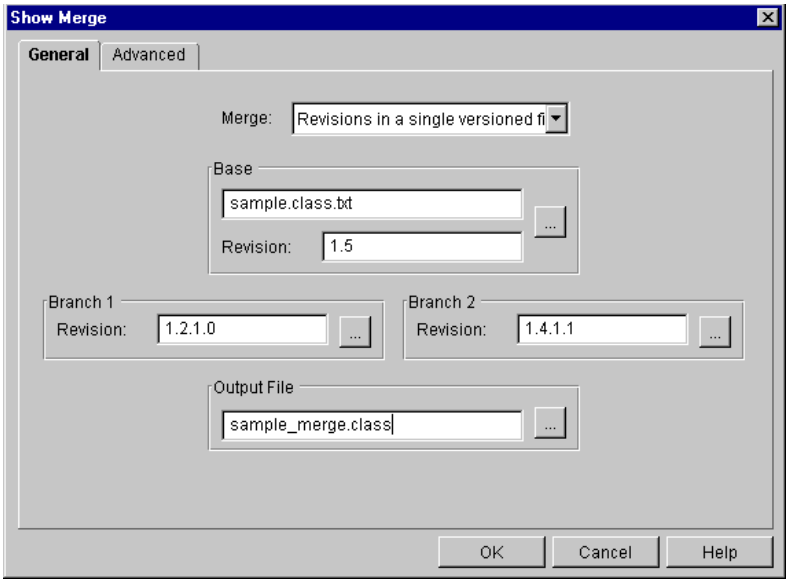
- The order in which you select the files determines which file will be the base and which the branch file or files. The first file that you select is the base file; the second and (on Windows only) third files that you select are the branch files.
- The items you select (project, versioned file, or revision) affect the initial options that appear.

NOTE You can change the options when the Show Merge dialog box appears.

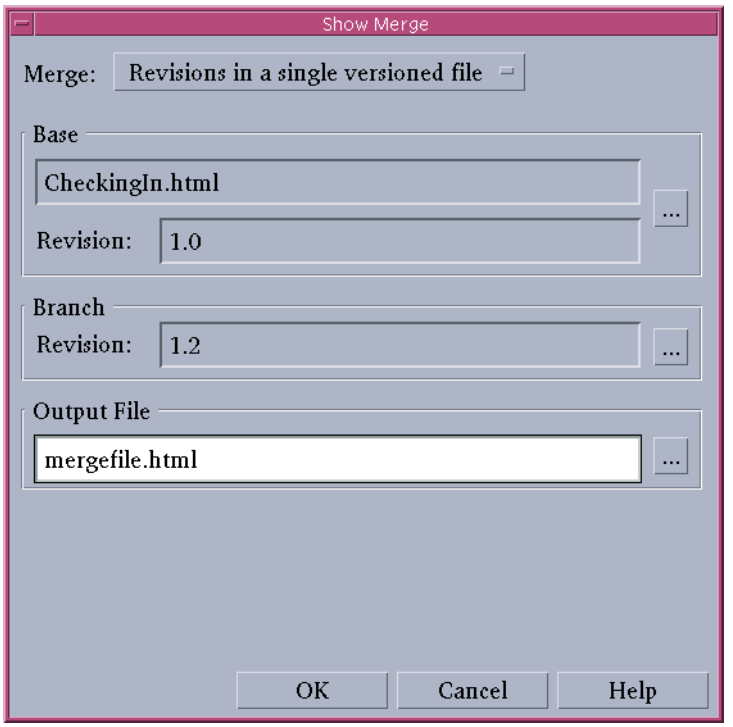
- (UNIX only) To merge a trunk and branch revision, you must set the base file to the tip revision on the trunk, and the branch file to the tip revision on the branch.
- To merge revisions located in different projects (but within the same project database), select the project database. You cannot merge revisions located in different project databases.
- (Windows only) If you want to merge just two files, enter the same file in the **Base** and **Branch 1** fields; use the **Branch 2** field for the second file.

3 Select Actions | Show Merge. The Show Merge dialog box appears.

Windows dialog



UNIX dialog



NOTE The appearance of the Show Merge dialog box will vary depending on the files you select and the type of merge you want.

4 Select the type of merge that you want to perform from the **Merge** drop-down menu.

To merge...	On Windows you must specify...	On UNIX you must specify...
A revision and a workfile	A base file and revision number, a branch revision number, and a branch workfile	A base file and revision number, and a branch workfile
Revisions in a single versioned file	A base file and revision number, and two branch revision numbers	A base file and revision number, and a branch revision number
Revisions in different versioned files	A base file and revision number, and two branch files with revision numbers	A base file and revision number, and a branch file and revision number
Workfiles	A base workfile and two workfiles	A base workfile and a branch workfile

5 To change the revision (or file, if merging workfiles) selected in the **Base** group, click the Browse button.

NOTE UNIX Only:

To merge a trunk and branch revision on UNIX, you must set the base file to the tip revision on the trunk.

6 To change the revision (or file, if merging workfiles) selected in the **Branch 1** (Windows) or **Branch** (UNIX) group, click the Browse button.

NOTE UNIX Only:

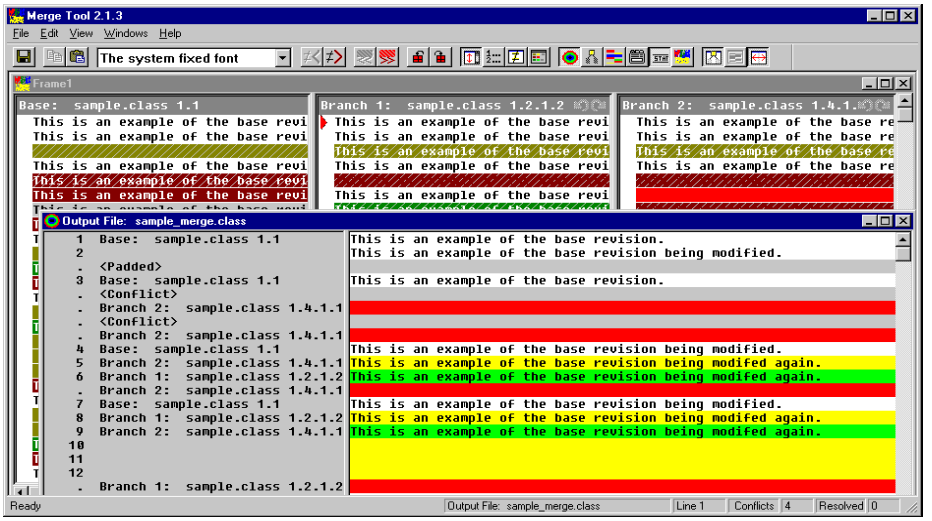
To merge a trunk and branch revision on UNIX, you must set the branch file to the tip revision on the branch.

7 (Windows only) To change the revision (or file, if merging workfiles) in the **Branch 2** group, click the Browse button.

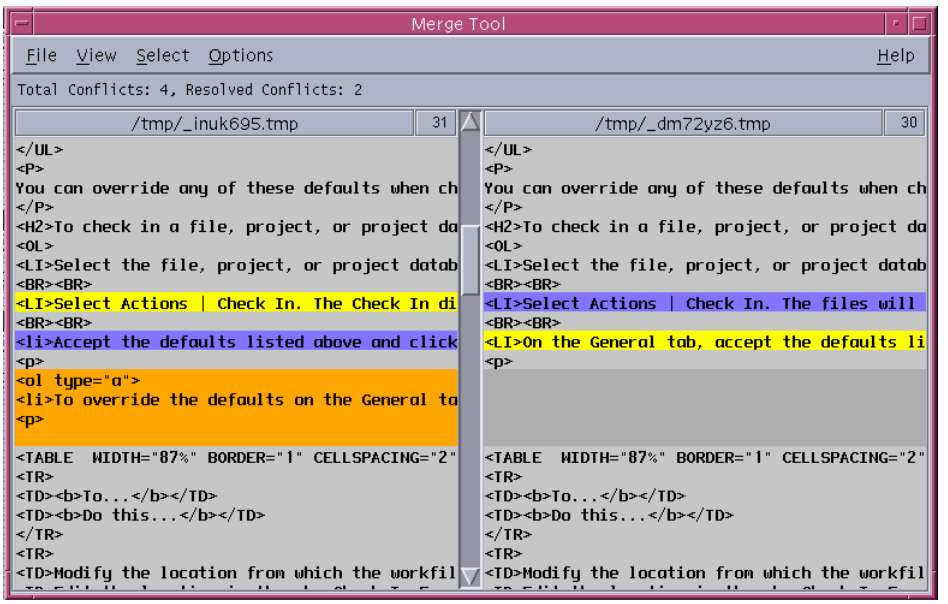
8 Enter the name of the output (target) file in the **Output File** field, or click the Browse button to select an existing workfile.

NOTE If you enter an output filename without entering a path, the output file will be created in the BIN directory within your Version Manager installation directory. If the filename already exists in the selected location, the old file will be overwritten.

9 Click **OK**. On Windows, the base file and branch files appear in the Frames window and the results of the merge appear in the Output File window.



On UNIX, the base file and branch file appear in the Merge Tool window.



10 Resolve the conflicts between the base and branch files. For information on how to resolve the conflicts, see one of the following:

- "Resolving Conflicts Between Files in Windows" on page 224
- "Resolving Conflicts Between Files in UNIX" on page 225

11 To save the results of the merge to an output file, do one of the following:

- On Windows, select File | Save Target.
- On UNIX, select File | Save As and click **OK**.

NOTE You do not need to re-enter the output path or filename.

12 Open the output file and review it. If you are satisfied with the results of the merge, check the output file in to the tip of the branch or trunk you selected in Step 1.

Interpreting Difference Results in Windows

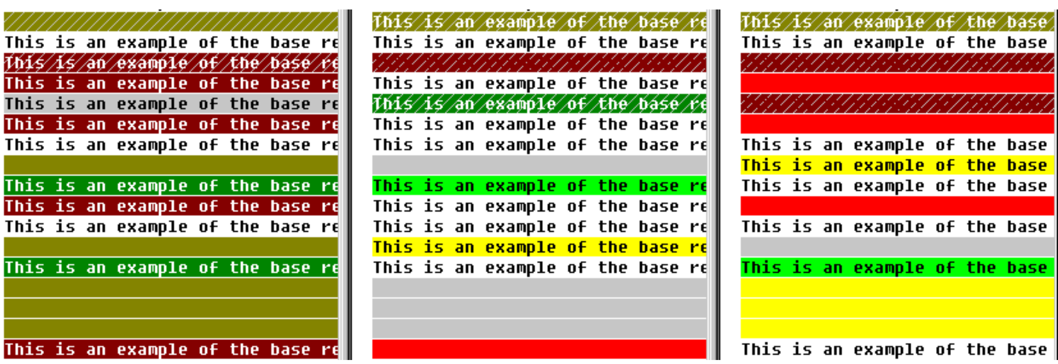
Types of differences

The file that you select as the base file is used as the basis of the output file. Branch 1 and branch 2 are compared against the base file. Differences between the base and branch files are categorized as additions, deletions, and changes:

- **Additions** are the lines of text that were added to a branch file. These lines of text are not in the base file.
- **Deletions** are the lines of text that were deleted from a branch file. These lines of text are in the base file.
- **Changes** are the lines of text that were modified in one of the branch files. The content of these lines of text differ between the branch file and the base file.
- **Conflicts** are the lines of text that were modified in multiple branch files. The content of these lines of text differ between the branch files and the base file.

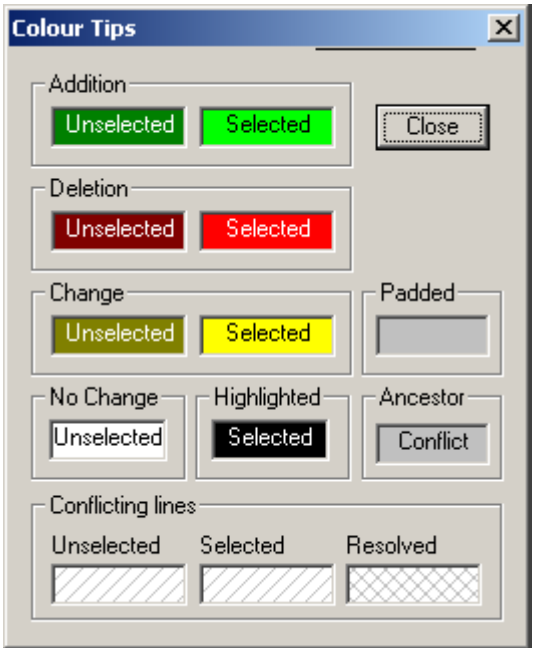
Placeholders

Placeholders are the blocks of colors used to identify the types of differences between the files. Each difference type (additions, deletions, and changes) has two placeholders (or colors) assigned to it: one for selected and one for unselected.



Displaying a color legend

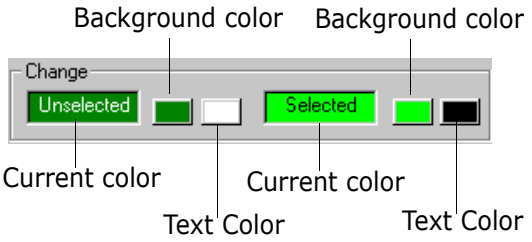
You can display a color legend to help identify what each placeholder stands for. To display the color legend, from the Merant Merge Tool, click the Color Tips button ( ). The Color Tips dialog appears.



Changing Placeholder Colors

To change the colors of the placeholders:

1. Click the Configuration button ( ).
2. Select the Colors tab.
3. Click the background or text color you want to change.

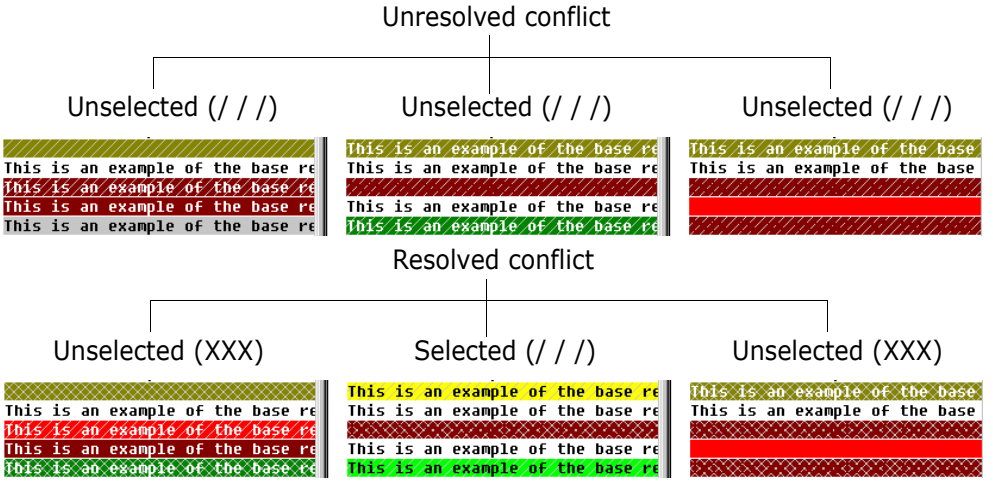


4. Select a new color from the color palette and click **OK**.

Conflicts

Conflicts occur when the content of a line in multiple branch files differ from the content of the line in the base file.

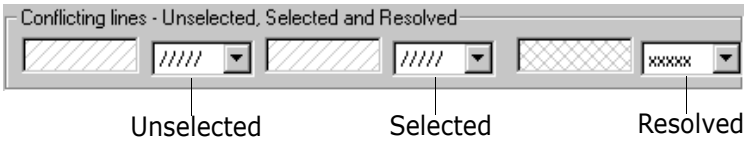
By default, conflicts appear as blocks of text that are shaded with slashes (/ / /). When you resolve a conflict, the slashes remain on the selected block of text while the unselected conflict blocks (which are now resolved) are shaded with Xs (XXX).



Changing Conflict Shading

To change the shading of the conflicts:

1. Click the Configuration button ( ).
2. Select the Colors tab.
3. Choose a shading pattern from the appropriate **Conflicting lines** drop-down menu.



Interpreting Difference Results in UNIX

Types of differences

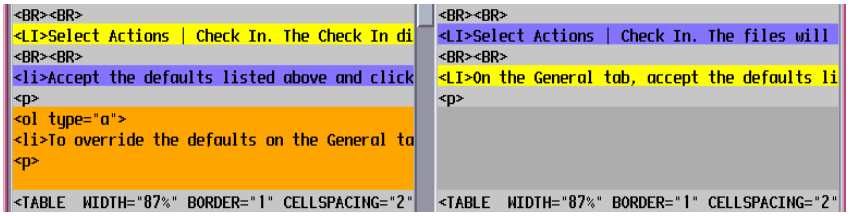
The file that you select as the base file is used as the basis of the output file. The branch file is compared against the base file. Differences between the base and branch files are categorized as additions, deletions, and changes:

- **Additions** are the lines of text that were added to the branch file. These lines of text are not in the base file.
- **Deletions** are the lines of text that were deleted from the branch file. These lines of text are in the base file.
- **Changes** are the lines of text that were modified in the branch file. The content of these lines of text differ between the branch file and the base file.

NOTE In UNIX, *changes* are treated as *conflicts* and must be resolved by the user.

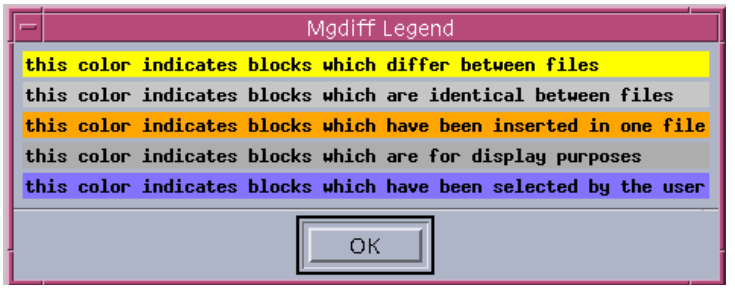
Placeholders

Placeholders are the blocks of colors used to identify the types of differences between the files. Each difference type (additions, deletions, and changes) has two placeholders (or colors) assigned to it: one for selected and one for unselected.



Displaying a color legend

You can display a color legend to help identify what each placeholder stands for. To display the color legend, from the Merant Merge Tool, select Help | Color Legend.



Changing Placeholder Colors

To change the colors of the placeholders, edit the `Mgdiff Legend` file with a text editor. This file is located in the `/bin/app-defaults` directory where you installed Version Manager.

Resolving Conflicts Between Files in Windows

By default, the Serena Merge Tool automatically inserts all of the additions, deletions, and changes into the output (target) file, but not the conflicts. Before you can successfully merge revisions, you must resolve all conflicts.

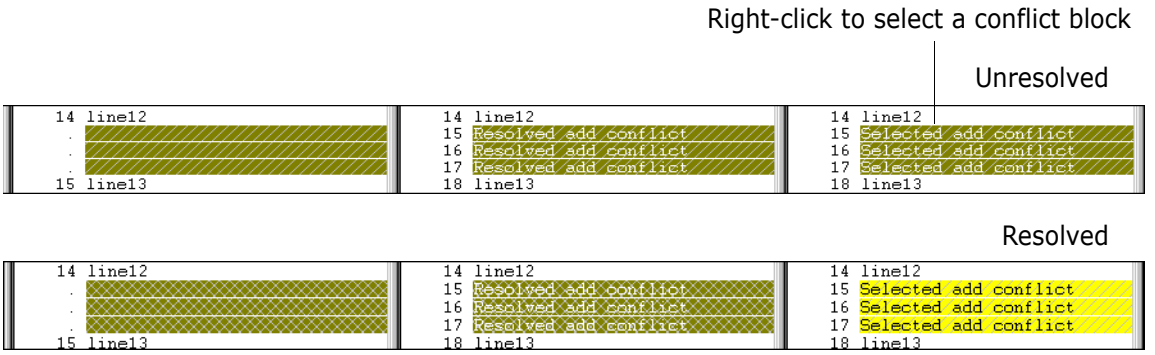
On Windows, a conflict occurs when a line of text differs between multiple branch files and the base file. The Merant Merge Tool cannot determine which block to select, so you must resolve the conflict by choosing which block of text you want in the output file.

NOTE To resolve conflicts, you must be able to identify the differences between the base and branch files. For information on how to identify the additions, deletions, and changes between base and branch files, see "Interpreting Difference Results in Windows" on page 221.

To resolve conflicts between the base and branch files:

1 From the Frames window in the Merant Merge Tool, click the Next Conflict button () to jump to the next conflict block.

2 Resolve conflicts by right-clicking the block of text that you want to appear in the output (target) file. The conflict block that you select changes color and the shading of the unselected files changes to resolved.



3 Repeat steps 1-2 until all conflicts have been resolved.

4 Check the contents of the output (target) file. Make sure that you have merged the appropriate content into the output file.

5 Select File | Save Target to save the output (target) file.

Resolving Conflicts Between Files in UNIX

By default, the Merant Merge Tool automatically inserts all of the additions and deletions into the output (target), but not the changes. Before you can successfully merge revisions, you must resolve all conflicts (changes).

On UNIX, a conflict occurs when a line of text has been modified in the branch file so that the content of this line of text differs between the branch file and the base file. The Merant Merge Tool cannot determine which block to select, so you must resolve the conflict by choosing which block of text you want in the output file.

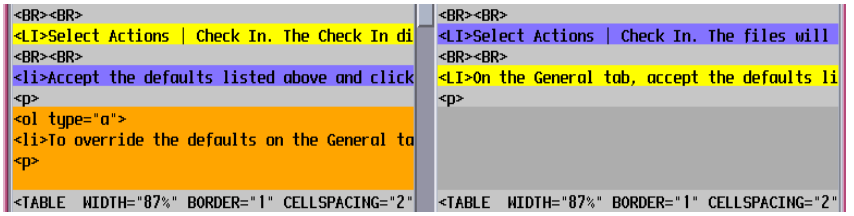
NOTE

- On UNIX, *changes* are treated as *conflicts* and must be resolved by the user.
- To resolve conflicts, you must be able to identify the differences between the base and branch file. For information on how to identify the additions, deletions, and changes between base and branch files, see "Interpreting Difference Results in UNIX" on page 223.

To resolve conflicts between the base and branch file:

1 From the Merge Tool window, scroll to a conflict block.

2 Resolve the conflict by clicking the block of text that you want to appear in the output (target) file. The conflict block that you select changes color.



3 Repeat steps 1-2 until all conflicts have been resolved.

4 Select File | Save As and click **OK** to save the output (target) file. You do not need to reenter the output path or filename.

Scenario: Compare Branch Revisions and Merging Revisions Back Into the Trunk

Terri finishes her bug fixes to the chessboard files. She is now ready to merge her changes back into the main development trunk of the Chess application. She notes that a new revision exists for each of the chessboard files (revision 1.7) and concludes that another Developer modified the files while she worked on the branch revisions. For each chessboard file, Terri's intention is to merge the tip revisions of the trunk and branch into a single file, and then check that merged file back in to the trunk as the newest revision, 1.8.

Terri selects the Board subproject and selects the Lock option. By locking the subproject, Terri ensures that no one will make changes to the Board files during the time she is merging in the changes from her branch development. Starting with the first chessboard file, board.jpr, Terri selects the file and the Show Merge option. She then selects the following options:

- From the **Merge** drop-down menu, Terri selects the **Revisions in a single versioned file** option. This selection indicates that she wants to merge the changes in two different revisions (1.7 and 1.6.1.5) of a single file, board.jpr.
- In the **Base** field, Terri clicks the Browse button to select the revision that corresponds to the common source for the two revisions. In this case, the base revision corresponds to the location where Terri originally started her branch development, revision 1.6.

  Version Manager uses this revision as the point of reference, or comparison, for the merge process. For example, Version Manager identifies additions, deletions, and changes in the trunk tip revision, revision 1.7, by comparing it line-by-line with revision 1.6. It then repeats this process for the branch tip revision, 1.6.1.5.
- In the **Branch 1** field, Terri clicks the Browse button to select the tip revision of the branch, 1.6.1.5. This revision corresponds to the first of the files that Terri wants to merge into a new file.
- In the **Branch 2** field, Terri clicks the Browse button to select the tip revision of the trunk, 1.7. This revision corresponds to the second of the files that Terri wants to merge into a new file.

- In the **Output File** field, Terri enters board_merge.exe as the name of the output or *target* file, which corresponds to the merged file that Terri will ultimately check in as revision 1.8. Because Terri wants to test the merged version of the executable before she checks it in to Version Manager, she provides a different name than that of the original file. Once she is ready to check in the file she can rename it to board.jpr and check it in to Version Manager.

From the Merant Merge Tool, the base and branch files display in the Frames window. By default, the Merant Merge Tool automatically merges the non-conflicting additions, deletions, and changes into the base file and displays the resulting merged version in the Output File window. Terri now must verify the merged changes and resolve the conflicts before she can save the output file.

To identify the differences between the trunk and branch tip revisions, Version Manager first compares revision 1.6.1.5 with the base and color-codes the additions, deletions, and changes in the File 1 section of the Frames window. Version Manager repeats this process for revision 1.7 and displays the results in the File 2 section of the Frames window. This non-conflicting information is automatically incorporated into the target file. If the same line has been changed in both files, Version Manager color-codes the line as a conflict. Terri must resolve these conflicts before she can continue with the merge process.

However, before resolving any conflicts, Terri wants to first verify the changes that have been automatically incorporated into the target file. Terri selects the Color Tips toolbar button, providing a legend that interprets the meaning of the color coding in the Frames window. Terri then selects View | Show Only Differences and uses the Next Difference and Previous Difference toolbar buttons to scan the changes.

Because Merge has automatically incorporated these changes into the Target file, the change blocks display as selected in the branch files. In this case, Terri agrees with all of the modifications made to the output file and, hence, does not make any modifications. However, if she had not agreed with something that had been automatically incorporated, she could have easily selected the change block in the branch file and removed it from the output file. The change block would have then displayed as unselected in the branch file and not displayed in the target file. Instead, the original line from the base file would have displayed as selected and displayed as part of the merged target file.

At this point, Terri must now resolve the conflicts between the files and incorporate the appropriate version into the output file. To do so, she moves to the top of the files in the Frames window and selects the Next Conflict button. Version Manager scrolls the Frames window to the first conflict. Terri double-clicks this conflict block, and the Global Comparison window displays.

Terri reviews the two versions of the line of code and then right-clicks the version that she wants to incorporate into the target file. The selected conflict block changes color to indicate that it is the chosen option. The unselected conflict block changes to a shade pattern that corresponds to it having been resolved elsewhere. She repeats this process for the rest of the conflicts.

When Terri has resolved all of the conflicts, she saves the target file, which is now the candidate for revision 1.8. She tests the code to verify that it still successfully executes. She renames the file back to board.jpr and checks in the new file as revision 1.8 of board.jpr. After she checks in the file, Terri removes the "Chessboard Bug Fix" version label from the revision.

Terri repeats the merge process for each of the files in the Board subproject. Once she checks in the last merged file, she unlocks the project and makes her changes available to the other Developers.

Chapter 18

Promoting Revisions

About Promotion Groups

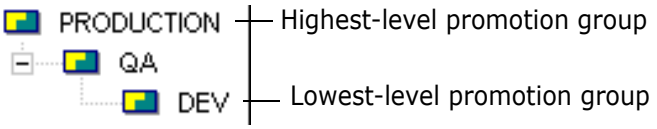
Before you begin…

Before you can begin working with promotion groups, you must have a promotion model set up for your project database. A *promotion model* is a hierarchy of milestones in a development cycle. These milestones are represented by promotion groups. For information on setting up a promotion model, see the "Using Promotion Models" chapter in the *Serena PVCS Version Manager Administrator's Guide*.

What is a promotion group?

A promotion group is a milestone within a promotion model hierarchy. When a promotion model is created for a project database, you assign revisions to the lowest-level promotion group. As development matures and reaches specified milestones, authorized users can promote revisions within the promotion model hierarchy. Development is considered complete when a revision reaches the highest-level promotion group, the top of the promotion model hierarchy.

The following figure is an example of a simple, but typical, promotion model.



Why use promotion groups?

Promotion groups are meant for tracking development as a revision moves through the different milestones in your development process. They help you regulate software changes by making it necessary to associate locked revisions, which are the only revisions that can be edited, with the lowest-level promotion group. You can also use promotion groups to control access to source code by integrating a promotion model with various security options.

Promotion groups are useful in development environments where formal procedures are in place for moving software from one stage to the next, and where Project Team Members have different but well defined responsibilities and tasks.

Logical vs. physical association

When associating revisions with promotion groups, think of the association as a logical association rather than a physical association. Unlike promotion in a mainframe environment, Version Manager does not require you to maintain files at different stages of development in separate physical locations.

Promote permissions

Project Leaders or Administrators are typically the only users who have the authority to promote revisions. To be able to promote revisions, you must have promote permissions assigned to you by your Administrator.

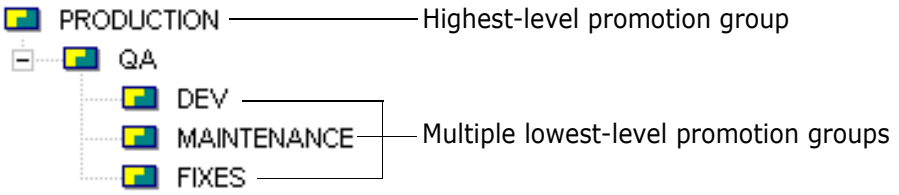
Checking Out Revisions Assigned to a Promotion Group

An important rule to remember with promotion groups: you can only check out revisions to the lowest-level promotion group, the level reserved for development work. Regardless of what promotion level a revision has reached, when you check out and lock a revision, you must assign the revision to a lowest-level promotion group to continue development.

Workspace settings may be used to define a Default Promotion Group. If a Default Promotion Group is not defined for the active workspace, Version Manager will either:

- Prompt you to select a lowest-level promotion group if more than one is defined within a promotion model, or

- Use the lowest-level promotion group if only one is defined within a promotion model.

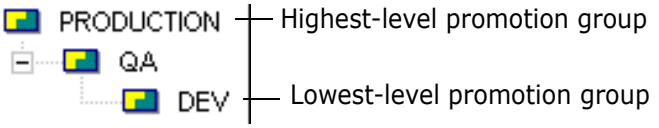


For a comprehensive list of promotion model rules, see the "Using Promotion Models" chapter in the *Serena PVCS Version Manager Administrator's Guide*.

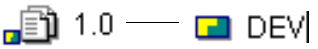
The Promotion Process

The following is an example of the promotion process:

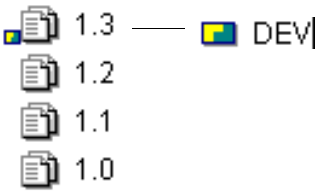
1 A promotion model is created for a project database by the Administrator.



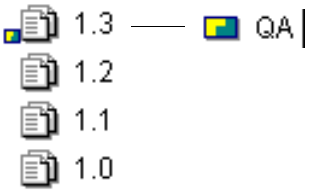
2 The Administrator assigns the DEV promotion group to an entire project. Revision 1.0 of a file within the project is shown below.



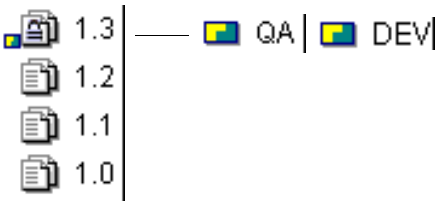
3 Multiple revisions are created in the versioned file as a Developer checks the revisions in and out of the archive. The DEV promotion group remains on the tip revision.



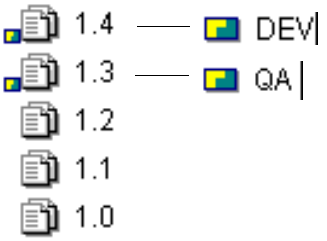
4 When the file is ready for testing, a Project Leader or Administrator promotes the revision to the next promotion group, QA.



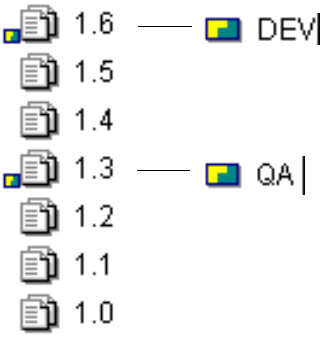
5 The Developer continues to work on the file, so the tip revision is checked out and locked. Because you can only check out revisions to the lowest-level promotion group, revision 1.3 is now assigned to both QA and DEV.



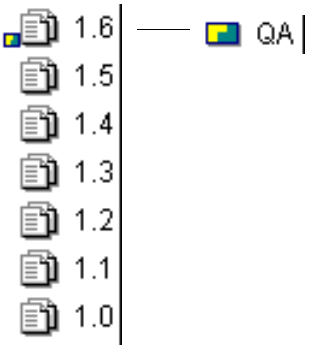
6 When the Developer is finished making changes, the revision is checked back in. The QA promotion group remains with revision 1.3, but the DEV promotion group moves with the tip, revision 1.4.



7 As the Developer continues to check in changes, the DEV promotion group stays with the tip revision and the QA promotion group stays on revision 1.3.



8 When the file is ready again for testing, it is promoted from DEV to the QA promotion group. However, because a promotion group can only be assigned to one revision within a versioned file, the QA promotion group is removed from revision 1.3 and assigned to revision 1.6.



9 After tests have been completed and milestones are achieved, the revision is promoted to the PRODUCTION promotion group, the highest-level promotion group.

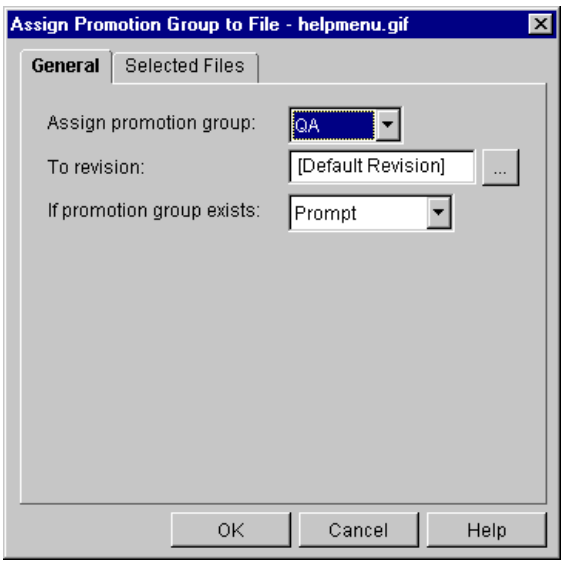
Because this revision will be used for production purposes, the Administrator assigns a fixed version label (such as BUILD) to the revision for identification purposes.

Assigning a Promotion Group to Revisions

You can assign a promotion group to revisions by selecting a revision, a single versioned file, multiple versioned files, a project, a folder, or a project database.

To assign a promotion group:

1. Select the item you want to promote.
2. Select Actions | Promotion Group | Assign. The Assign Promotion Group dialog box appears.



3. Select the promotion group you want to assign by selecting it from the **Assign promotion group** list box.
4. To assign a promotion group to a revision other than the default revision, enter the revision number, version label, or promotion group assigned to the revision you want to assign a promotion group in the **To revision** field, or click the Browse button to select it.

   To define the default revision for your project or project database, see "Defining the Default Version" on page 162.

5 In the **If promotion group exists** drop-down menu, select what you want Version Manager to do if the promotion group you are assigning is already assigned to a revision. **Prompt** is the default option, which asks what you want to do when a promotion group exists.

The other options include **Reassign**, which reassigns the promotion group when a promotion group exists, or **Don't reassign**, which does not reassign the promotion group to the selected revision.

6 (For projects and project databases only). To assign a promotion group to versioned files located in subprojects, select the **Include files in subprojects** check box.

7 Click **OK**.

Promoting Revisions to the Next Promotion Group

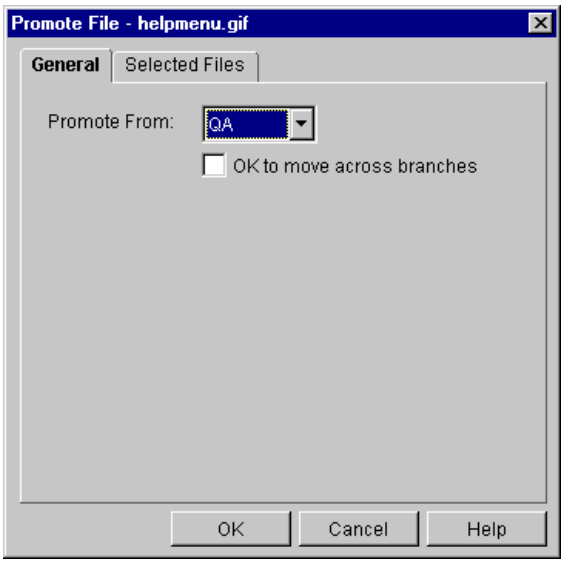
To promote revisions from one promotion group to the next promotion group in the hierarchy, use the **Promote** option. This option enforces the promotion model hierarchy assigned to your project database.

You can promote revisions to the next promotion group by selecting a revision, a single versioned file, multiple versioned files, a project, a folder, or a project database.

To promote revisions to the next promotion group:

1 Select the item that you want to promote.

2 Select Actions | Promotion Group | Promote. The Promote dialog box appears.



3 Select the promotion group that you want to promote from in the **Promote From** list box.

4 To allow reassignment of a promotion group across branches, select the **OK to move across branches** check box. Otherwise, the promotion will fail if a revision on another branch is already assigned the next highest promotion group.

5 (For projects and project databases only). To promote revisions located in subprojects, select the **Include files in subprojects** check box.

6 Click **OK**.

Changing a Promotion Group

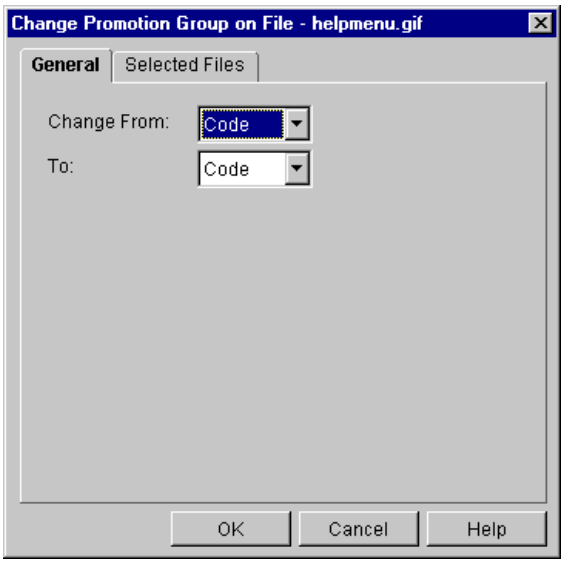
You can change a promotion group if you accidently assigned the wrong promotion group to a revision, or if you want to reassign a promotion group to a different promotion group within the same promotion level. You should not use the **Change** option as a means of promoting revisions because it does not enforce the promotion model hierarchy assigned to your project database.

You can change promotion groups by selecting a revision, a single versioned file, multiple versioned files, a project, a folder, or a project database.

To change a promotion group:

1 Select the item that contains the promotion group you want to change.

2 Select Actions | Promotion Group | Change. The Change Promotion Group dialog box appears.



3 Select the promotion group that you want to change from in the **Change From** list box.

4 Select the promotion group that you want to change to in the **To** drop-down menu.

5 (For projects and project databases only). To change promotion groups assigned to revisions located in subprojects, select the **Include files in subprojects** check box.

6 Click **OK**.

Removing a Promotion Group

Remove a promotion group from a revision whenever the promotion group is no longer needed or necessary. You are not actually removing a promotion group from a promotion model, you are simply removing the promotion group from a revision.

NOTE Even if you remove a promotion group from a revision, as long as a promotion model is in effect for your project database, every time you check out a revision, it will be associated with the lowest-level promotion group.

You can remove promotion groups by selecting a revision, a single versioned file, multiple versioned files, a project, a folder, or a project database.

To remove a promotion group from a revision:

1. Select the item that has been assigned the promotion group you want to remove.
2. Select Actions | Promotion Group | Remove. The Remove Promotion Group dialog box appears.

3. Select the promotion group that you want to remove from the revision in the **Promotion Group** list box.
4. (For projects and project databases only). To remove promotion groups assigned to revisions located in subprojects, select the **Include files in subprojects** check box.
5. Click **OK**.

Scenario: Promoting a Set of Files

The Quality Assurance team is ready to verify the Bridge files. Before they begin their work, they need to promote the files to the QA_Test development milestone. They promote the files for two reasons:

1. To accurately reflect the development stage of the files.

2 To enable the Quality Assurance team to verify the files. The Administrator has associated access privileges with the Development and QA_Test promotion groups. The QA Engineers cannot work with files that are assigned to the Development promotion level and vice versa.

Tim, the QA Project Leader, decides to filter his view by only displaying the files that are associated with the Field Test version label. He selects the Version Label option from the file filter on the tool bar and selects the Field Test version label. Tim selects the Select All option to select all of the versioned files in the File pane, and selects the Promote option from the Promotion Group options. He selects Development from the Promote From list box and clicks **OK**. The Bridge files move from the Development promotion group to the QA_Test promotion level.

The Quality Assurance Engineers can now begin their test process. The Developers no longer have access to the files associated with the QA_Test promotion level. They can, however, continue their work on the general release version of the software, checking out and checking in more changes associated with the Development promotion group. For example, after a Developer adds two additional revisions to the Bridge archive, the revisions would look like this:

- Rev 1.7 Development
- Rev 1.6
- Rev 1.5 QA_Test
- Rev 1.4
- Rev 1.3
- Rev 1.2
- Rev 1.1
- Rev 1.0

QA does not have to concern themselves with any of the new development changes because they are referencing the QA_Test promotion group when they check out and check in revisions. Later, when QA is satisfied with the field test release, the revisions are promoted to the Release promotion group in the same fashion as they were promoted to QA_Test. The Release Manager then associates a version label named Field Test Bridge 2/25/99 with the release files by assigning the version label based on the Release promotion group. This label identifies the revisions used to build the field test version of the bridge application.

The Release Manager can then run a configuration script that builds the application based on the Release promotion group. From this point on, the team can use the fixed version label to rebuild the Field Test release even after the team creates new revisions and associates them with the Release promotion group.

Promotion groups are transiently associated with revisions while version labels are permanently associated (unless explicitly removed), as illustrated below:

- Rev 1.9 Release (Promotion Group)
- Rev 1.8
- Rev 1.7
- Rev 1.6
- Rev 1.5 Field Test Bridge 02/25/99 (Version Label)

- Rev 1.4
- Rev 1.3
- Rev 1.2
- Rev 1.1
- Rev 1.0

Scenario: Demoting Revisions Back to Development for Additional Work

Lisa, a Developer on the Bridge project, realizes that there are two minor errors in two files now associated with the production build of the Field Test software. Before Lisa can make the corrections, she first needs to reassign the files to the Development promotion group. Developers do not have privileges for accessing files while they are at the Release development milestone.

Lisa selects the two files and selects the Assign option from the Promotion Group options. She selects Development from the Assign promotion group list box and clicks **OK**. She can now check out and edit the files.

When Lisa completes her changes, she checks in the files and associates them with the Production promotion group. The Project Manager can rebuild the application based on the Production promotion group. The new build will now incorporate the corrected versions of the two files.

Chapter 19

Using Reports

About Reports

Types of reports

This chapter explains how to set report options, how to customize the format of HTML reports, and how to generate reports in the desktop client. You can use Serena PVCS Version Manager to generate the following reports:

This report...	Contains...
Journal Report	Information about actions that modify an archive.
History Report (previously known as the Archive report)	Information about archives—the archive and workfile names, the archive description, revision numbers, revision history, and more.
Security Report	Information about the contents of the access control database. For more information, refer to the *Serena PVCS Version Manager Administrator's Guide*.

Displaying Reports

You can display reports in an HTML browser or a text editor. On Windows platforms, if you want to display reports in HTML format, Version Manager uses your default browser. On UNIX platforms, if you want to display reports in HTML format, you must specify which HTML browser to use.

Version Manager provides HTML templates that allow you to customize the color and size of the fonts as well as the background color. You can also add graphic files (such as your company logo) to your reports.

Report limitations

When generating large reports (over 500 files), you should display the report in a text editor. Displaying large reports in an HTML editor can take several minutes to display.

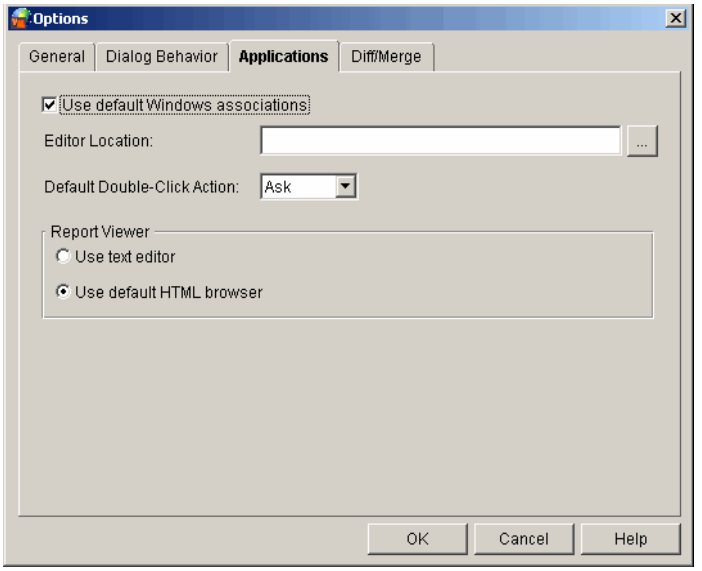
Setting Report Options

Report options vary depending on the platform that you are using to run Version Manager.

To set report options on Windows:

1. Select View | Options.
2. Click the Applications tab. In the Report Viewer group, select either:
   - **Use text editor** if you want to generate and display the report in a text editor (such as Notepad).

- **Use default HTML browser** if you want to generate and display the report in your default HTML browser.



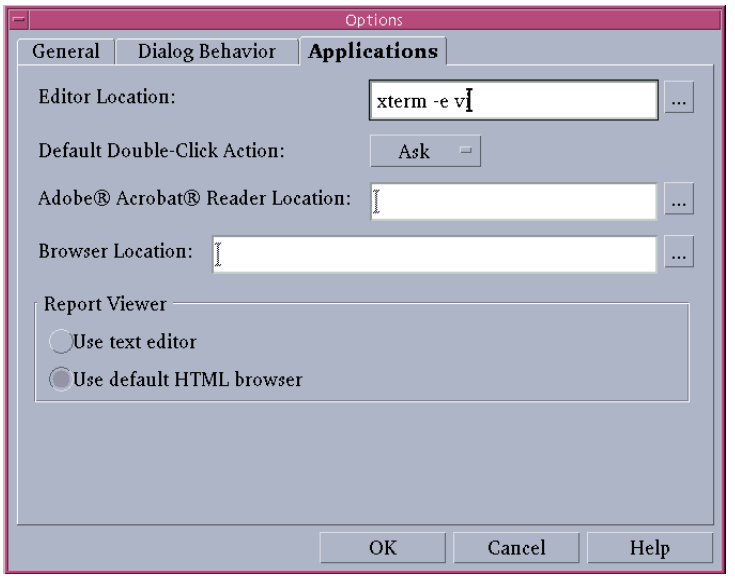
To set report options on UNIX:

1 Select View | Options. The Options dialog box appears with the General tab active.

2 Click the Applications tab. In the Report Viewer group, select either:

- **Use text editor** if you want to generate and display the report in a text editor (such as vi)
- **Use default HTML browser** if you want to generate and display the report in your default HTML browser

If you select **Use default HTML browser**, you must specify the location of the browser in the **Browser Location** field.



Customizing the Format of HTML Reports

You can customize the HTML templates that Version Manager uses to display Journal and History reports. There is one template file per type of report—journal.template and history.template. By default, these files are placed in the following location during Version Manager installation:

- On Windows: *drive*:\Program Files\Serena\vm\common\pvcsprop\pvcs\vm
- On UNIX: /usr/serena/vm/common/pvcsprop/pvcs/vm

Both of these default template files are the same; their contents are as follows:

```
<HTML>
<HEAD>
<TITLE><subst data="REPORT_TITLE_PROPERTY"></subst></TITLE>
<META NAME=GENERATOR CONTENT="Version Manager 6.8">
</HEAD>
<BODY BGCOLOR="#FFFFFF" LINK="#0000EE" VLINK="#0000EE" ALINK="#9933EE"
    TEXT="#000000" LEFTMARGIN="8" TOPMARGIN="8" BACKGROUND="/
    vminet_images/Bkgrd.gif">
<TABLE BORDER="0" CELLSPACING="0" CELLPADDING="0">
<TR VALIGN="top">
<TD>
<B><FONT SIZE="+2"><subst data="REPORT_TITLE_PROPERTY"></subst></
    FONT></B>
</TD>
</TR>
<TR VALIGN="top">
<TD>
<HR WIDTH="100%" SIZE="2" COLOR="Black" NOSHADE>
</TD>
</TR>
<TR VALIGN="top">
<TD>
<PRE>
<subst data="REPORT_FILE_PROPERTY"></subst>
</PRE>
</TD>
</TR>
<TR VALIGN="top">
<TD>
<HR WIDTH="100%" SIZE="2" COLOR="Black" NOSHADE>
</TD>
</TR>
</TABLE><!--Footnote-->
<P>
</BODY>
</HTML>
```

To display the report title, the template must have the tag:

```
<subst data="REPORT_TITLE_PROPERTY"></subst>
```

To display the report, the template must have the tag:

```
<subst data="REPORT_FILE_PROPERTY"></subst>
```

You can change any of the formatting tags in the template files. For example, you can customize the report by changing the color and size of the font, the background color, etc.

You can also add formatting tags, any text that you want, links to other pages, and graphics files (for example, your company logo).

About Journal Reports

Before you begin…

A Journal Report is based on the information in a journal file. A journal file contains a record of the actions that users perform on archives. Your project database must be configured to keep a journal file; otherwise, you will not be able to generate a Journal Report. For information on keeping a journal file, see the "Configuring Version Manager" chapter in the *Serena PVCS Version Manager Administrator's Guide*.

What is a Journal Report?

A Journal Report contains information about changes made to archives, such as assigning version labels, checking workfiles in and out, and assigning a promotion group.

Why generate a Journal Report?

A Journal Report is helpful when you want a quick summary of specific information about archive activity, such as finding out when revisions were checked out of a particular archive or when version labels were assigned.

Journal vs. History Report

Using a Journal Report is faster than generating a History Report because Version Manager does not have to open the archives before generating a Journal Report—the data is already in a journal file. If you want general information about an archive, such as the date of creation, revision information, or access restrictions on the archive, generate a History Report. See "Generating History Reports" on page 248.

Journal Report options

You can set many different options before you generate a Journal Report that will limit the contents of the report. You can generate a report that shows:

- Changes made between certain dates
- Changes made to certain archives
- Archive activity by certain actions (commands)
- Changes made by certain users
- Currently locked revisions in archives by specified users

Generating a Journal Report

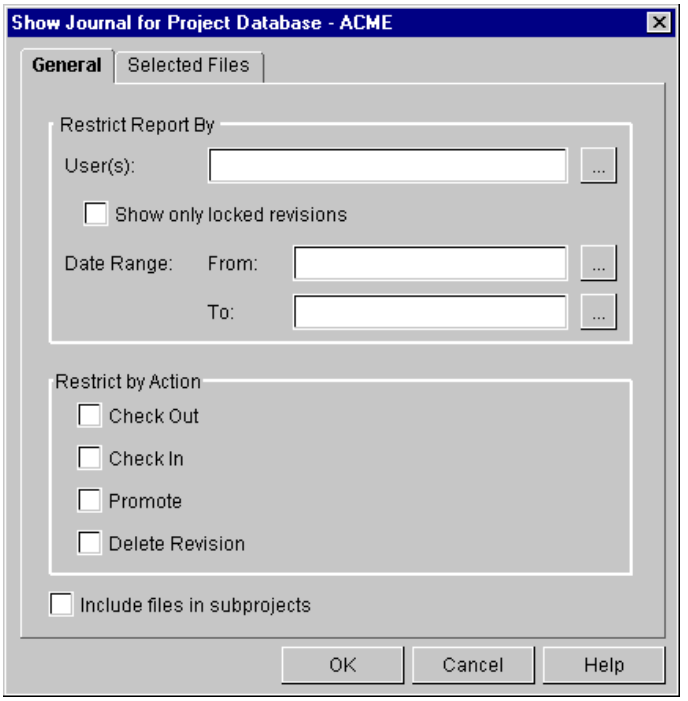
You can generate a Journal Report for a project database, project, multiple versioned files, or a single versioned file. The journal file that is used to generate the report is the one that is specified in the configuration file associated with the item you select. If a journal file is not specified in the configuration file for the selected item, a Journal Report cannot be generated.

You must have the Journal Report privilege to generate a Journal Report. For Journal Report privileges, see your Version Manager Administrator.

To generate a Journal Report:

1 Select a project database, a project, or one or more versioned files.

2 Select Actions | Show Journal. The Show Journal dialog box appears.



3 To generate a report that contains every modification that has been made to an archive by every user and all currently locked revisions, click **OK**. Otherwise, you can limit the contents of the report by doing any of the following:

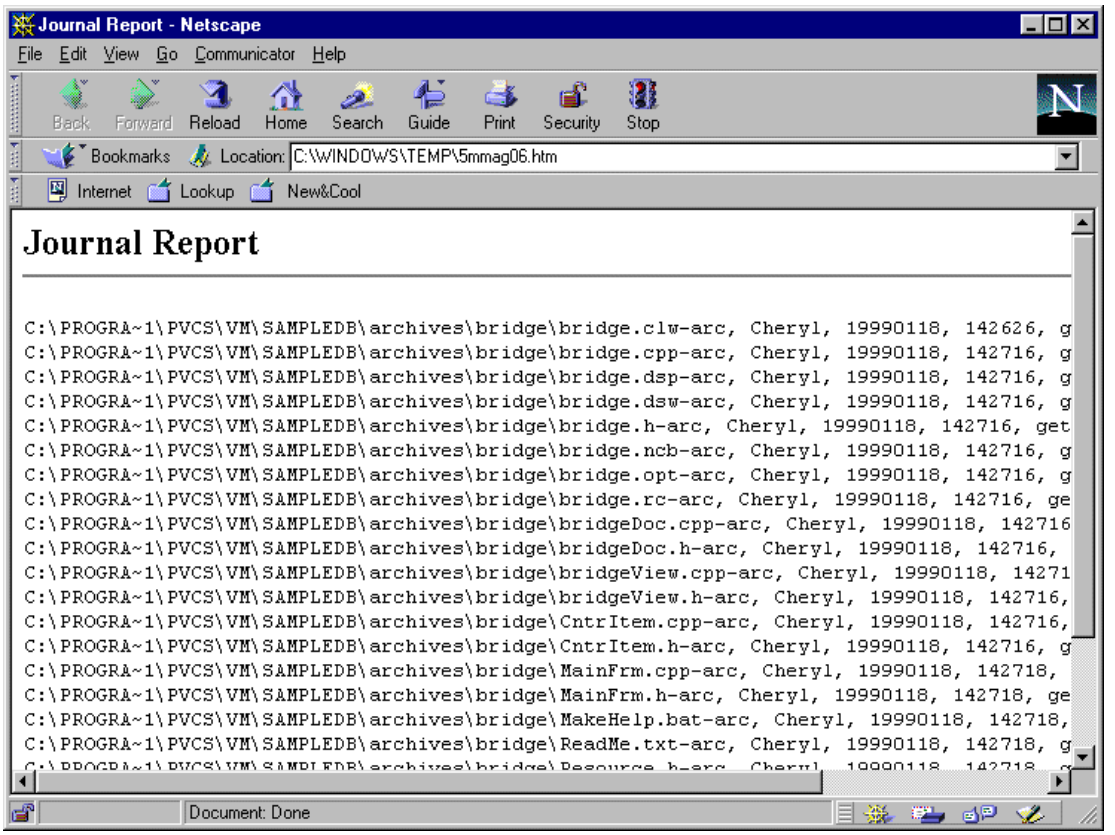
To view...	Do this...
Changes made by certain users	Enter the user IDs of the users in the **User(s)** field. Separate multiple IDs by commas.
Currently locked revisions	Select the **Show only locked revisions** check box. If you entered user IDs in the **User(s)** field and select this check box, Version Manager will report the currently locked revisions of the users specified.

To view...	Do this...
Changes made in a certain range	Enter the start date in the **Date Range From** field and the stop date in the **Date Range To** field, or click the icon next to the field and choose the date. You must enter the date and time in the formats that you have set for your operating system. On Windows, you set the date and time formats from the Control Panel | Regional Settings. On UNIX, you define the formats by setting the PVCS_DATE_FORMAT and PVCS_TIME_FORMAT environment variables. **NOTE** On UNIX, you should set the PVCS_DATE_FORMAT to MM/dd/yyyy, or if you want days to lead months to dd/MM/yyyy. Notice that the month is specified in capital letters, and that the year is specified in four digits. These formats will give you the best results. If there are no control panel settings and these environment variables are not set, then the format is mm/dd/yy hh:mm in the US and dd/mm/yy hh:mm in the UK. An uppercase H indicates the 24 hour clock display.
Only archives that have had revisions checked out	Select the **Check Out** check box.
Only archives that have had workfiles checked in	Select the **Check In** check box.
Only archives that have had revisions promoted	Select the **Promote** check box.
Only archives that have had revisions deleted	Select the **Delete Revision** check box.
A report for an entire project, including all of the versioned files within its subprojects	Select the **Include files in subprojects** check box.

4 Click **OK**. The Journal Report is generated and displayed.

How to Read a Journal Report

The following is an example of a Journal Report:



Each Journal Report line has the following format:
archive, user_id, date, time, action, cl_option

where:

archive is the path and file name of the archive.

user_id is the user ID of the user who made the changes to the archive.

date is the date of the changes in packed decimal numeric form. For example, March 31, 1999 appears as `19990331`.

time is the time of the changes in packed decimal numeric form (using a 24-hour clock). For example, 4:55:20 P.M. would be `165520`.

cl_option is the command-line option of the action that was taken. For example, `put -r1.0`, which means revision 1.0 was checked in.

About History Reports

Before you begin… Your Version Manager Administrator can restrict your ability to generate a History Report. If you do not have privileges to generate a History Report, contact your Version Manager Administrator.

What is a History Report?

A History Report summarizes information about archives and/or revisions. It provides information about archives that you can use to monitor the development process, review archive histories, and check archive attributes.

NOTE The History Report was called archive report in previous releases of Version Manager.

Archive and revision information

History reports can include a combination of archive and revision information. Archive information includes the:

- User ID of the person who created the archive
- Date the archive was created
- Archive attributes
- User ID of the person who has a locked revision
- Archive and workfile names

Revision information includes:

- A description of the revision
- The revision history

History Report options

You can set many different options before you generate a History Report that will limit the contents of the report. You can generate a report that shows:

- Archive information but not revision information.
- Revision information but not archive information.
- Currently locked revisions.
- Revisions that correspond to a specified version label.
- Revisions that are associated with a specified promotion group.
- The newest revisions on the trunk.
- Archives whose tip revisions are not the same as the specified revision number or version label. This is useful when you want to list all archives that have changed since a particular version.

You can further restrict any of the report types you choose by any two of the following:

- Date
- Authors of revisions
- Users who have revisions locked
- Owners of the archive

Generating History Reports

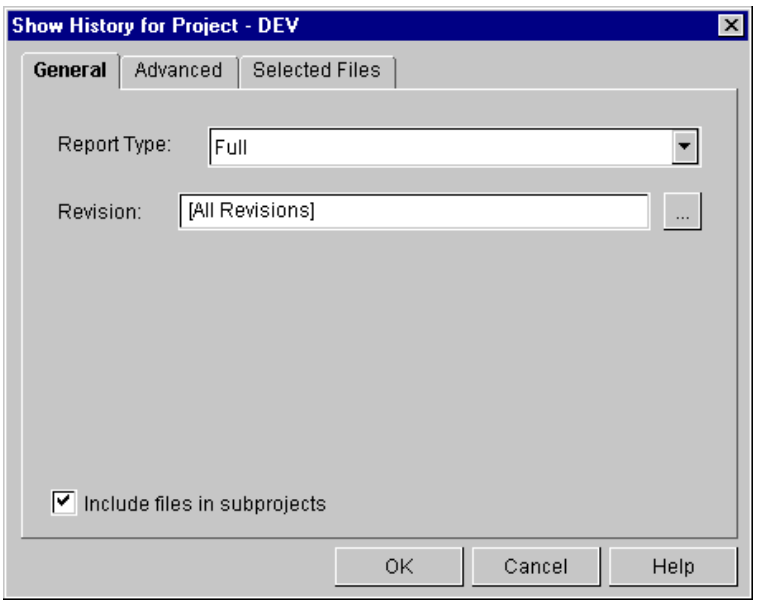
You can generate a History Report for a project database, project, multiple versioned files, a single versioned file, or a specific revision.

NOTE You cannot generate a History Report for a 5.3/6.0 project. You can generate the report on the contents of the project, but not the project itself.

To generate a History Report:

1 Select a project database, a project, one or more versioned files, or a specific revision.

2 Select Actions | Show History. The Show History dialog box appears.

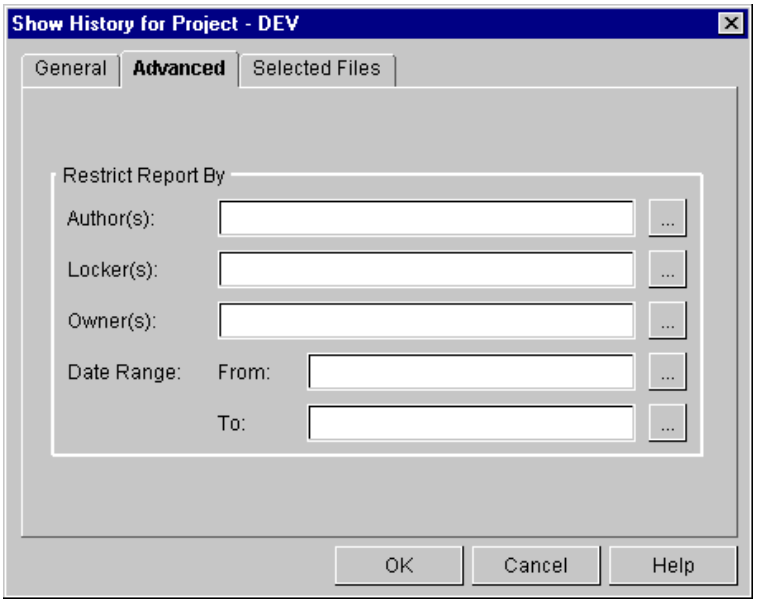


3 Under the General tab, select a report type in the **Report Type** field.

Select this report type...	If you want a report that contains...
Full	Comprehensive (including file, revision, lock, and version label) information on all or selected revisions. To select specific revisions, enter the revision number or version label in the **Revision** field, or click the Browse button to select one.
File information only	Only archive information (such as creation date, owner, locks, version labels). This report does not contain revision history.
Revision information only	Only revision information on all or selected revisions. To select specific revisions, enter the revision numbers (separated by commas) in the **Revision** field.
List locked revisions	A list of the locked revisions within the selected versioned file(s).

Select this report type...	If you want a report that contains...
List revisions with version label	A list of revisions that match a specific version label. To select the version label, enter the version label in the **Label** field or click the Browse button to select a label. To specify a version label that begins with a number, you must precede it with a backslash (\). For example, `\1.2` or `\1abc`.
List revisions in group	A list of revisions that match a specific promotion group. To select a promotion group, enter the promotion group in the **Group** field, or click the Browse button to select a promotion group.
List newest revisions	A list of the latest revisions (if multiple versioned files are selected).
Check tips against version/ revision	Information that compares the latest revision with a specified revision. To specify the revision, enter the revision number, version label, or promotion group in the **Revision** field, or click the Browse button to select a revision. If selecting the revision by a version label, enter a backslash before a version label that begins with a numeric character (such as \1.2label).

4 Under the Advanced tab, you have the option of restricting report information by generating a History Report based on author, user locks, owners, date range, or a combination of two of these options.



- To generate a history report based on a specific author or authors, enter one or more authors' names in the **Author(s)** field, or click the Browse button to select an author. If you are entering more than one author, separate the authors with a comma (,).
- To generate a history report based on user locks, enter one or more users' names in the **Locker(s)** field, or click the Browse button to select a user. If you are entering more than one user, separate the users with a comma (,).

- To generate a history report based on a specific owner or owners, enter one or more owners' names in the **Owner(s)** field, or click the Browse button to select an owner. If you are entering more than one owner, separate the owners with a comma (,).
- To generate a report based on a date range, enter the start date in the **From** field and end date in the **To** field. You must enter the date and time in the formats that you have set for your operating system. On Windows, you set the date and time formats from the Control Panel | Regional Settings. On UNIX, you define the formats by setting the PVCS_DATE_FORMAT and PVCS_TIME_FORMAT environment variables. If there are no control panel settings and these environment variables are not set, then the format is mm/dd/yy hh:mm in the US and dd/mm/yy hh:mm in the UK.

NOTE On UNIX, it is recommend that you set the PVCS_DATE_FORMAT to MM/dd/yyyy, or if you want days to lead months to dd/MM/yyyy. Notice that the month is specified in capital letters, and that the year is specified in four digits. These formats will give you the best results.

- Click **OK**.

How to Read a History Report

The following is an example of a History Report. This example is a complete History Report that contains both archive and revision information. The information above the dotted line is the archive information; the information below the dotted line is revision information.

Change History

```
Archive:          C:\Program Files\PVCS\VM\SampleDb\archives\bridge\bridge.clw-arc
Workfile:         bridge.clw
Archive created:  18 May 1998 15:37:40
Owner:            Admin
Last trunk rev:   1.0
Locks:
Groups:           Development : 1.0
Rev count:        1
Attributes:
   WRITEPROTECT
   CHECKLOCK
   NOEXCLUSIVELOCK
   NOEXPANDKEYWORDS
   NOTRANSLATE
   NOCOMPRESSDELTA
   NOCOMPRESSWORKIMAGE
   NOGENERATEDELTA
   COMMENTPREFIX = "   "
   NEWLINE = "\r\n"
Version labels:
Description:
Sample Project Database - first revision.


----------------------------------
Rev 1.0
Checked in:     18 May 1998 15:37:40
Last modified:  18 May 1998 15:37:40
Author id: Admin     lines deleted/added/moved: 0/0/0
Initial revision.
==================================
```

Archive information

Revision information

Appendix A

Naming Conventions and Restrictions

General Naming Conventions and Restrictions

You can use most alpha, numeric, and special characters when creating or renaming Version Manager entities and paths. However, your operating system also determines the conventions that apply to file and directory names.

Prohibited Characters for Files and Directories

The following characters are prohibited by Version Manager, and most operating systems, when naming files or directories:

- Angle brackets (>) and (<)
- Asterisk (*)
- Colon (:)
- Pipe (|)
- Question mark (?)
- Quotation mark (")
- Slashes, forward (/) and backward (\)
- Space ( ) as the first or last character
- Tab

Naming Considerations for Cross-Platform Environments

When working in a cross-platform environment, be aware of any incompatibilities between the systems and limit your usage to that which they have in common.

IMPORTANT! In a cross-platform environment, you cannot place files into the same directory if they differ only be case. Such usage is possible only in UNIX-only environments.

Specific Naming Conventions and Restrictions

The following table lists naming conventions and restrictions that apply to specific Version Manager entities and paths.

	Restrictions	
Item Type	**Characters**	**Length**
Archives	As listed in "Prohibited Characters for Files and Directories" on page 252, plus cannot use: Ampersand: & Brackets: [ ] Parenthesis: ( ) Plus sign: + Semicolon: ;	254 (full path including the file name) **NOTE** Only the first 10 characters of the archive suffix are significant in distinguishing identically named files in the same project.
Files and paths (unless otherwise noted)	As listed in "Prohibited Characters for Files and Directories" on page 252.	254 (full path including the file name)
pvcs_bindir	As listed in "Prohibited Characters for Files and Directories" on page 252.	254 (full path including the file name) **NOTE** On UNIX, the name of the `vconfig` file and the separator character also count against the total length.
Project databases	Cannot begin or end with a tab or space character. Any character can be used within the name.	
Projects	As listed in "Prohibited Characters for Files and Directories" on page 252, plus cannot be: ■ The two-character name of: .. ■ The one-character name of: . ■ The one-character name of: @	

	Restrictions	
Item Type	**Characters**	**Length**
Promotion groups	Ampersand: & Brackets: [ ] Comma: , Equal sign: = Parenthesis: ( ) Plus sign: + Question mark: ? Semicolon: ; Slash: /	
User, group, and privilege names	Asterisk: * Colon: : Backward slash: \ Single quote: ' Quotation mark: " Parenthesis: ( )	30
Version labels	Ampersand: & Asterisk: * Brackets: [ ] Colon: : Equal sign: = Minus sign: - Parenthesis: ( ) Plus sign: + Question mark: ? Quotation mark: " Semicolon: ; Slash: /	254

Index

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W