SERENA®

PVCS® VERSION MANAGER™ 8.3

Administrator'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 describes how to configure, administer, and maintain Version Manager projects and archives using the desktop client or command-line interface.

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.

Contacting Technical Support

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

Chapter 1

Setting Up Version Manager Using the Desktop Client

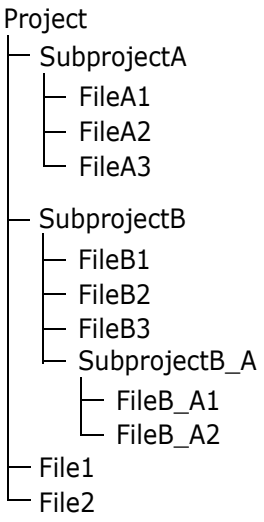
Introduction

To understand how to set up Serena PVCS Version Manager using the desktop client, you must understand Version Manager project databases and projects. This chapter provides information to help you plan and define Version Manager project databases using the desktop client.

For information about how to set up Version Manager using the command-line interface, see "Setting Up Version Manager Using the Command-Line Interface" on page 113.

About Version Manager Projects

A project is a set of interrelated files that you store in Version Manager; it is the entity that Version Manager uses to organize your files. Within a project, you can have subprojects and files, as shown in the following figure.



A Version Manager project is similar to a file system directory. Like a directory, a project is a set of files that you create and maintain. Also, like a directory, a project is hierarchical, meaning subprojects can exist under projects and subprojects under other subprojects. However, projects are not exactly like directories. Unlike a directory, a Version Manager project stores changes to files as deltas in archives—directories do not store old versions of files.

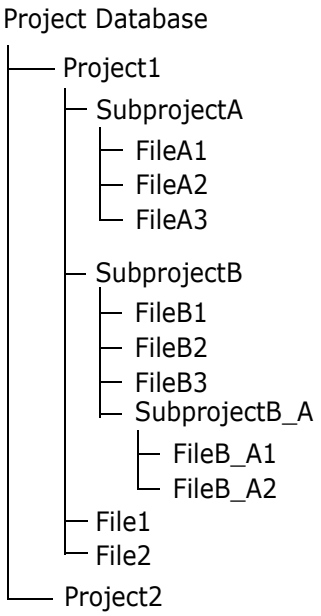
NOTE Change Log

Changes to the contents of a project, such as adding or removing subprojects or versioned files, are recorded in a `change.log` file located in the root directory of the project database. For more information, see "Viewing Changes to Projects (change.log)" on page 309.

All Version Manager projects are contained within a Version Manager *project database*.

About Version Manager Project Databases

A project database is the container for projects (see the following figure).



A project database defines how Version Manager operates on all of the projects within the project database. By default, projects inherit the configuration settings of the project database. You can also change the configuration settings for each individual project.

The first time you run Version Manager, you can let Version Manager create a new empty project database for you. This project database is configured but contains no projects. This empty project database is created in a directory named newdb beneath the Version Manager install directory.

NOTE The creation of project databases may be restricted by the System Administrator. If this privilege is restricted, Version Manager does not create a new, empty project database the first time it is run.

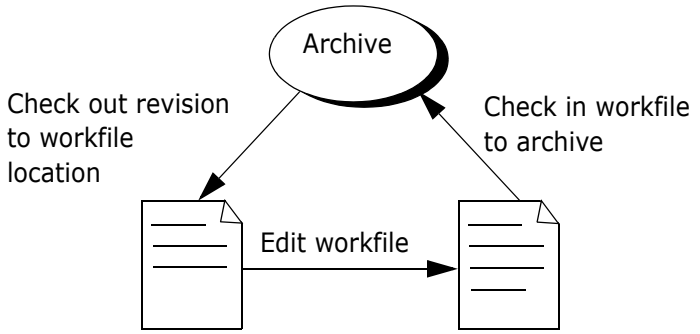
The directories and files that are created, by default, when you create a project database are as follows:

- *Archives* directory
- A configuration file
- An access control database
- *Work* directory
- *Pvcsuser* directory
- A tools configuration file
- *Lib* directory
- A system identification file and metadata files

Archives directory	The *archives* directory contains no archive files. It is populated with archive files when you create projects within the project database and add files to the projects. You specify the location of this directory when you create a project database.
Configuration file	By default, a new configuration file is created and placed in the archives directory. For security purposes, Version Manager masks the name of this configuration file. For example, this file may be assigned a name such as "c6bonpj1.cfg." A configuration file contains configuration settings for all projects within the project database. Version Manager reads this file before any other configuration file. This ensures that all users share the same settings for options. You can modify the settings in this file to suit your needs. Optionally, when you create a project database, you can specify to use an existing configuration file or copy an existing configuration file to the default location (the archives directory). Refer to "Understanding Configuration Files in the Desktop Client" on page 58 for a complete explanation of configuration files.
Access control database	By default, a new access control database is created and placed in the archives directory. This file is not enabled in the configuration file. For security purposes, Version Manager masks the name of this access control database. For example, this file may be assigned a name such as "c6bonpj1.db." An access control database defines the users who are authorized to perform actions on projects and defines the actions that users can perform. Optionally, when you create a project database, you can specify to use an existing access control database or copy an existing access control database to the default location (the archives directory). Refer to "About Access Control Databases" on page 204 for a complete explanation of access control databases.
Work directory	The *work* directory is empty and is the default location for the workfiles when you check out revisions from Version Manager archives. You specify the location for this directory when you create a project database.
pvcsuser directory	The *pvcsuser* directory contains user information such as private workspace settings.
Tools configuration file	The tools configuration file contains the definition of one custom tool that launches PVCS TeamLink if Serena Dimensions CM is installed on your system. You can add custom tools to this file. Refer to "Adding Custom Tools" on page 183.
lib directory	The *lib* directory contains information about the Version Manager release you used to create the project database.
Metadata files	A counter file for entity identifications named pvcsid.ser is created. Do **not** move or delete this file. The pvcsroot.ser and pvcsproj.ser files are created and may contain metadata such as which configuration file is being used, as well as the workfile and archive locations.

Planning a Project Database

Typically, a Version Manager project database, which includes the archives, is on a networked file system, and developers work on files locally. The workflow is as follows:



Before you define a project database, you need to know the following:

- Where the workfiles, or source files, that you want to place under source control are located. See "Workfiles" on page 17.
- Where you want the archives for the files to be located. See "Archive Locations" on page 18.
- Where, by default, you want users to edit the workfiles. Often the workfile location is the directory in which the files originally reside. See "Workfile Locations" on page 18.
- Whether any of the files that you are placing under source control are shared by more than one project or subproject. See "Shared Archives" on page 19.
- Whether you have existing (5.3/6.0) Version Manager archives and projects that you want to upgrade to the new release of Version Manager. See "Upgrading 5.3/6.0 Projects" on page 20.
- Whether you have SourceSafe files to convert. See "Converting SourceSafe Databases" on page 21.
- Whether your environment is cross-platform. See "Cross-Platform Environment" on page 21.
- Whether you will integrate the project database with Serena Builder. See "Serena Builder" on page 22.

Workfiles

You should mirror the directory structure of the workfiles when you set up a project database. For example, if you have the following directory structure, we recommend that you define a project database named MyApp with three projects beneath it—include, resource, and source.

By default, Version Manager automatically creates the archive locations to reflect the workfile structure. You can redefine these locations when you create a project database.

Archive Locations

The archive location is where the new archives for the workfiles are located. Typically, this location is on a networked file system, in a location accessible to all authorized users. By default, the archive location is a directory beneath the project database location, but you can specify a different archive location when you create a project database.

NOTE If you are using the Version Manager File Server, the archives may be located on the server. See "Configuring and Using the Version Manager File Server" on page 129.

For UNIX users: Version Manager files are installed in setuid mode to implement an additional level of security for your archives. In setuid, your users will log in as themselves, but Version Manager will create public archives as the user who owns the executables. We recommend that you create a user named pvcs for this purpose. This will be the only user with access to read and write to your archives.

NOTE If you are using the Version Manager File Server, you do not need to use setuid to secure your archives. See "Configuring and Using the Version Manager File Server" on page 129.

When you run in setuid mode, all files are created under the user pvcs, except for workfiles, temporary files, pvcspriv project domain, and `$HOME/.islvrc`. Access control privileges can be controlled by the Version Manager access control database (see "Using Security" on page 203). Individual users who are not pvcs cannot modify, add, or delete the files or directories unless they are using Version Manager commands.

If you don't run in setuid mode, archives can be moved, renamed, corrupted, or deleted with basic UNIX commands by anyone with permissions to your archive directories.

NOTE You may not be able to use setuid if you are running within a cross-platform environment. For information on configuring setuid in a cross-platform environment, see the *Serena PVCS Version Manager Installation Guide*. For information on turning off setuid, see "Turning Off Setuid" on page 41.

Workfile Locations

A workfile location is the location where Version Manager places a revision checked out of an archive. Generally, the workfile location is a directory beneath the project database location, but you can specify any workfile location when you create a project database. When you check out a revision, it becomes a workfile.

Each project within a project database also has a workfile location. For example, `c:\prjdb\work\project1`, `c:\prjdb\work\project2`, and so forth.

The workfile locations of a project database and its projects are stored in a workspace. Workspaces can be either for public or private use. A default public workspace (called the Root Workspace) is created for each project database.

As the Administrator, you can create other public workspaces and override the original workfile locations. For example, you could create public workspaces that define the workfile locations to be on a networked file system to which all authorized users have access. By doing this, you can be assured that the default workfile locations are ones that users can access.

Users can then create private workspaces that change the workfile locations to their local hard drive so that they can work on the files locally. Or, you can specify a common local drive as the default workfile location (for example, C: in Windows). See "Creating a Public Workspace" on page 179 for information about how to create workspaces.

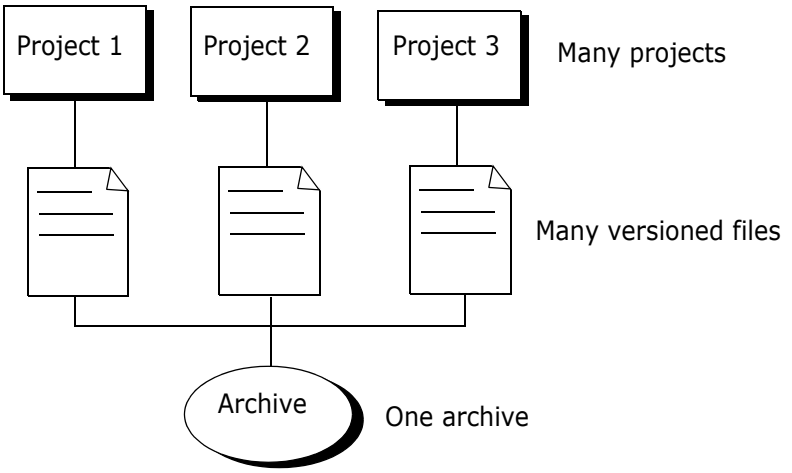
If you are setting up a project for files that already exist, you would typically define the workfile locations to be the directories in which the existing files reside.

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.

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.

Shared Archives

Version Manager lets you share archives across multiple projects. In this case, you have one archive to store the file and many projects that reference the file, as shown in the following figure.



For example, a company develops and sells three different software products, and each one of the products uses the same license agreement. The license agreement can be shared among all three projects of the product. In this case, the archive for the license agreement is created in one project and the other two projects reference the archive.

By sharing archives, you can be assured that the license agreement is the same for each product. In contrast, having multiple copies of a common file, each in its own archive, increases the risk of the files being different—one is updated but no one remembers to update the others.

To share files across projects, set up one project with the archives for all of the shared files. Then copy the versioned files into the other projects. The other projects will not have an actual archive for the copied files, but will reference the archive in the original project.

NOTE Archives are shared, not projects. If you eventually add a new file to the project, it will not be shared with any other project until you copy the versioned file to the desired project or projects.

Once your projects are defined, you can copy files anytime by using Edit | Copy Files. A copy of the versioned file is created in the destination project that references the archive.

Upgrading 5.3/6.0 Projects

To fully utilize the newest Version Manager and the functions available on your 5.3/6.0 Version Manager projects, you must copy the projects into the new Version Manager project database format.

You can also upgrade 5.3/6.0 projects that were created using the Version Manager Development Interface. For complete information about how to upgrade 5.3/6.0 projects that were created in the Version Manager Development Interface, refer to the *Serena PVCS Version Manager IDE Client Implementation Guide*.

If you do not copy the projects but just open the older projects in Version Manager, the projects are recognized and displayed, but you cannot use all of the Version Manager functions available to you. You can, however, use the majority of the functions, including:

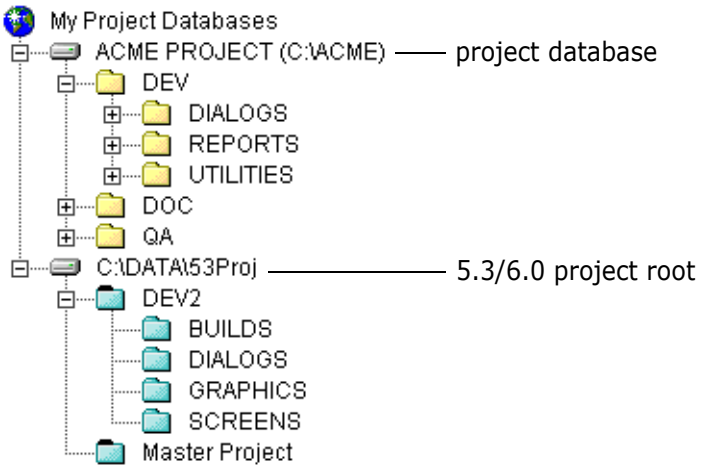
- Creating and deleting folders.
- Checking files in and out of archives.
- Adding workfiles (but not adding directory structures).
- Defining and setting workspaces.
- Deleting revisions.
- Using promotion models and version labels.
- Importing archives (but not adding directory structures).

You cannot:

- Create new projects within the 5.3/6.0 project root.
- Delete projects from the 5.3/6.0 project root.
- Create subprojects within the 5.3/6.0 project root.
- Copy projects, subprojects, or versioned files from a newer project database into a 5.3/6.0 project root.
- Configure a 5.3/6.0 project root.
- Change security settings in the access control database.

When you open a 5.3/6.0 project root in Version Manager, it is displayed in the projects pane as a path name—unlike new project databases, which are displayed with a name and

a pathname. Also, the icons that represent the projects within a 5.3/6.0 project root are blue, whereas the icons within a newer project database are yellow.



Converting SourceSafe Databases

Files in a Visual SourceSafe database may be converted into Version Manager archives. The converter:

- Retains file histories and comments on each SourceSafe file you choose to convert
- Creates Version Manager projects that correspond to the SourceSafe projects being converted
- Retains user-defined labels and comments (see "User-Defined Labels" on page 322)
- Converts branches
- Configures projects to use the shared files encountered during the conversion

For more information on converting SourceSafe files to Version Manager archives, refer to Chapter 12, "Using the Version Manager Conversion Utility for SourceSafe" on page 321.

Cross-Platform Environment

Using the Version Manager desktop client, you can share configuration files, access control databases, project databases, and archives between Windows and UNIX. You can store archives on UNIX and access them from Windows, or vice versa, through project databases.

File Server Based Cross-Platform Environments

The Version Manager File Server works with Version Manager archives and clients regardless of O/S. No special steps are required to enable a cross-platform environment.

For more information on the Version Manager File Server, see "Configuring and Using the Version Manager File Server" on page 129.

Network-Based Cross-Platform Environments

To access a project database in a network-based cross-platform environment, you must create the project database using the Version Manager desktop client on Windows. From Windows or UNIX, you can access project databases created in Windows (whether or not they reside on UNIX or Windows); however, you can access project databases created using the Version Manager desktop client on UNIX only from UNIX.

NOTE You may not be able to use setuid if you are running within a cross-platform environment. For information on configuring setuid in a cross-platform environment, see the *Serena PVCS Version Manager Installation Guide*.

Serena Builder

In order to use Serena Builder with a Version Manager project database, you must ensure that the path to the project database does NOT contain a dollar sign ($). Outside of the integration to Builder, this is not a concern.

For more information about integrating to Serena Builder, see the *Serena Builder for Professional User's Guide*.

Setting Up a Project Database

The following steps describe how to set up a Version Manager project database. Following this section are detailed procedures.

1. Create a project database, which defines where the archives for the projects are located. Version Manager can create a new empty project database the first time you run the program. You can use this project database rather than create a new one.
2. Create projects within the database (optional), which define the workfile location for the files in the project.

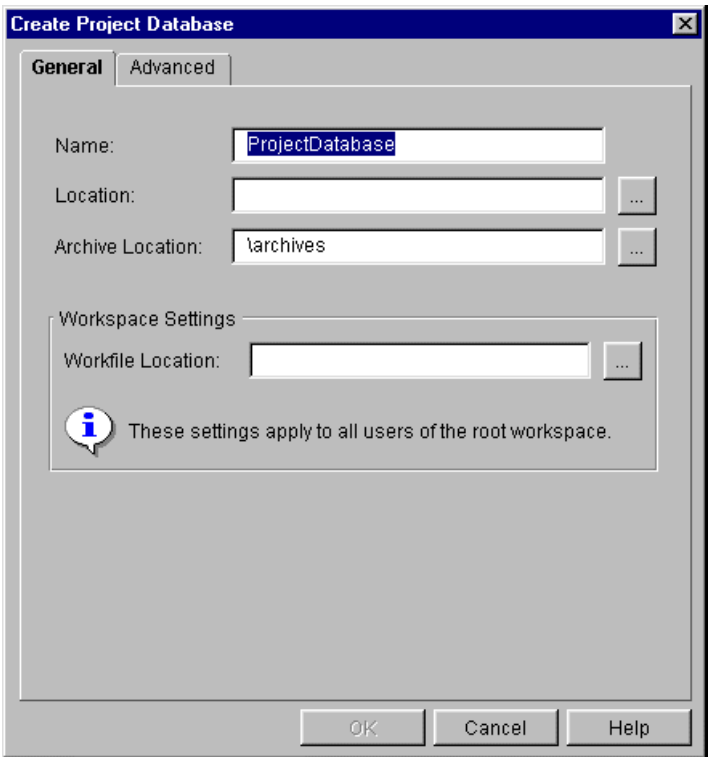
   If you are setting up a project for existing files, you do not have to create a project. You can simply add directories to the project database, and Version Manager will create the project for you. Only create a project if you want to add individual files to it or organize the files in a manner other than the current workfile structure.
3. Add the workfiles to the project database or project.
4. Copy files across projects to share archives (optional).
5. Upgrade 5.3/6.0 projects to the new project format (optional).

Creating a Project Database

Version Manager can create a new, empty project database the first time you run the program. To get started, you can use this project database rather than create a new one, or you can create a new one.

To create a project database:

1 Select Admin | Create Project Database. The Create Project Database dialog box appears.



2 Specify a name for the project database in the **Name** field.

The name cannot begin or end with a tab or blank space; all characters are valid.

3 Specify the location of the project database in the **Location** 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.

NOTE

- You **cannot** create a project database in the same location as an existing 5.3/6.0 project root.
- The path must **NOT** contain a dollar sign ($) **IF** you wish to integrate with Serena Builder. See "Serena Builder" on page 22.

4 If you want the archive location to be a location other than the default, specify the location in the **Archive Location** field.

The default location is a directory named archives beneath the project database location. This location must be accessible to all users who will be accessing the archives.

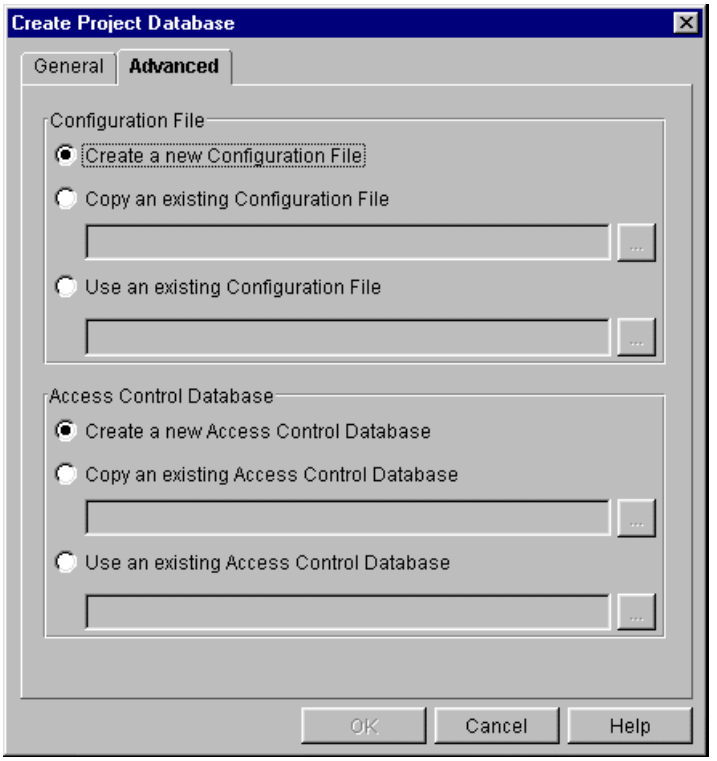
5 Specify a workfile location in the **Workfile Location** field. This is a required field.

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. 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 create a private workspace to change the workfile location to a different location if they want.

6 Click the Advanced tab to optionally specify a configuration file and access control database to associate with this project database.

If you do not do this step, Version Manager creates a new configuration file and access control database for the project database.



7 Click OK.

Version Manager creates several directories and files in the project database location. See "About Version Manager Project Databases" on page 15 for an explanation of the directories and files that are created.

Creating a Project

If you are setting up a project for existing files, you do not have to create a project. Instead, you can simply add directories and files to the project database using File | Add Workfiles, and Version Manager will create the projects for you. In this case, Version Manager creates projects with the same name as the directories and adds the files within the directories to the projects.

If the directories have subdirectories, you can have Version Manager optionally create subprojects and populate the subprojects with the appropriate files.

You may want to create a project to organize the versioned files into a structure that is different from their workfile structure or to add individual files.

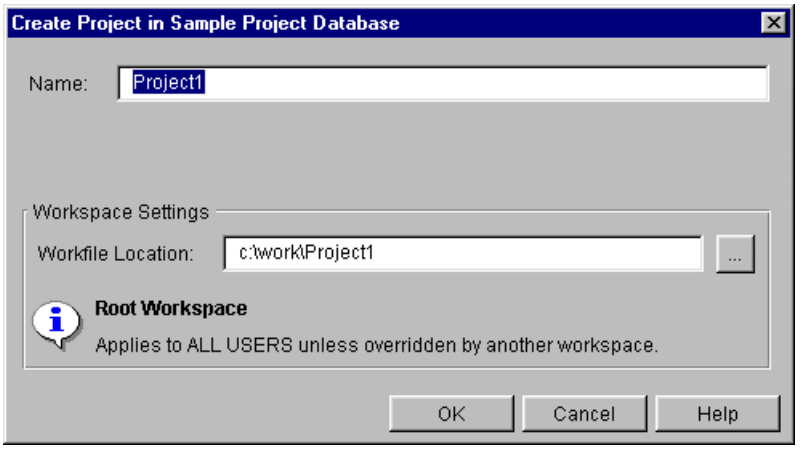
The directories that are created when you create a project are as follows:

- A project archives directory beneath the archives directory of the project database (for example, `\myapps\ProjectDB1\archives\ProjectA`).
- A project work directory in the location you specify. By default, it is beneath the work directory of the project database (for example, `\myapps\ProjectDB1\work\ProjectA`).
- A project metadata directory beneath the project database location. This directory name is masked for security purposes (for example, p1otxdt.prj). After you add workfiles to the project, this directory contains one file named pvcsproj.ser, which contains project metadata such as subprojects, versioned files, a configuration file, and archive and workfile locations.

To create a project:

1 Select the project database or project to which the new project will belong.

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 an asterisk (*), a colon (:), a vertical bar (|), forward and backward slashes (/ \), a question mark (?), and angle brackets (< >). No project name can be the two character name of .. or the one character name of . or @.

4 If you want the workfile location to be a location other than the default, specify the location in the **Workfile Location** field.

The default location is a subdirectory beneath the workfile location of the project or project database to which this new project belongs. For example, if you are creating a project named ProjectA beneath a project database that is located in `\myapps\ProjectDB1`, the default workfile location for the new project would be `\myapps\ProjectDB1\work\ProjectA`.

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.

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 You should specify a local drive that is available to most users. Remember that users can create a private workspace to change the workfile location to a different location.

Adding Workfiles

You can add a directory tree, a single directory, or individual files to a project database or project. When adding a directory, Version Manager automatically:

- Creates a project with the same name as the directory and adds the files in the directories to the project.
- Creates an archive for each file in the directory.
- Creates a versioned file that references the new archive.
- Creates subprojects when subdirectories are added and adds the files in the subdirectories to the subprojects.

By default, Version Manager uses a revision description of "Initial Revision" when the workfiles are checked in. You cannot change this revision description from the Add Workfiles dialog box.

Associating Issues

If you have installed Tracker TrackerLink or TeamTrack SourceBridge, you can associate issues (defects, change requests, tasks, etc) using TrackerLink or SourceBridge from the Add Workfiles dialog box using the **Associate Issues** feature. You may configure Version Manager to automatically invoke TrackerLink or SourceBridge to require that you associate issues when checking in files. See "Setting Up TrackerLink and SourceBridge" on page 105.

Initial defaults

Version Manager uses the following initial defaults when you add a workfile or directory to a project:

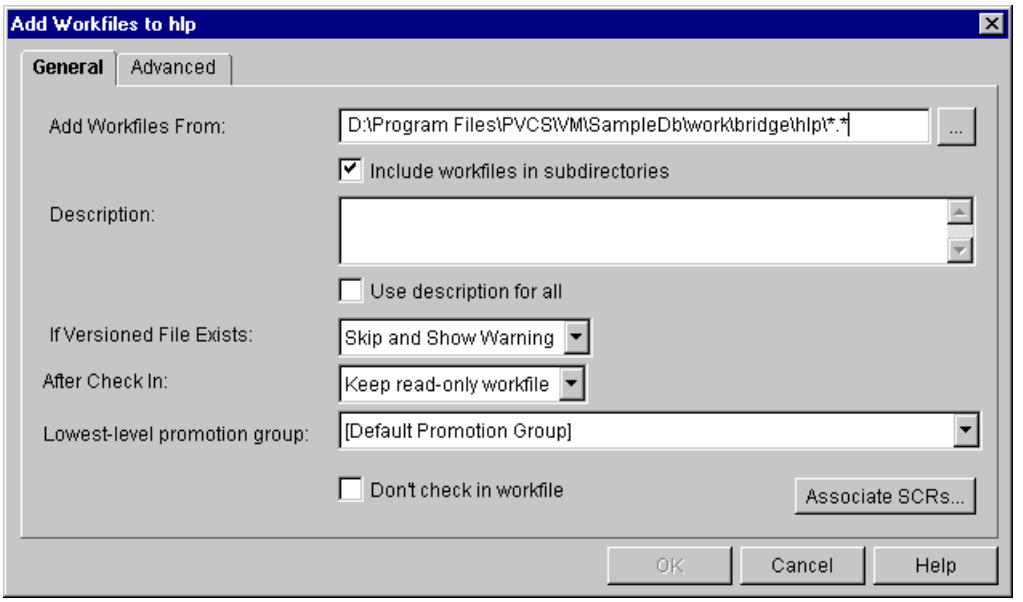
- Copies the workfiles from the selected directory to the workfile location of the project.
- Adds the files in subdirectories if you are adding a directory that has subdirectories.
- Does not add the workfile if a versioned file with the same name already exists in the project and displays a warning that the versioned file already exists.
- Keeps a read-only copy of the workfile—does not delete the workfile or keep the revision locked after adding the file.
- Does not assign a version label or create a branch.

- Checks in the workfiles to the archives. This default can only be changed if you have the SuperUser privilege. If you do not, the default is the only available option.

NOTE If you are adding workfiles to a 5.3/6.0 project, you must choose the archive directory in which you want to place the workfile archives from the **Archive Location** list.

To add workfiles:

1. Select the project database or project 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 are added directly without the creation of a top-level project. However, if subdirectories are also added, subprojects are created for them and only files matching the filter are added.

   b. (Only available if a directory is selected). 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.
   d. (For adding multiple workfiles or an entire directory only). To be prompted for a unique description for each file, deselect the **Use description for all** check box. Otherwise, Version Manager will use the same description for each file.

e In the **If Versioned File Exists** drop-down list, select the action that you want Version Manager to perform if a versioned file with the same name already 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** drop-down list, select the action that you want Version Manager to perform after it 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** drop-down list, select one of the following:

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

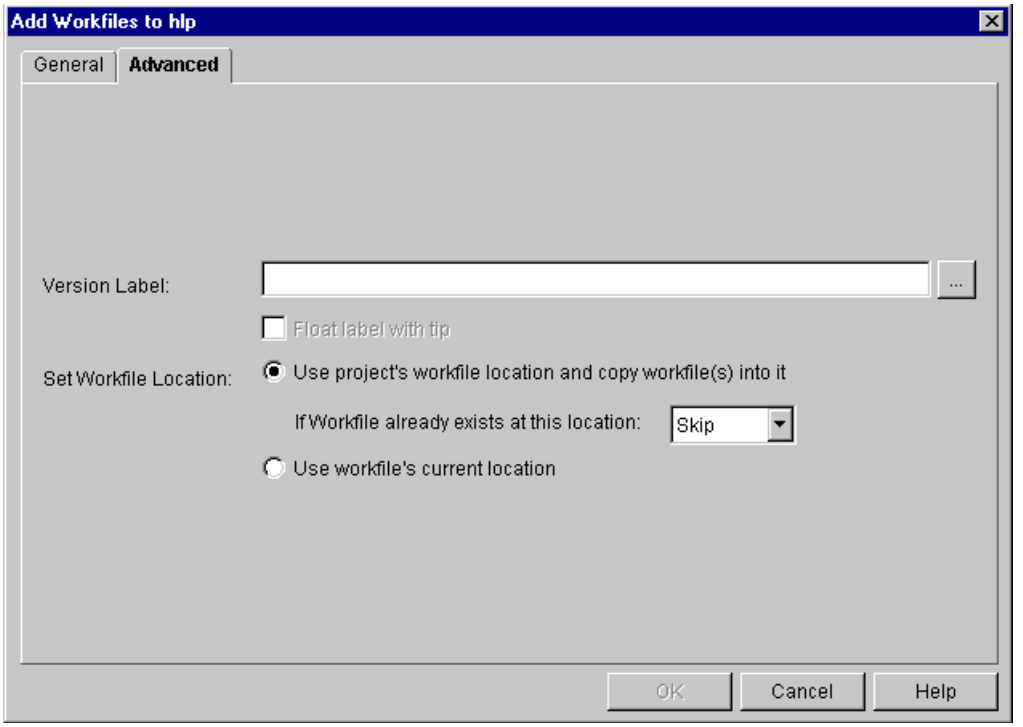
The default value is [Default Promotion Group], which is a workspace setting that defines a lowest-level promotion group to use for this action. If a value for this workspace setting is not defined and you do not select a value, then no promotion group is associated with the revision.

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 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 TrackerLink or SourceBridge association dialog box.

NOTE TrackerLink or SourceBridge is invoked automatically if you have set up Version Manager to require that you associate issues when adding workfiles. See "Setting Up TrackerLink and SourceBridge" on page 105.

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

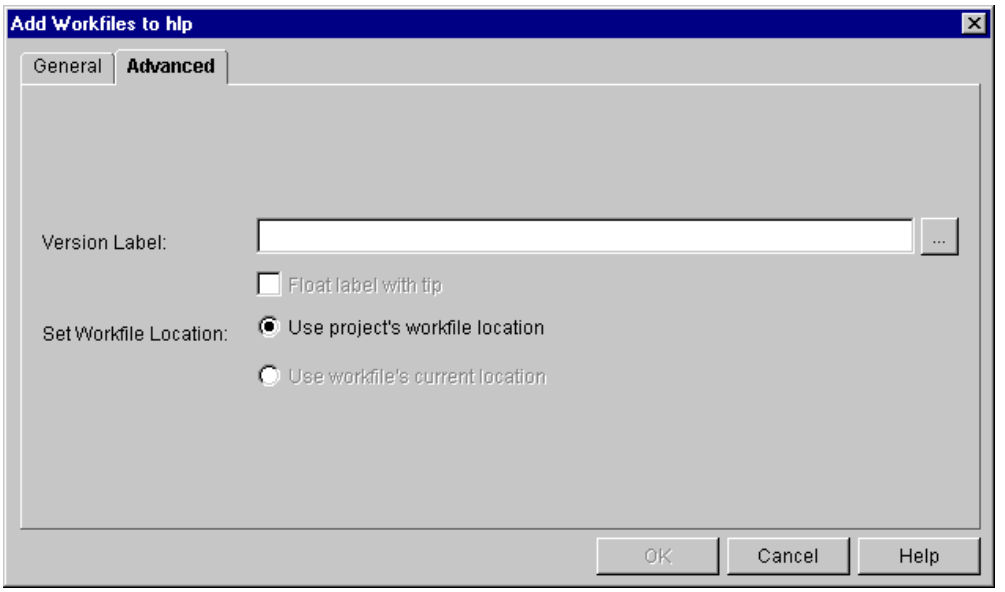
c Set the location of the workfile you are adding by choosing one of the following options (These options are not available if the **Don't check in workfile** check box is selected on the General tab or if the workfiles are already in the workfile location.):

- Select the **Use project's workfile location and copy workfile(s) into it** option to copy the file from its current location to the workfile location of the project. The workfile will use the workfile location of the project 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**.

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

NOTE The above options are available only 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**, as shown below.



5 Click **OK**.

Sharing Archives

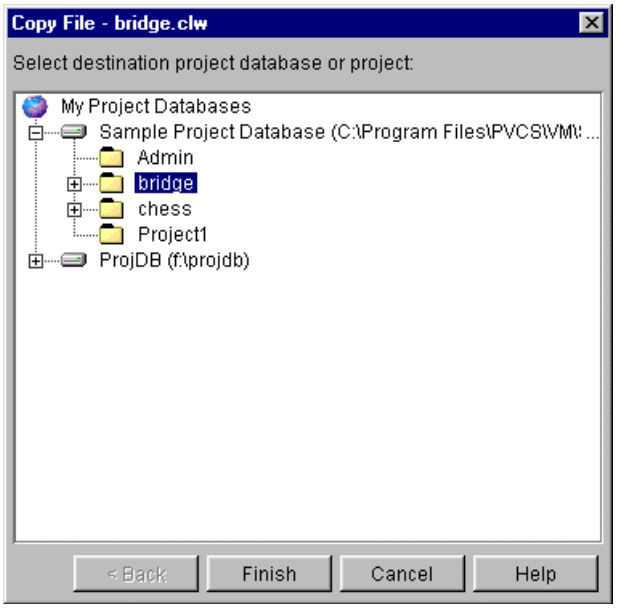
If you have files that are used by more than one project, you can share the archives of those files with the other projects. You do not need to create archives in each project for the shared files. Instead, you copy the versioned files to the other projects and use the archives in the existing location. The copied versioned files reference the original archives.

NOTE Archives are shared, not projects. If you eventually add a new file to the project, it will not be shared with any other project until you copy the versioned file to the desired project or projects.

To share archives:

1 Select the versioned file that you want to share (copy).

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



3 Select the destination project. This project will share the archive of the versioned file you selected in Step 1.

4 Click **Finish**.

Copying 5.3/6.0 Projects into Project Databases

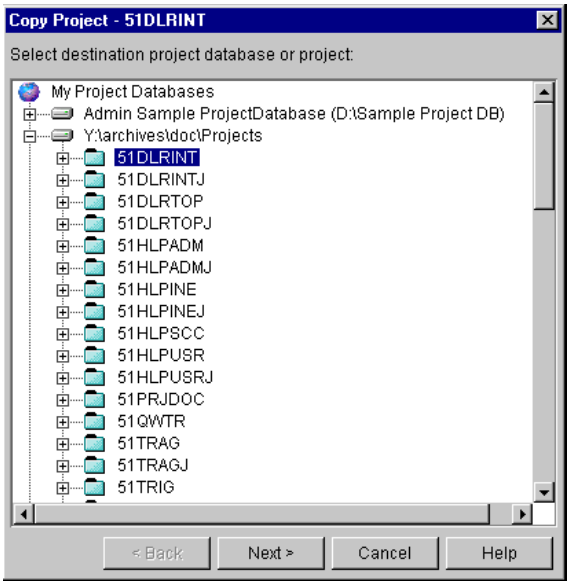
You can copy 5.3/6.0 projects into an existing project database, which upgrades the 5.3/6.0 project to the new format. You can also copy 5.3/6.0 project roots, and Version Manager upgrades the project root to a new project database. You do not need to create a project database before you copy the 5.3/6.0 project root.

You can also upgrade 5.3/6.0 projects that were created using the Version Manager Development Interfaces. For complete information about how to upgrade these projects, refer to the *Serena PVCS Version Manager Development Interface Implementation Guide* for your interface.

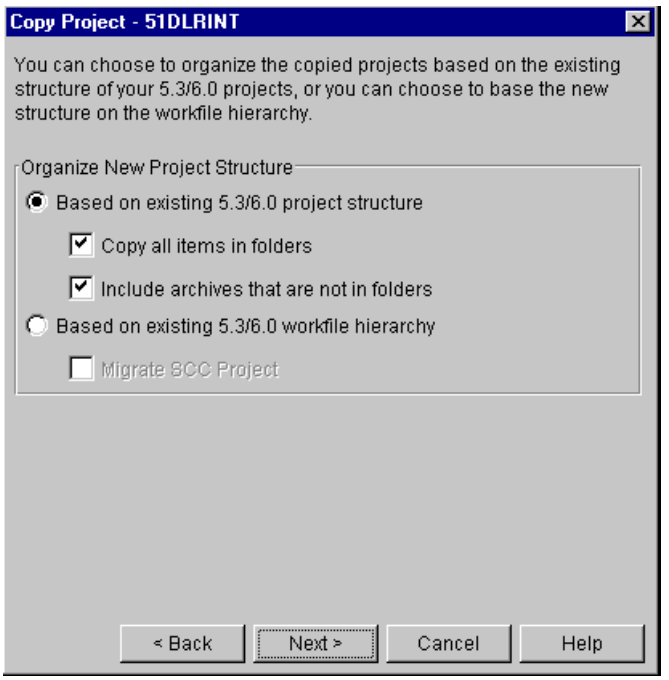
To copy 5.3/6.0 projects:

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

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



3 Select the project database or project within a project database to which you want to copy the project and click **Next**. A second Copy Project dialog box appears.



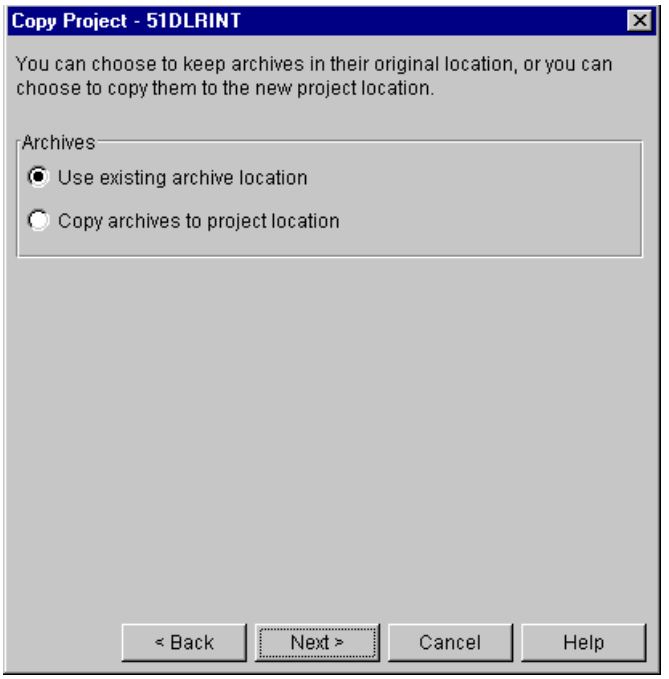
4 Organize the new project structure by selecting one of the following:

- **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, including those stored in folders in the archive location, check the **Copy all items in folders** check box. This option is selected by default.

- 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 option is selected by default.

- **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 use the **Based on existing 5.3/6.0 workfile hierarchy** option.

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



6 In the Archives group, select one of the following:

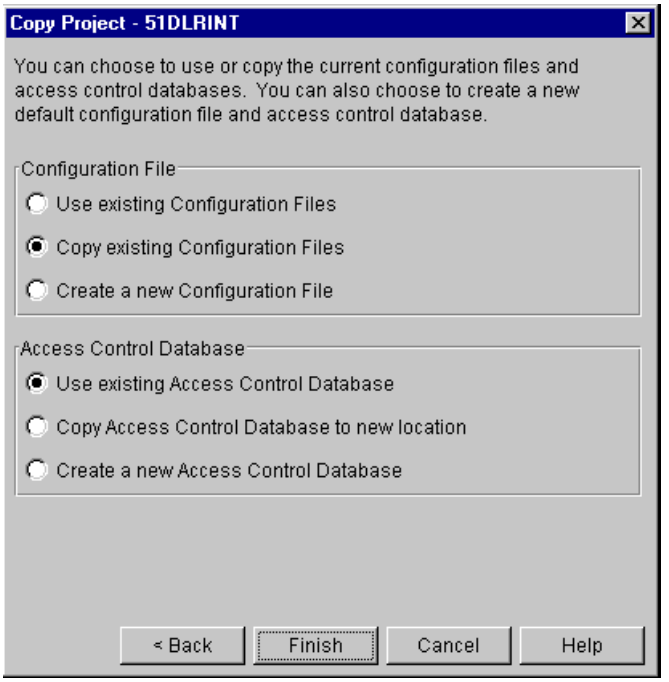
- **Use existing archive location** to continue referencing the existing archives in their current location, which means the copied project will share the archives of the 5.3/6.0 project. This is the default option.

  Version Manager creates a new archive directory for this project beneath the archives directory for the project database (for example, `prjdb\archives\copiedprj`). Any new files that you add to the copied project will reference the project database's archive location, not the 5.3/6.0 project's archive location.

- **Copy archives to project location** to make a copy of the archives and place them into a new Archives directory under the new project. The original archives are kept in the current location.

NOTE (For users migrating SCC projects). If other interface development 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 archives' association with the other projects.

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



8 If your project does not use configuration files, then the options in the **Configuration File** group are grayed. If your project uses configuration files, select one of the following:

- **Use existing Configuration Files** to continue referencing existing configuration files, thereby, sharing the existing configuration files with the 5.3/6.0 projects and the new, copied projects. We do not recommend that you use existing configuration files if any of the projects use an access control database.

  When you choose this option, the Access Control Database options are grayed.
- **Copy existing Configuration Files** to make copies of the existing configuration files and place them in the new project's structure. This is the default option.
- **Create a new Configuration File** to create a default configuration file for the project, which has no settings defined.

9 If you chose to use an existing configuration file in the Configuration File group, then the options in the **Access Control Database** group are grayed. If not, you can select one of the following:

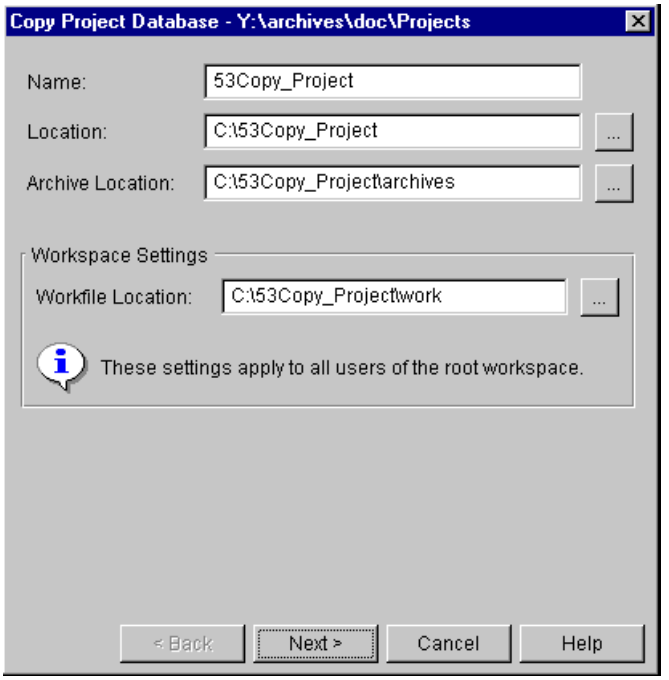
- **Use existing Access Control Database** to continue referencing existing access control databases. This is the default.
- **Copy Access Control Database to new location** to make a copy of all existing access control databases and place them in the new project's structure.
- **Create a New Access Control Database** to create a new access control database. By default, a new access control database contains one user with your user ID and the SuperUser privilege set assigned to you.

10 Click **Finish**.

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 Specify a name for the new project database in the **Name** field. The name cannot begin or end with a tab or blank space; all characters are valid.

4 Specify the location of the project database in the **Location** 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. Also, you cannot create a project database in the same location as an existing 5.3/6.0 project root.

5 If you want the archive location to be a location other than the default, which is a directory named archives beneath the project database location, specify the location in the **Archive Location** field. This location must be accessible to all users who will be accessing the archives.

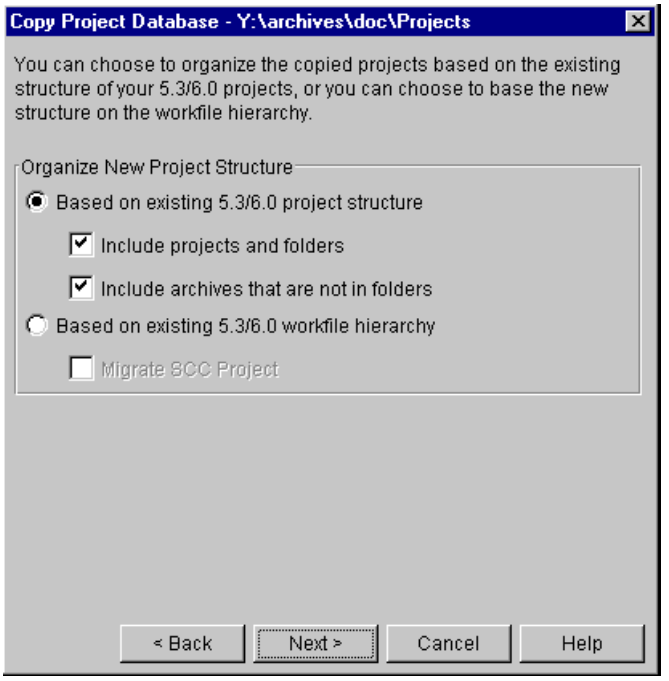
6 Specify the location in the **Workfile Location** field. This field is required; you will not be able to go to the next dialog box until you have entered a workfile location.

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. 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 create a private workspace to change the workfile location to a different location if they want.

7 Click Next. A second Copy Project Database dialog box appears.



8 Organize the new project database structure by selecting one of the following:

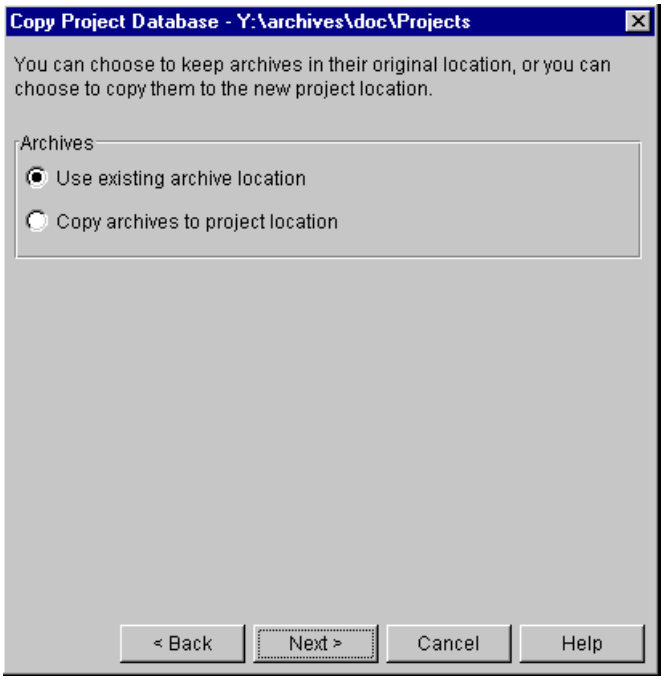
- **Based on existing 5.3/6.0 Project structure** if you want to organize the new project database based on the structure of the existing 5.3/6.0 project root. This structure will contain no more than two levels; a project and a subproject. This is the default option.

  You can modify this structure later by moving projects into other projects. When you move the projects into other projects, the moved projects retain their references to archive and workfile locations.

  - If you want to copy all archives, including those stored in folders in the archive location, check the **Include projects and 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. This option is selected by default.
  - 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 option is selected by default.

- **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 use the **Based on existing 5.3/6.0 workfile hierarchy** option.

9 Click Next. A third Copy Project Database dialog box appears.



10 In the Archives group, select one of the following:

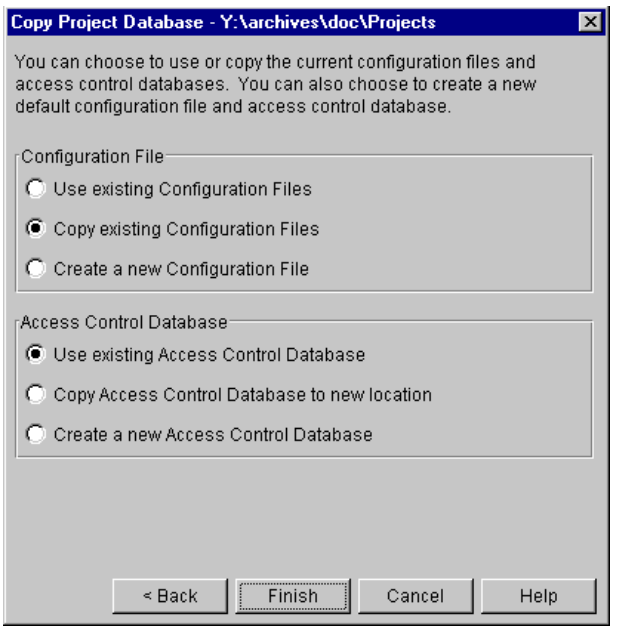
- **Use existing archive location** to continue referencing the existing archives in their current location. This is the default option.

  Version Manager creates an Archives directory for the project database with individual directories for each project of the project database (for example, `prjdb\archives\projecta`, `prjdb\archives\projectb`, etc.).

  Any new files that you add to the projects of the copied project database will reference the project database's archive location, not the 5.3/6.0 project root's archive location.

- **Copy archives to project location** if you want to make a copy of the archives and place them in a new archives directory under the new project database.

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



12 If your project database does not use configuration files, then the options in the **Configuration File** group are grayed. If your project database uses configuration files, select one of the following:

- **Use existing Configuration Files** to continue referencing existing configuration files, thereby, sharing the existing configuration files of the 5.3/6.0 project root with the new project database. We do not recommend that you use existing configuration files if the project root or any of the projects in the project root use an access control database.

  When you choose this option, the Access Control Database options are grayed.

- **Copy existing Configuration Files** to make a copy of the existing configuration files and place them in the new project database's structure. This is the default option.

- **Create a new Configuration File** to create a default master configuration file.

13 If you chose to use existing configuration files in the Configuration File group, then the options in the **Access Control Database** group are grayed. If not, you can select one of the following:

- **Use existing Access Control Database** to continue referencing existing access control databases. This is the default.

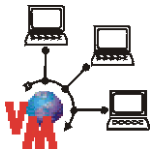
- **Copy Access Control Database to new location** to make a copy of all existing access control databases and place them in the new project database's structure.

- **Create a new Access Control Database** to create a new access control database. By default, a new access control database contains one user with your user ID and the SuperUser privilege set assigned to you.

14 Click Finish.

Setting Up Version Manager for Cross-Platform Use

This section describes how to configure Version Manager and assign privileges to directories and files for sharing archives, configuration files, access control databases, and project databases between UNIX and Windows users.



NOTE If you are using the Version Manager File Server for all archive access, only certain parts of this section apply. These parts are denoted by the graphic to the left. For more information on the Version Manager File Server, see "Configuring and Using the Version Manager File Server" on page 129.

The instructions in this section assume that you:

- Have installed Version Manager on both your Windows and UNIX systems
- Are using NFS or Samba for sharing file systems between UNIX and Windows

NOTE If users are running Version Manager from Windows but are storing archives on UNIX, all user IDs for Windows users must belong to the same UNIX primary group because of Windows NFS translation.

The configuration files you create for your Version Manager implementation must contain Windows paths. The nfsmap file will map the Windows drive letters or UNC paths to their corresponding UNIX path names so that the same configuration files can be used by both UNIX and Windows.

For information on using setuid on UNIX within a cross-platform environment, see the *Serena PVCS Version Manager Installation Guide*.

The following basic steps provide an overview for setting up Version Manager for cross-platform use. Following this list are sections that provide detailed procedures.

1. On UNIX, we suggest that you create a Serena Administrator user ID such as *pvcs* for you to use and create (if not already created) a group that contains all of the PVCS users, such as *pvcsgrp*. Make this group your primary group or change the group ownership of each PVCS file and directory to the PVCS group.

   The Serena Administrator should own all of the directories that contain configuration files, access control databases, the nfsmap file, and the Version Manager executables and the files within these directories.

   Refer to the documentation for your operating system for more information on creating user IDs and groups.

2. To use Version Manager in a cross-platform environment, you must turn off setuid mode.

3. On UNIX, edit the nfsmap file to map the Windows drive letters to the corresponding UNIX path names so that users on both Windows and UNIX systems can share configuration files.

NOTE

- This step must be performed before you create archives.
- There is a limit of 26 active (uncommented) entries in the nfsmap file. Additional entries are silently ignored.

4 Assign the proper privileges on UNIX and Windows:

- On UNIX:

For...	Assign these privileges...	To...
PVCS users and admin	read, write, execute	Directories that are specified in the configuration files
PVCS users	read, execute	Directories that contain the master configuration file, the access control databases, and the nfsmap file
PVCS users	read	The master configuration file, access control database, and nfsmap file
NOTE If you want Version Manager to create users in the access control database automatically, and if you want users to change their own passwords, PVCS users must have read and write privileges to the access control database.		
PVCS admin	read, write, execute	Directories that contain the master configuration file, the access control databases, and the nfsmap file
PVCS admin	read, write	The master configuration file, access control database, and nfsmap file

- In Windows:

For...	Assign these privileges...	To...
PVCS users and admin	full permissions	Directories that are specified in the configuration files
PVCS users	read	Directories that contain the master configuration file, the access control databases, and the files within the directories
NOTE If you want Version Manager to automatically create users in the access control database, and if you want users to change their own passwords, PVCS users must have read and write privileges to the access control database.		
PVCS admin	full permissions	Directories that contain the master configuration file, the access control databases, and the nfsmap file, and the files within the directories

5 In Windows, create the project databases that you want to share between platforms. You **must** create the project databases using the Version Manager desktop client on Windows. If you create the project databases using the Version Manager desktop client on UNIX, Windows users cannot access them.

6 Configure Version Manager to:

- Make newly created and existing archives writable.

- Translate the end-of-line (EOL) sequence for newly created and existing archives that store text files.
- Do **not** translate the EOL sequence for newly created and existing archives that store binary files to prevent data corruption.
- Make user IDs case-insensitive so that the UNIX and Windows systems will read the user IDs the same way.

7 Make the case of file and directory names consistent for files and directories shared between Windows and UNIX systems.

Turning Off Setuid

On UNIX, use the following command to turn off setuid permissions for Version Manager executables:

`chmod u-s` *`executable`*

where *`executable`* is the following:

get	regen	vjournal
ident	vcompres	vlog
vmrg	put	vcs
vpromote	pvcsvmsuid	vdel
vsql	vdiff	

Editing the nfsmap File

Version Manager provides the `nfsmap` file to enable users to share configuration files between Windows and UNIX systems. The `nfsmap` file is a text file that maps Windows drive letters or UNC path names to their corresponding UNIX path names. The file is stored on UNIX. When this file is available, Version Manager uses the MS-DOS path format when reading or writing path names from or to a configuration file.

NOTE You must complete this procedure before you create your archives.

On UNIX, to edit the nfsmap file:

1 Open the `nfsmap` file in a text editor. The default location for the `nfsmap` file is:

`Install_Location``/vm/common/bin/`*`OS`*`/nfsmap`

Where *`OS`* is the name of the operating system. For example, on Solaris, the path would be:

`Install_Location``/vm/common/bin/solaris/nfsmap`

2 Enter each mapping as two columns on its own line, with the drive letter or UNC path name in the left column and the corresponding UNIX directory in the right column. If the UNC path name contains spaces, then it must be enclosed in double quotes (").

Example 1: To map the `M:` drive to the UNIX directory `/product/dev`, enter:

```
M    /product/dev
```

Example 2: To map the UNC path name `"\\myserver\vol2 abc"` to the UNIX directory `/product/dev`, enter:

```
"\\myserver\vol2 abc"    /product/dev
```

IMPORTANT! Each drive letter, UNC path, and UNIX directory must be unique. You cannot repeat a drive letter, UNC path, or UNIX directory in another mapping.

3 Save your changes and exit the editor.

Assigning Privileges

You must assign the proper privileges to users on both UNIX and Windows so that:

- Users can read and write archives but only read the master configuration file, access control databases, and the nfsmap file (the nfsmap file is on UNIX only).

NOTE If you want Version Manager to automatically create users in the access control database, and if you want users to change their own passwords, PVCS users must have read and write privileges to the access control database.

- Administrators can read and write archives, the master configuration file, access control databases, and the nfsmap file (the nfsmap file is on UNIX only). Users should **not** have write access to the master configuration file, access control databases, and the nfsmap file because they could reconfigure Version Manager.

To assign privileges on UNIX and Windows:

1 On UNIX, assign read, write, and execute privileges to PVCS users and the Serena Administrator for directories (and applicable subdirectories) that are specified in the configuration files, which could include the:

- Archive directory (VCSDir)
- Semaphore directory (SemaphoreDir)
- Directory that contains internal temporary files (WorkDir)
- Temporary archive file directory (ArchiveWork)
- Reference directories (ReferenceDir and VCSDir)
- Journal file directory (Journal)

For each directory, enter the following command:

```
chmod ug+rwx directory
```

2 On UNIX, assign the read and execute privileges to PVCS users for directories that contain Version Manager configuration files, access control databases, and the nfsmap file. Also, assign read, write, and execute privileges to the Serena Administrator for these directories.

For each directory, enter the following command:

```
chmod ug+rx,u+w directory
```

3 On UNIX, assign only the read privilege to PVCS users for the files listed in Step 2. Also, assign read and write privileges to the Serena Administrator for these files.

Navigate to each directory and enter the following command for each of these files:

```
chmod ug+r,u+w file_name
```

4 In Windows, assign full permissions to PVCS users and the Serena Administrator for directories that are specified in your configuration file, which could include the:

- Archive directory (VCSDir)
- Semaphore directory (SemaphoreDir)
- Work directory (WorkDir)
- Archive work directory (ArchiveWork)
- Reference directories (ReferenceDir and VCSDir)
- Journal file directory (Journal)

5 In Windows, assign only the read privilege to PVCS users for directories that contain Version Manager configuration files and access control databases and the files in those directories. Users should not be allowed to change these files.

6 In Windows, assign full permissions to the Serena Administrator for the files and directories specified in Step 5; the Administrator must be allowed to changes these files.

Configuring Version Manager

In this section, you will configure Version Manager to:

- Make archives writable.
- Translate the end-of-line (EOL) sequence for text files and **not** translate the EOL sequence for binary files.
- Make user IDs case-insensitive.

Making Archives Writable

When sharing archives between Windows and UNIX, the archive files must be made writable. In Windows, you must do the following:

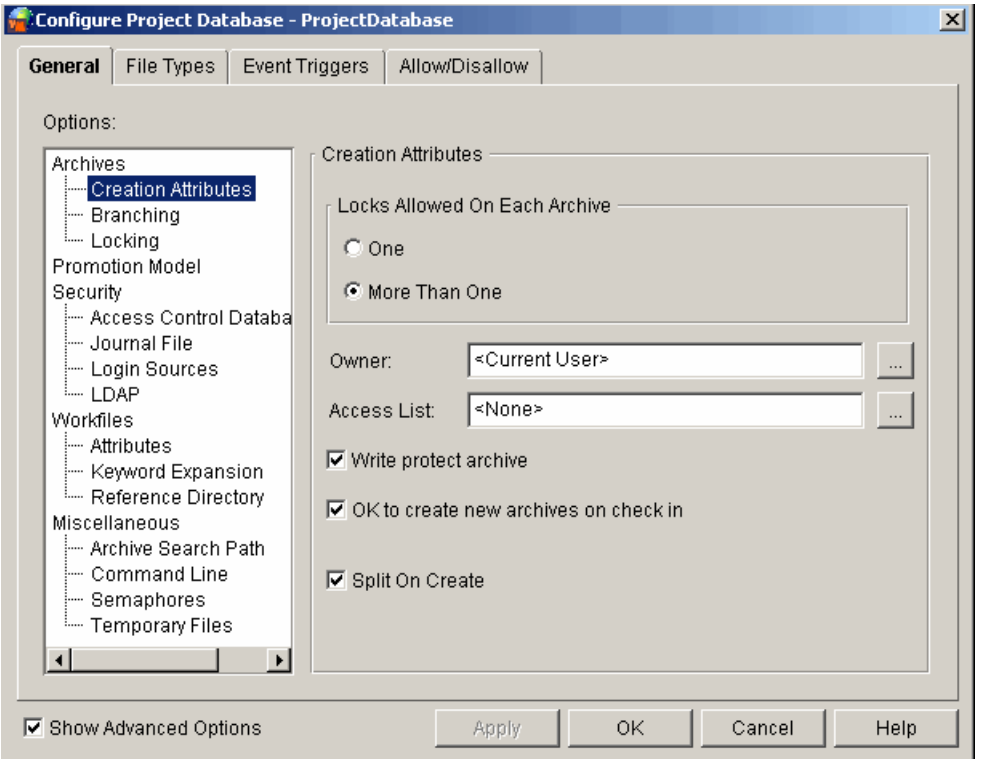
1 Configure each project database to make newly created archives writable by doing the following:

a Select the project database that you want to configure.

b Select Admin | Configure Project. The Configure Project dialog box appears.

c If not already selected, select the **Show Advanced Options** check box.

d Select Creation Attributes beneath Archives. The Creation Attributes pane appears on the right.

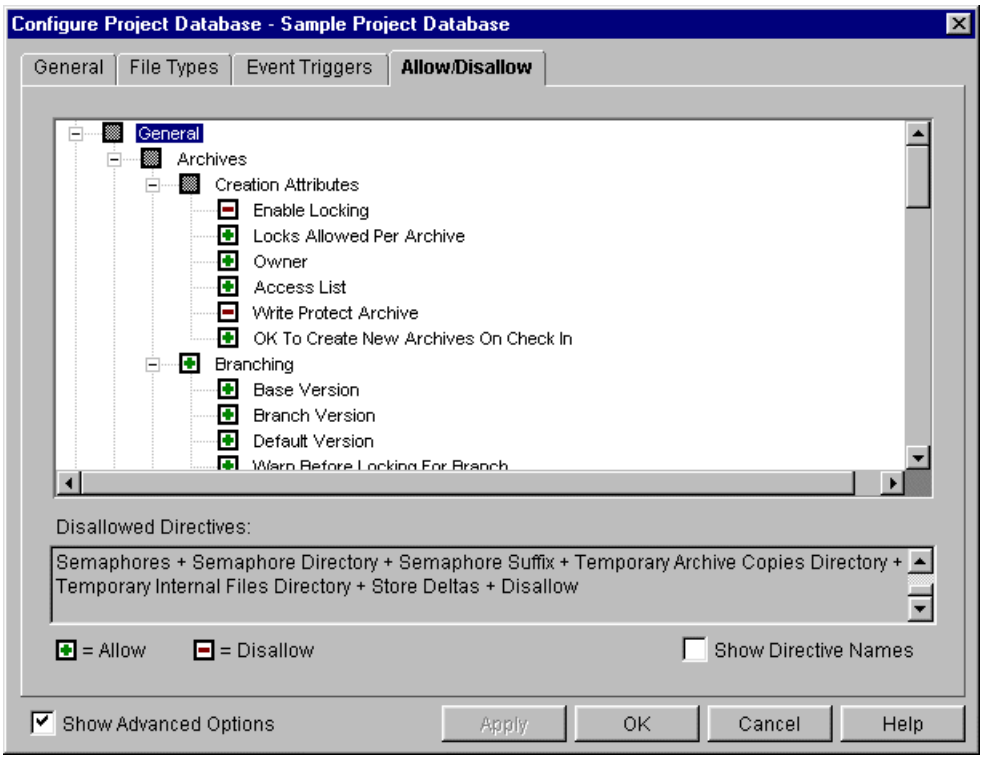


e Deselect the **Write protect archive** check box.

f Click Apply.

2 Configure each project database so that users cannot change the setting of the **Write protect archive** option. To disallow the user from changing this option, do the following:

a Select the Allow/Disallow tab. This tab displays a check box tree of configuration options that you can allow and disallow.



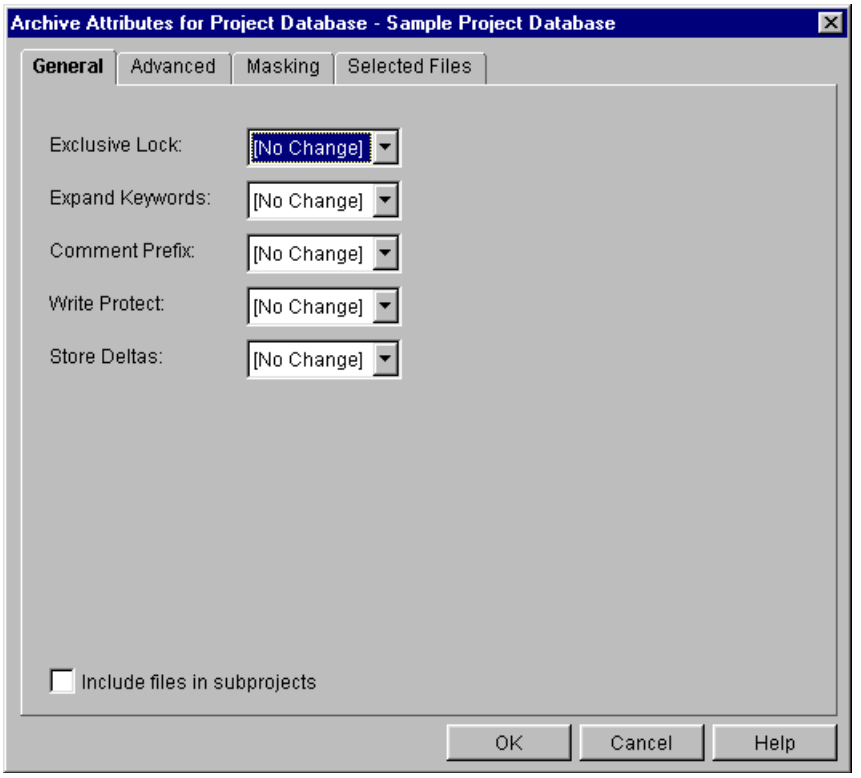
b If the **Write Protect Archive** option has a plus sign (+) in the check box, click the check box to place a minus sign (−) in the check box. The minus sign (−) indicates that the option is disallowed.

c Click OK.

3 Make all existing Version Manager archives in a project database writable by doing the following:

a Select the project database that has the existing archives that you want to make writable.

b Select Admin | Archives Attributes. The Archives Attributes dialog box appears with the General tab active.



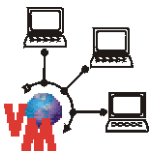
c Select **No** for **Write Protect**.

d Select the **Include files in subprojects** check box to change the attribute for existing archives in all of the projects/subprojects of the project database.

e Click OK.

4 Repeat Steps 1 through 3 for each project database that you want to share across platforms.

Translating the EOL Sequence



To share archives on UNIX and Windows, you must configure each project database to translate the end-of-line (EOL) sequence on UNIX from line feed to carriage return plus line feed when you check in a revision **for text files**, and to make the same translation in reverse when you check out a revision.

CAUTION! You must not translate the EOL sequence for binary files; doing so may corrupt your workfiles.

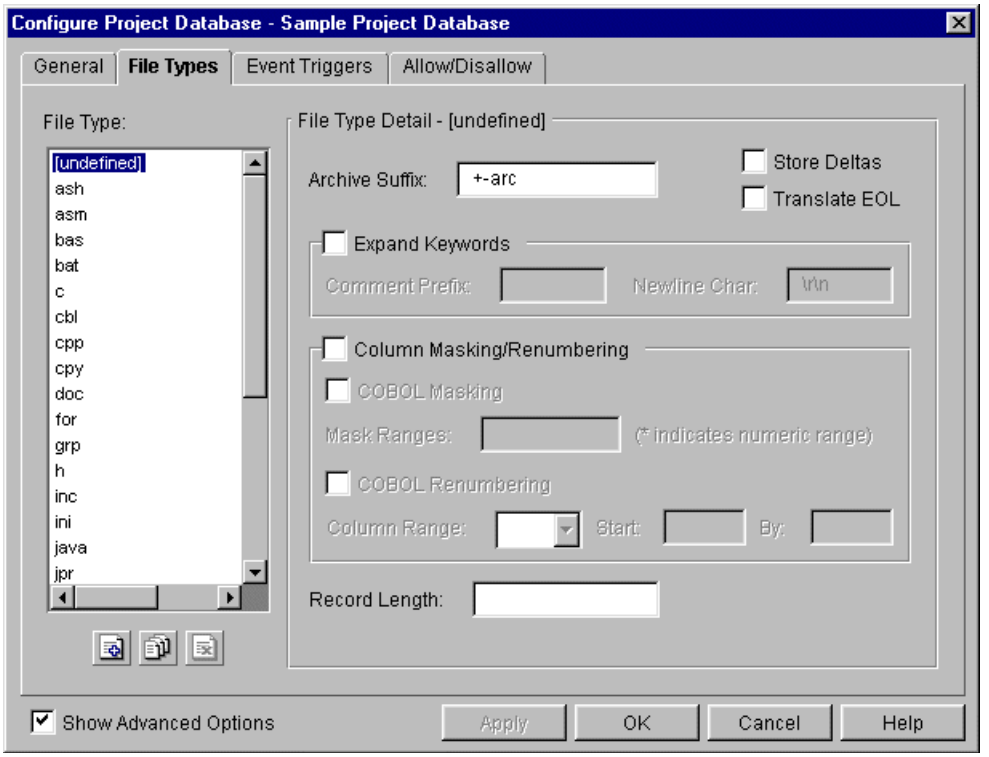
To translate the EOL sequence for text files:

1 Configure each project database to make newly created archives translate the EOL sequence when checking in or out text files. Do the following:

a Select the project database that you want to configure.

b Select Admin | Configure Project. The Configure Project dialog box appears with the General tab active.

c Select the File Types tab.



d In the **Files Type** list on the left, select the file type (a text file extension) for which you want to translate the EOL sequence.

e Select the **Translate EOL** check box.

f Repeat Steps d and e for each text file extension. Then, click OK.

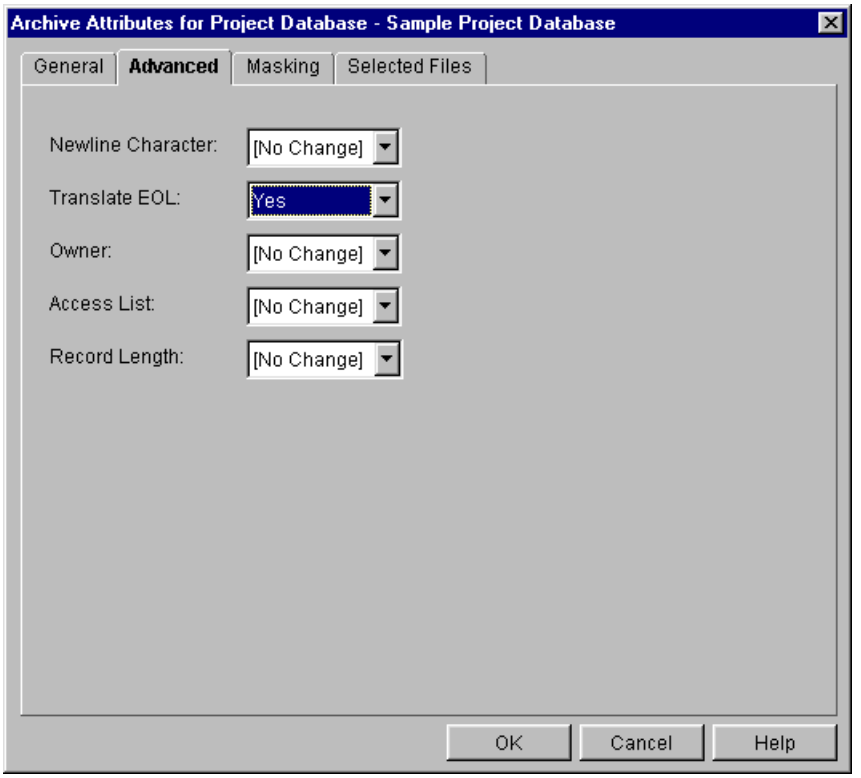
2 Make all existing archives of a project database translate the EOL sequence for text files by doing the following:

a Use the Version Manager File Filter (View | Filter | Wild Card) to display all versioned files that have an extension that indicates the file is a text file. For example, you can display all files with the extension .txt.

b Select the versioned file(s) that were filtered in Step a.

c Select Admin | Archive Attributes. The Archive Attributes dialog box appears with the General tab active.

d Click the Advanced tab. On this tab, select **Yes** for **Translate EOL**.

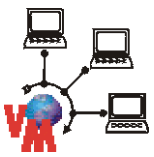


e Click **OK**. Translate EOL is enabled for all of the selected files.

f Repeat Step 2 for text files of other types.

3 By default, Version Manager does not translate the EOL sequence for binary files. Therefore, you should not have to turn off this option for binary files. However, if you have any doubt that this option might be turned on for binary files, it is essential that you turn it off.

Making User IDs Case-Insensitive



For Version Manager to read user IDs the same on UNIX and Windows systems, you must configure each project database to make user IDs case-insensitive.

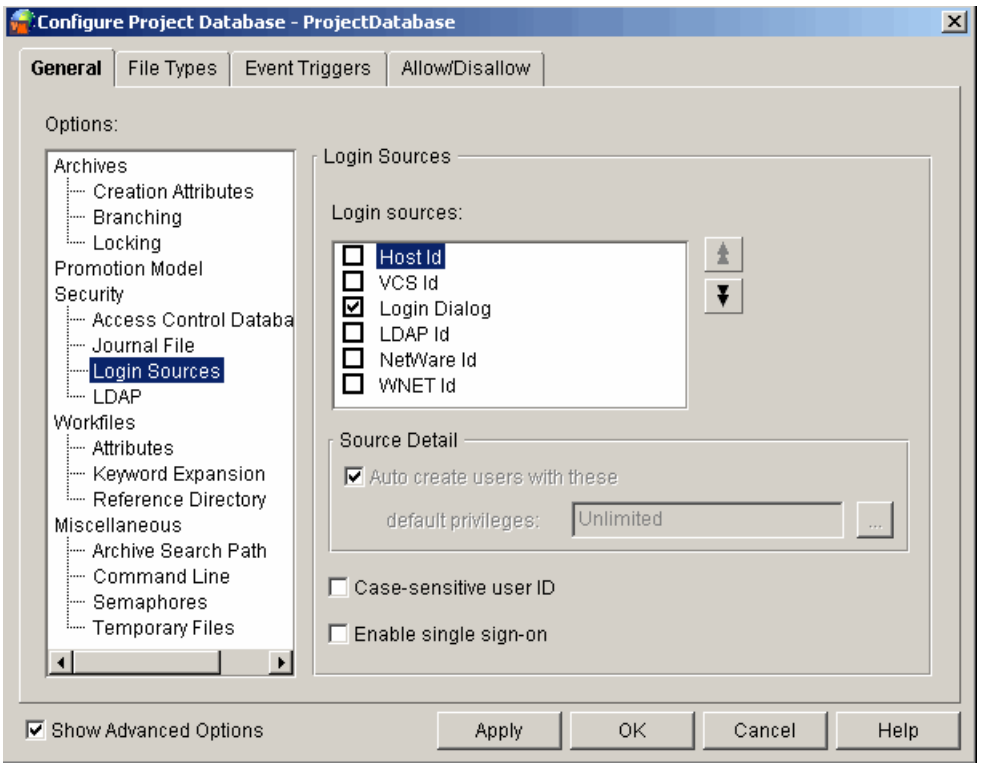
To configure each project database to make user IDs case-insensitive:

1 Select the project database that you want to configure.

2 Select Admin | Configure Project. The Configure Project dialog box appears.

3 If not already selected, select the **Show Advanced Options** check box.

4 Select Login Sources beneath Security. The Login Sources pane appears on the right.

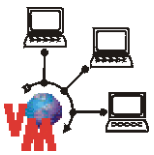


5 Deselect the **Case Sensitive User ID** check box.

6 Click OK.

7 Repeat Steps 1–6 for each project database that you want to share across platforms.

Making the Case of File and Directory Names Consistent



To make the case of file and directory names consistent for files and directories shared between Windows and UNIX, we recommend that you use lowercase for all names. In that case, UNIX will be able to read the names.

If you create the new archives on Windows, use the VCS command to change the workfile names in the archives to reflect the proper case.

IMPORTANT! Before using Windows clients with a UNIX server, be sure to look for any files or projects that are at the same level but differ only by case.

What's Next

After you set up a Version Manager project database, you can:

- Configure project databases and projects. See "Configuring Version Manager" on page 51.
- Add security. See "Using Security" on page 203.
- Define a promotion model. See "Using Promotion Models" on page 247.
- Define event triggers. See "Using Event Triggers" on page 275.

Chapter 2

Configuring Version Manager

Introduction

After you have a basic project model in place as discussed in "Setting Up Version Manager Using the Desktop Client" on page 13 or "Setting Up Version Manager Using the Command-Line Interface" on page 113, you can define how you want Version Manager to operate if the default way Version Manager operates does not suit your needs. "Default Settings in the Desktop Client" on page 54 explains Version Manager's default behavior in the desktop client and "Default Settings when No Configuration File Is Used" on page 56 explains Version Manager's default behavior in the command-line interface.

You can configure Version Manager at the project database and/or project level, depending on your needs. For example, to make:

- All project databases operate the same, embed a configuration file into Version Manager.
- All projects in a particular project database operate the same, configure the configuration file associated with the project database rather than at the project level.
- Some projects operate differently in a project database, create specific configuration files for those projects. This allows you to, for example, define a promotion model for some projects and not for others.

This chapter first explains configuration options and provides a table with the available options. Second, the initial Version Manager defaults for configuration options are given so that you can decide if this model suits your needs. Third, this chapter explains configuration files and how Version Manager uses them. Finally, this chapter provides the available settings for all of the project configuration options and how to set them using the desktop client and the command-line interface.

About 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 which access control database is used, if any.

Configuration options are set in a configuration file, and you can set configuration options in one configuration file or multiple configuration files.

For more information about configuration files, see "Understanding Configuration Files in the Desktop Client" on page 58 and "Understanding Configuration Files in the Command-Line Interface" on page 62.

Table 2-1 lists the types of available configuration options.

Table 2-1. Configuration Options

Use these options...	To do this...
Archive Creation	■ Exclusively lock an archive. ■ Assign ownership of archives. ■ Identify users and groups of users who are allowed to access archives. ■ Write-protect archives. ■ Automatically create an archive when you check in a workfile if Version Manager cannot find one for the file.
Branching	■ Specify the version label that identifies the revision where a branch begins. ■ Specify the version label assigned to branch revisions. ■ Identify the revision to use if no other is specified. ■ Warn the user before a branch is created.
Locking	■ Permit a user to lock more than one revision. ■ Permit more than one lock per revision. ■ Remove the lock when storing an unchanged revision. ■ Enforce pessimistic locking on clients capable of optimistic locking.
Promotion Model	■ Define a promotion model.
Access Control Database	■ Identify the access control database. ■ Enable the use of the access control database.
Journal File	■ Record an audit trail of archive activity.
Login Sources	■ Specify the user identification sources. ■ Specify whether or not users are automatically created in the access control database and, if so, the default privileges assigned to each user (desktop client only).
Workfile Attributes	■ Delete workfiles after storing them. ■ Keep read-only copies of workfiles after storing them. ■ Keep writable copies of workfiles after storing them.
Keyword Expansion	■ Specify the path separator character used in keyword expansion. ■ Update the timestamp when expanding keywords.
Reference Directory	■ Specify the directory where current workfile copies are automatically stored. ■ Keep a reference directory for each archive directory. ■ Store trunk revisions only. ■ Write-protect the files in the reference directory.

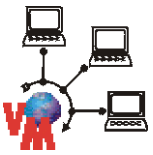
Use these options...	To do this...
Archive Search Path (Command-line interface options only)	■ Specify where Version Manager searches for archives when you perform an action on a file. ■ Specify whether or not wildcard specifications match archives in directories other than the first archive directory in which they are found. ■ Specify how Version Manager searches for archives when only a workfile name is specified.
Command Line (Command-line interface options only)	■ Display sign-on messages. ■ Display detailed messages. ■ Specify a command-line editor. ■ Specify the directory for temporary files. ■ Specify the message file suffix. ■ Delete message files after reading them.
Semaphores	■ Specify the directory where semaphore files are created. ■ Specify the suffix used for semaphore files. ■ Specify the number of attempts to access an archive. ■ Specify the delay between attempts to access shared archives.
Temporary Files	■ Specify the directory for temporary archive files. ■ Specify the directory for temporary internal files.
File Types	■ Specify an archive suffix. ■ Store deltas. ■ Translate the end-of-line character. ■ Expand keywords. ■ Specify the comment prefix to use when expanding keywords. ■ Specify the newline character to use when expanding keywords. ■ Specify columns to ignore when comparing files. ■ Renumber masked columns. ■ Specify the logical record length.
Event Triggers	■ Execute user-defined instructions before or after a Version Manager event.
Allow/Disallow	■ Disallow users from changing a configuration option setting.

Default Settings in the Desktop Client

When a project database is created, by default, a master configuration file is created and associated with the project database.

Version Manager creates this master configuration file by copying the default.cfg file, which is installed when you install Version Manager. By default, this file is placed in the following location during Version Manager installation:

- In Windows:
  drive:\Program Files\Serena\vm\common\pvcsprop\pvcs\vm
- On UNIX: /usr/serena/vm/common/pvcsprop/pvcs/vm



NOTE For project databases created on a file server, the default configuration file is named defaultfs.cfg.

You can modify the settings in the default.cfg file to suit your needs, and then Version Manager will use the modified file to create new configuration files.

If you do not change any settings for the configuration options in this file and do not define any project configuration files, Version Manager will operate according to the settings in this configuration file. You can check the settings of the file by selecting the project database and selecting Configure | Project or by opening the file in a text editor. The settings in the Configure Project Database dialog box reflect the settings in the configuration file. Version Manager will operate as follows:

- The archive suffix is +-arc (for example, text.txt-arc). See "Changing the Archive Suffix" on page 97.
- Archives are automatically created on check in of a workfile if one does not exist.
- Archives are write-protected.
- The user is warned that a branch will be created.
- Workfiles are **not** deleted upon check in.
- Keywords are **not** expanded in binary files. Keywords are expanded in text files with certain suffixes. This behavior applies to newly created archives only and does not affect archives that already exist.
- End-of-line characters are **not** translated in binary files. They are translated on UNIX text files with certain suffixes. This behavior applies to newly created archives only and does not affect archives that already exist.
- Multiple locks on a single revision are **not** allowed.
- A user can lock more than one revision of an archive.
- More than one revision in an archive can be locked.
- Optimistic locking is allowed in clients capable of this.
- File semaphores are specified.
- The semaphore suffix is +-sem (for example, text.txt-sem).
- An archive semaphore is created in the same directory in which the archive resides.
- The number of times Version Manager attempts to access an archive is six.
- The delay between attempts to access an archive is three seconds.
- **Non-file-server project databases:** The login source is Host. In the desktop client, Version Manager will automatically create users in the access control database if they do not exist and assign them the Unlimited privilege set.
  File-server project databases: The login source is VLOGIN. Users are not automatically added to the access control database.

- **Non-file-server project databases:** User IDs are **not** case-sensitive.
  File-server project databases: User IDs **are** case sensitive.
- The lock on an unchanged revision is removed upon check in if the check in is canceled.
- Revisions must be checked in by the user who locked the file.
- Revisions are **not** stored as a set of deltas for binary files. They are stored as a set of deltas for text files with certain suffixes. This behavior applies to newly created archives only and does not affect archives that already exist.
- The timestamp is **not** updated when Version Manager expands keywords upon check in.
- Copies of checked in workfiles are not placed in reference directories.
- Characters are not ignored after CTRL+Z.
- Automatic branching is not set up.
- **Non-file-server project databases:** No access control database is enabled.
  File-server project databases: An access control is enabled.
- A journal file is not kept.
- The forward slash (/) is specified as the separator character for keyword expansion.
- A revision must be locked before you can check in a workfile.

The following list applies to the operation of the command-line interface only:

- Message files are **not** deleted after reading them.
- The message file suffix is +-msg (for example, text.txt-msg).
- Sign-on messages are displayed.
- Detailed messages are displayed.
- VCSDir is set to the root of the project's archives and the directories beneath it are treated as possible archive directories.
- A specified workfile path is used to locate archives instead of VCSDir.
- A default editor is not specified.
- The default date format is 'mm/dd/yyyy hh:mm:ss'.
- The names of the months and the AM/PM symbols are 'January February March April May June July August September October November December AM PM.'

Default Settings when No Configuration File Is Used

There is no default configuration file associated with Version Manager in the command-line interface; there is, however, a default configuration file, default.cfg, shipped with the product. By default, this file is placed in the following location during Version Manager installation:

- In Windows: *drive*:\Program Files\Serena\vm\common\pvcsprop\pvcs\vm
- On UNIX: /usr/serena/vm/common/pvcsprop/pvcs/vm

To use this file as the master configuration file in the command-line interface, make a copy of the file, change any of the options you want, and embed the copied file into Version Manager using the VCONFIG command.

If you do not associate a configuration file with Version Manager when using the command-line interface or if you disassociate a configuration file from project databases and projects when using the desktop client, Version Manager operates as follows:

- The archive suffix template is ??V___ , for example, text.txv. See "Changing the Archive Suffix" on page 97.
- Archives are automatically created on check in of workfiles if they do not exist.
- Archives are write-protected.
- The current archive directory is where Version Manager looks for archives.
- A reference directory is **not** specified.
- The user is **not** warned that a branch will be created.
- Workfiles are deleted upon check in.
- Keywords are **not** expanded.
- End-of-line characters are **not** translated for all files types except: .c, .h, .pas, .mak, .for, .bas, .asm, .bat.
- Multiple locks on a single revision are **not** allowed.
- A user **cannot** lock more than one revision of an archive.
- More than one revision in an archive can be locked.
- The semaphore suffix template is ??$___ , for example, text.tx$.
- The number of times Version Manager will attempt to access an archive is three.
- The delay between attempts to access an archive is one second.
- The login source is Host.
- The lock on an unchanged revision is **not** removed upon check in.
- Archives are **not** compressed.
- Deltas are **not** compressed.
- The initial copy of a file (work image) that is placed in an archive is **not** compressed.
- Message files are **not** deleted after reading them.
- A revision must be locked before you can check in a workfile.
- Revisions must be checked in by the user who locked the file.
- Revisions are stored as a set of deltas.
- The workfile path is used when locating archives.
- The timestamp is updated when Version Manager expands keywords upon check in.
- Characters are **not** ignored after CTRL+Z.

Understanding Configuration Files in the Desktop Client

Version Manager stores configuration options in configuration files. The two types of configuration files for the desktop client are:

- Master configuration file, which contains configuration options for a project database and all of its projects.
- Project configuration file, which contains configuration options for a project that override the settings in the master configuration file, unless the option is disallowed 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.

The Version Manager desktop client uses a configuration file if one is:

- Associated with a project database or project. A configuration file associated with a project database is called a master configuration file.
- Embedded into Version Manager. See "Embedding a Master Configuration File into Version Manager" on page 103.

Master Configuration File

A master configuration file lets you standardize how Version Manager operates by identifying options that cannot be reset by other configuration files. When you disallow options in the master configuration file, this forces Version Manager to operate according to the settings in the master configuration file.

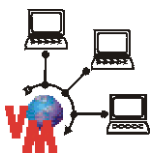
For example, if the master configuration file sets the option to automatically create an archive when checking in a new file and then disallows the option, you cannot change the option in other configuration files and, therefore, Version Manager automatically create archives. In other words, you set the option in the master configuration file and then disallow it so that other configuration files cannot override the option.

The Default Master Configuration File

Version Manager automatically creates a master configuration file for each project database you create. You can change the settings in this file to meet your needs. For security purposes Version Manager masks the name of this configuration file. For example, this file may be assigned a name such as "c6bonpj1.cfg." The file is located, by default, in the archives directory.

Version Manager creates this master configuration file by making a copy of the default.cfg file, which is installed when you install Version Manager. By default, this file is placed in the following location during Version Manager installation:

- In Windows: *drive*:\Program Files\Serena\vm\common\pvcsprop\pvcs\vm
- On UNIX: /usr/serena/vm/common/pvcsprop/pvcs/vm



NOTE For project databases created on a file server, the default configuration file is named defaultfs.cfg.

You can modify the settings in the default.cfg file to suit your needs, and then Version Manager will use the modified file to create new master configuration files.

Embedding a Master Configuration File

Optionally, you can embed one master configuration file into Version Manager. This configuration file will control how Version Manager operates for all project databases. When you embed a master configuration file into Version Manager, this file becomes the master configuration file, and the configuration file that you have associated with each project database becomes a project configuration file. The project configuration file is no longer a master configuration file; therefore, any Disallow options in the project configuration file are ignored.

NOTE Do not embed the default.cfg file into Version Manager because this is the file that Version Manager uses to create the configuration file for project databases. If you do this, the embedded master configuration file and the project databases' configuration files will initially have the same settings.

Embedding the configuration file into Version Manager ensures that all users will be using the same configuration for Version Manager and that users cannot use a different master configuration file.

A master configuration file that is embedded into Version Manager affects all desktop client and command-line users who are using the copy of Version Manager that has the file embedded in it. For instructions on how to embed a master configuration file, see "Embedding a Master Configuration File into Version Manager" on page 103.

Guidelines for Setting Configuration Options in the Desktop Client

This section provides some guidelines for the types of options to set in the master and project configuration files in the desktop client.

Master Configuration File

Use the master configuration file to set options that dictate how Version Manager operates on a project database and specify (disallow) options that cannot be changed in other configuration files (see the previous explanation of master configuration files). For example:

- The login sources that Version Manager uses to obtain user identification.
- The location of program files that all users must be able to access, such as the access control database.
- Locking options that affect revision access and parallel development.
- Semaphore options that affect how Version Manager protects against archive corruption in the event of simultaneous access.
- A promotion model that all projects should use.

Project Configuration File

Use project configuration files to set project-specific options that have not been disallowed in the master configuration file. Remember that project configuration settings override the settings in the master configuration file. Set options for each project that determine how Version Manager operates for that particular project. For example:

- A promotion model that is custom-designed for a project
- A project-specific journal file
- A project-specific reference directory

How Version Manager Reads Configuration Files in the Desktop Client

Version Manager always reads the master configuration file before reading project configuration files. The reason for this is that the master configuration file defines options that cannot be reset by other configuration files.

The project configuration file settings override the master configuration file settings unless the master disallows the options. For example, if the master configuration file sets the archive suffix to +rev and does not disallow this option and the project configuration file sets the archive suffix to _src, then the project configuration file's setting for the archive suffix is used.

NOTE The one exception to the project configuration file settings overriding the master configuration file settings is the promotion option. See "Defining a Promotion Model" on page 250.

If Version Manager cannot find a master or project configuration file, it uses the default setting for each option. See "Default Settings when No Configuration File Is Used" on page 56.

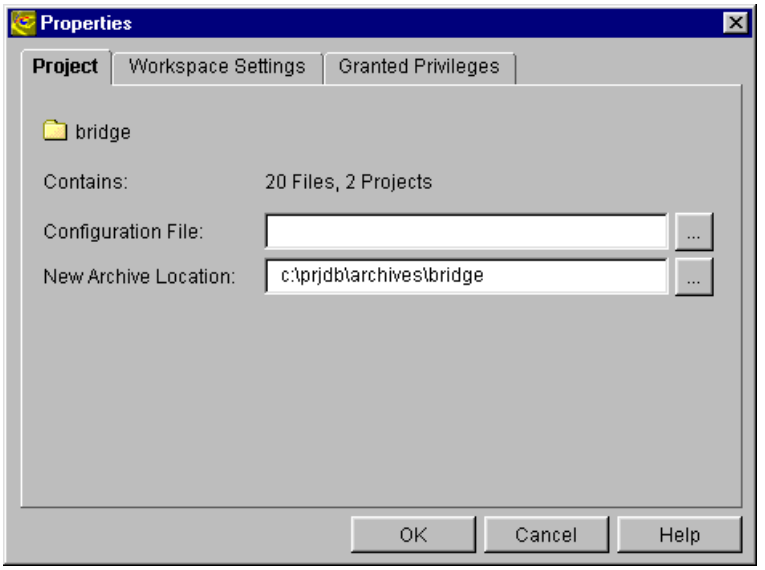
Associating Configuration Files with Project Databases and Projects

By default in the desktop client, Version Manager associates a master configuration file with each project database that is created. You have the option when creating a project database of using an existing configuration file.

Projects, by default, are not associated with a project configuration file. To associate a project configuration file, with a project:

1 Select the project you want to associate with a configuration file.

2 Select File | Properties. The Properties dialog box appears with the Project tab active.



3 In the **Configuration File** field, enter the location and name of the configuration file to be associated with the project. If the configuration file does not already exist, Version Manager creates it automatically, using the name and location you specify here. The default name is project.cfg. The configuration file has no options set. You must now configure the project. See "Setting Configuration Options" on page 64.

4 Click OK.

When working with configured project databases and projects, you may find that you want to:

- Rename configuration files.
- Move configuration files.
- Associate multiple project databases with one master configuration file and multiple projects with one project configuration file.
- Store all of your configuration files in one secure location.

Renaming Configuration Files

To rename a configuration file that is associated with a project database or project:

1 Make a copy of the configuration file.

2 Rename the configuration file.

3 From the Version Manager desktop client, associate the renamed configuration file with the appropriate project database or project (File | Properties).

4 Optionally, you can delete the old configuration file.

Moving Configuration Files

To move a configuration file that is associated with a project database or project:

1 Make a copy of the configuration file.

2 Move the copied file to the new location.

3 From the Version Manager desktop client, associate the copied configuration file with the appropriate project database or project (File | Properties).

4 Optionally, you can delete the old configuration file.

Understanding Configuration Files in the Command-Line Interface

Version Manager stores configuration options in configuration files. The two types of configuration files for the command-line interface are:

- Master configuration file, which contains configuration options for how Version Manager operates on all archives and commands.
- Local configuration file, which contains configuration options for files stored in the directory where the configuration file is located. This file is named vcs.cfg by default.

The Version Manager command-line interface uses a configuration file if it is:

- Embedded into Version Manager. A configuration file that is embedded is the master. See "Embedding a Master Configuration File into Version Manager" on page 103.
- Specified on the command line when you issue a command. Most commands provide the -C command-line option that lets you specify a configuration file. See the *Serena PVCS Version Manager Command-Line Reference Guide*.
- Defined in the VCSCFG environment variable. See the *Serena PVCS Version Manager Command-Line Reference Guide*.
- A local configuration file that is located in your current working directory.

Master Configuration File

A master configuration file lets you standardize how Version Manager operates by identifying options that cannot be reset by other configuration files. By disallowing options in the master configuration file, you force Version Manager to operate according to the settings in the master. For example, if the master configuration file sets the option to automatically create an archive when checking in a new file and then disallows the option, you cannot change the option in other configuration files and, therefore, Version Manager automatically creates archives. In other words, you set the option in the master configuration file and then disallow it so that other configuration files cannot reset the option.

Version Manager installs a default master configuration file named default.cfg. By default, this file is placed in the following location during Version Manager installation:

- In Windows: *drive*:\Program Files\Serena\vm\common\pvcsprop\pvcs\vm
- On UNIX: /usr/serena/vm/common/pvcsprop/pvcs/vm

You can change the settings in this file to suit your needs. To use this file as the master configuration file in the command-line interface, make a copy of this file, change any of the options you want, and embed the copied file into Version Manager using the VCONFIG command. Embedding the configuration file into Version Manager ensures that all users

will be using the same configuration for Version Manager and that users cannot use a different master configuration file.

A master configuration file that is embedded into Version Manager affects all desktop client and command-line users who are using the copy of Version Manager that has the file embedded in it. For instructions on how to embed a master configuration file, see "Embedding a Master Configuration File into Version Manager" on page 103.

Guidelines for Setting Configuration Options in the Command-Line Interface

This section provides some guidelines for the types of options to set in the master and local configuration files in the command-line interface.

Master Configuration File

Use the master configuration file to set global options and specify (disallow) options that cannot be changed in other configuration files (see the previous explanation of master configuration files). For example:

- The login sources that Version Manager uses to obtain user identification
- The location of program files that all users must be able to access, such as the access control database
- Locking options that affect revision access and parallel development
- Semaphore options that affect how Version Manager protects against archive corruption in the event of simultaneous access
- A promotion model that all projects should use

Local Configuration File

Use local configuration files to set user-specific options. Set options in local configuration files when you want to make exceptions to master configuration settings. If you want to set configuration options one way for the majority of the users and another way for a few users, you can create a local configuration file.

For example, if the project is set up to operate on a specific version label and one user needs to work on a different one, you can use a local configuration file to define this exception. Also, if a project is not set up to maintain a reference directory, you can define local configuration files so that individual users can use reference directories to maintain reference copies of the files that they modify on their local computers.

How Version Manager Reads Configuration Files in the Command-Line Interface

Version Manager always reads the master configuration file before reading local configuration files. The master configuration file defines options that cannot be reset by other configuration files. The local configuration file settings override the master configuration file settings unless the master configuration file disallows the options. For example, if the master configuration file specifies that multiple locks on archives are allowed and does not disallow this option and the local configuration file specifies that

multiple locks on archives are **not** allowed, then the local configuration file setting for archive locking is used.

If Version Manager cannot find a master or local configuration file, it uses the default setting for each option. The default settings for options are provided in the section "Default Settings when No Configuration File Is Used" on page 56.

Most commands provide the -C command-line option that lets you specify a configuration file to use.

How the Command-Line Interface Uses Configuration Files Created in the Desktop Client

A configuration file that is created and maintained in the desktop client is compatible with the command-line interface. The command-line interface needs to use the VCSDir directive to find the location of archives; whereas, the desktop client does not. However, you can set the VCSDir directive in the desktop client (Admin | Configure Project | General tab | Miscellaneous | Archive Search Path).

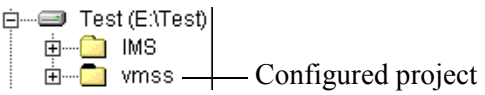
By default, when you create a project database in the desktop client, Version Manager sets the VCSDir to the database's archive locations in the configuration file. If you add workfiles to the project database and use archive locations for these files outside of the project database's archive structure, you must update the VCSDir setting so that command-line operations work.

Setting Configuration Options

This section tells you how to set configuration options in the desktop client and the command-line interface and defines each configuration option. In some cases, the options are described in detail in another chapter of this manual. When this is the case, you are referred to that chapter.

Using the Desktop Client

Projects that have a configuration file already associated with them are displayed in the desktop client with a folder icon that has a black tab.

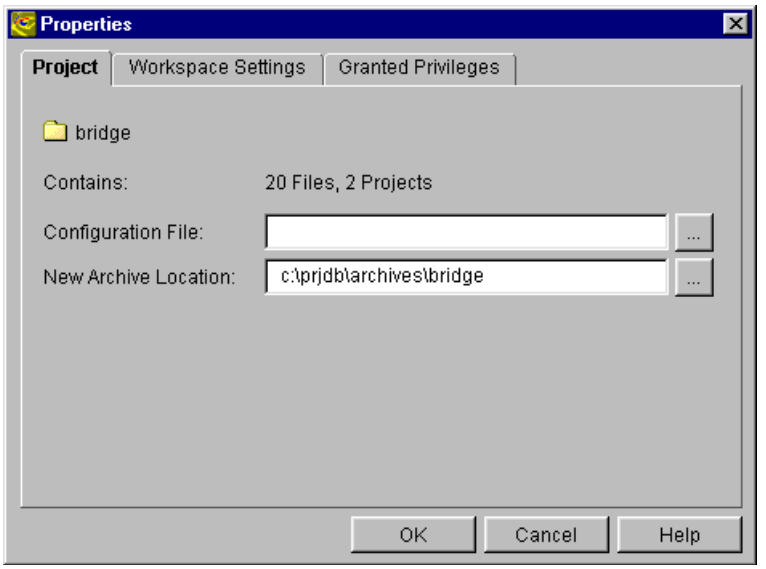


To set configuration options for a project database or project:

1 Select the project database or project you want to configure.

2 If the project database or project has a configuration file associated with it, go to Step 3. Otherwise, associate a configuration file with it by doing the following:

a Select File | Properties. The Properties dialog box appears with the Project or Project Database tab active. The following graphic shows the Properties dialog box for a project.



b In the **Configuration File** field, enter the location and name of the configuration file to be associated with the project database or project. If the configuration file does not already exist, Version Manager creates it automatically, using the name and location you specify here.

c Click OK.

3 Select Admin | Configure Project. The Configure Project dialog box appears with the General tab active.

4 If not already selected, select the **Show Advanced Options** check box to see all of the available options.

5 Select the option you want to define in the Options list on the General tab, or select one of the other tabs. The options are described later in this chapter.

Options	See page
Archive Creation	67
Branching	69
Locking	69
Promotion	71
Access Control Database	71
Journal File	72
Login Sources	73
LDAP Configuration	79
Workfile Attributes	83
Keyword Expansion	84

Options	See page
Reference Directory	86
Archive Search Path	88
Command Line	90
Semaphore	92
Temporary Files	95
File Types	96
Event Triggers	101
Allow/Disallow	101

6 Click OK if you are finished defining options, or click Apply to save the settings and continue to define other options.

Using the Command-Line Interface

If you typically use the Version Manager command-line interface, you may want to edit configuration files using a text editor. If you find it easier, you can use the desktop client to configure Version Manager. A configuration file that is maintained in the desktop client is compatible with the command-line interface.

Before you can manually edit a configuration file, you must understand its format. You insert directives into a configuration file. Directives are the names of the configuration options, for example, ArchiveSuffix is a directive that specifies the extension used for archives. You must specify a value for some directives, for example `ArchiveSuffix = +-arc`. Others do not need a value, for example, `Journal`, which tells Version Manager to make an entry in the journal file after it modifies an archive.

The rules governing configuration files are:

- They must be standard text files.
- Comment lines must begin with a pound sign (#).
- To continue a line in a configuration file, place a backslash (\) at the end of the line. The backslash must be the last character on the line, spaces cannot follow it.
- The Disallow directive can only be used in master configuration files.

Using Conditional Constructs in Configuration Files

You can use conditional constructs in configuration files to direct Version Manager to read different parts of the configuration file under specific circumstances.

For example, a System Administrator may want to save time by including the configuration information for all projects in one configuration file—instead of providing one configuration file per project, which would require excessive maintenance.

For more information, refer to "Using Conditional Constructs in Configuration Files" in the *Serena PVCS Version Manager Command-Line Reference Guide*.

Archive Creation Options

The archive creation options affect only newly created archives. You can set these options on existing archives by using the
Admin | Archive Attributes menu item in the desktop client and the VCS command in the command-line interface.

The archive creation options let you:

- Prevent multiple locks on a single archive. If you allow multiple locks, different users can place locks on different revisions in an archive at the same time. The directive for this option is ExclusiveLock.

NOTE The MultiLock and ExclusiveLock directives are mutually exclusive. The MultiLock directive has no affect on an archive that was created while the ExclusiveLock directive was in effect. See "Locking Options" on page 69 for information about MultiLock.

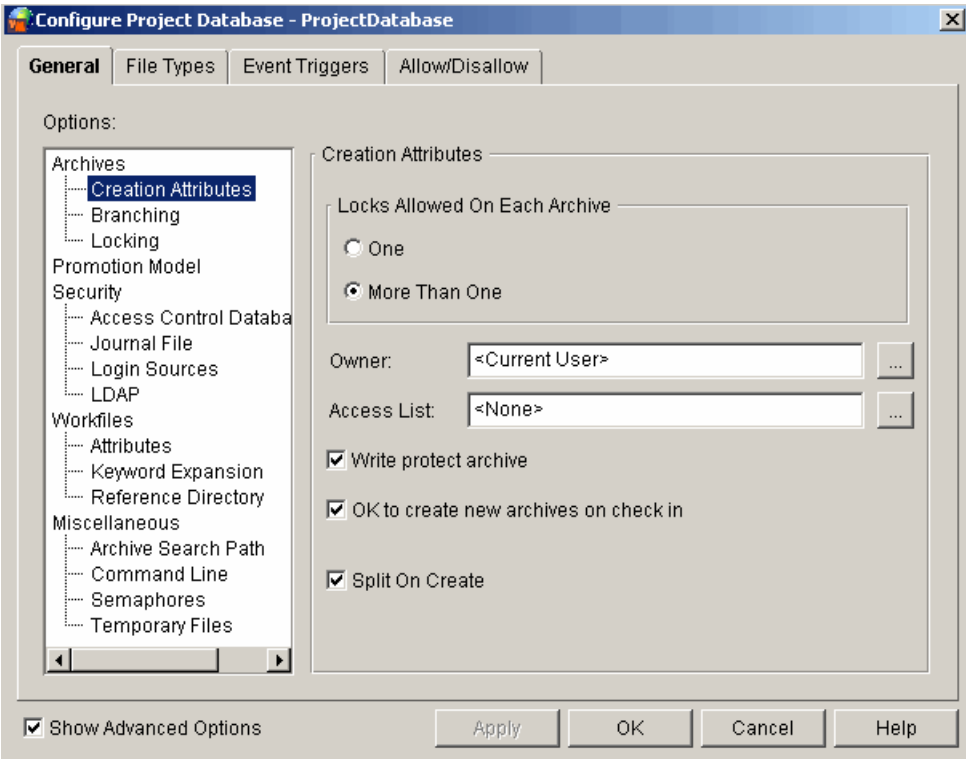
- Assign ownership of archives. The directive for this option is Owner.
- Define an access list for archives, which identifies users and groups of users who are allowed to access archives. The directive for this option is AccessList.
- Write-protect archives. Doing this protects archives from inadvertent deletion or modification. Version Manager commands automatically remove write-protection before modifying archives and replace it afterwards. The directive for this option is WriteProtect.
- Automatically create an archive when you check in a workfile if Version Manager cannot find one for the file. The directive for this option is AutoCreate.

Desktop client

To set these options in the desktop client:

1. Select Admin | Configure Project. The Configure Project dialog box appears.
2. If not already selected, select the **Show Advanced Options** check box.

3 Select **Creation Attributes** beneath Archives. The Creation Attributes pane appears on the right. The values that display for each option are the settings that are currently defined in the configuration file associated with the project database or project.



4 To set the options, do any of the following:

- Select either **One** or **More Than One** for the locks allowed on each archive.
- Enter the user ID of the owner of new archives in the **Owner** field. The value <Current User> is the user who is running Version Manager at the time of archive creation.
- Enter the access list groups and user IDs that can access newly created archives in the **Access List** field. The combined value of this field cannot exceed 254 characters in length. The groups and user IDs that you specify must be defined in the access control database. Values must be separated by commas. See "Using Security" on page 203 for a complete explanation of access lists.
- Select the **Write protect archive** check box to write-protect the archives. Setting this option protects archives from inadvertent deletion or modification.
- Select the **OK to create new archives on check in** check box to automatically create an archive when you check in a workfile if Version Manager cannot find one for the file. Otherwise, a message is issued that no archive is created. This option applies only to the command-line interface.
- Select the **Split On Create** check box to automatically split new archives into separate metadata and revision components if they are mapped to a revision path by the Version Manager File Server. This directive is for use with the Revision Library feature of the Version Manager File Server. By default, it is enabled for project databases created on a Version Manager File Server. For more information, see "Creating Revision Libraries" on page 157.

5 Click **OK** if you are finished defining options, or click Apply to save these settings and continue to define other options.

Command-line interface

The directives that define the archive creation options are:

- `ExclusiveLock | NoExclusiveLock`. The default is NoExclusiveLock, which means multiple locks are allowed on each archive.
- `Owner`. The default is the user who creates the archive.
- `AccessList`. The default is no access list.
- `WriteProtect | NoWriteProtect`. The default is to write-protect the archive.
- `AutoCreate | NoAutoCreate`. The default is to automatically create the archives.
- `RFSSplitOnCreate | NoRFSSplitOnCreate`. The default is not to split new archives.

Refer to the *Serena PVCS Version Manager Command-Line Reference Guide* for complete information about these directives.

Branching Options

The branching options let you set up projects for automatic branching. A branch is a separate line of development that has one or more revisions that diverge from a revision on the trunk or from another development branch. Branching lets you develop different versions of a file in parallel with other developers who are working on the trunk. The three directives that let you set up automatic branching are: DefaultVersion, BaseVersion, and BranchVersion.

By default, the Branching options have no values set in the desktop client or the command-line interface, meaning automatic branching is not defined.

For complete information about branching and how to set it up, see "Branching and Merging Files" on page 263.

Locking Options

The locking options for archives let you:

- Permit a user to lock more than one revision of an archive and/or permit more than one lock per revision. The directive for this option is MultiLock. For a complete explanation of multiple locking, see "Using Multiple Locks for Branching" on page 270.

NOTE The MultiLock and ExclusiveLock directives are mutually exclusive. The MultiLock directive has no affect on an archive that was created while the ExclusiveLock directive was in effect. See "Archive Creation Options" on page 67 for information about ExclusiveLock.

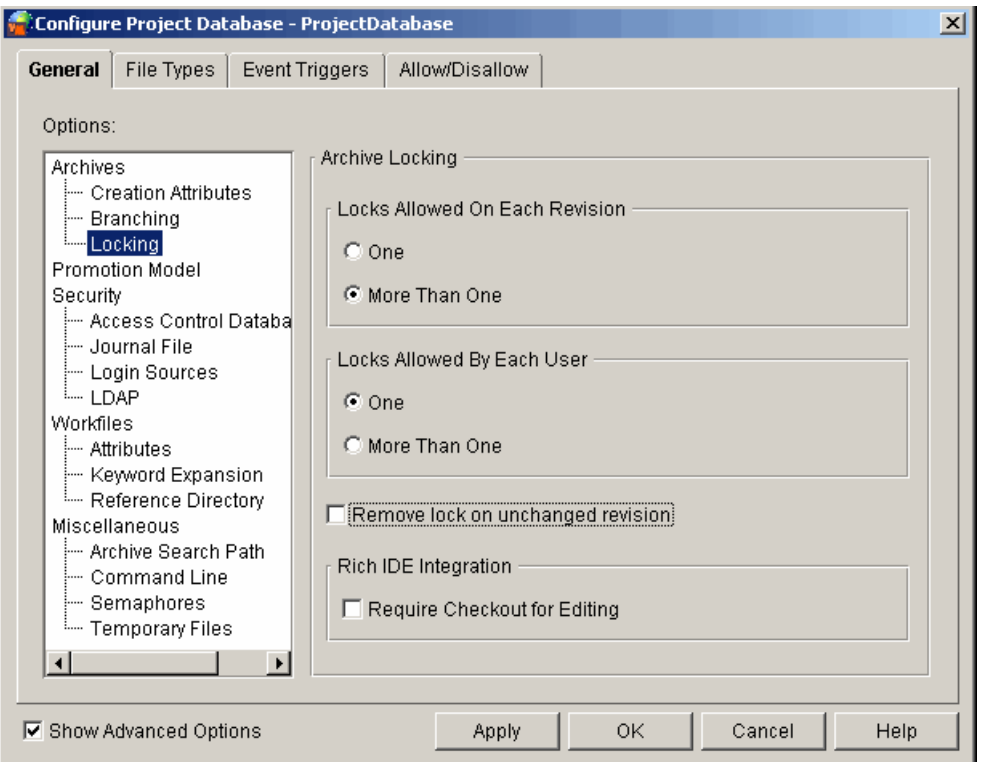
- Configure Version Manager to remove the lock when checking in an unchanged revision and to not increment the tip revision number. The directive for this option is ForceUnlock.
- Enforce pessimistic locking on clients capable of optimistic locking. To support team-oriented agile development, some Version Manager clients allow users to modify files and check in and merge changes without having first checked out the files (optimistic locking). In this model, multiple users may be working on the same file at the same

time. Optimistic locking depends upon file merges and synchronization of each user's workspace with the repository rather than upon pessimistic locks.

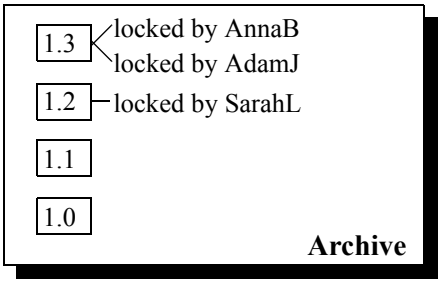
NOTE The rich integrations to both Visual Studio and Eclipse can use optimistic locking. However, at present (Version Manager 8.2), pessimistic locking can be enforced only on Visual Studio 2005.

Desktop client **To set these options in the desktop client:**

1. Select Admin | Configure Project. The Configure Project dialog box appears.
2. If not already selected, select the **Show Advanced Options** check box.
3. Select **Locking** beneath Archives. The Archive Locking pane appears on the right. The values that display for each option are the settings that are currently defined in the configuration file associated with the project database or project.

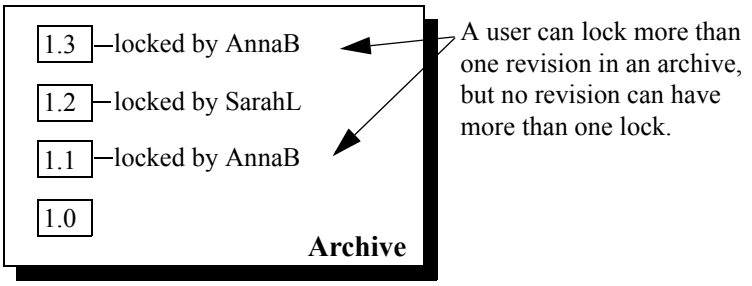


4. Set any of the following options:
   - Select either **One** or **More Than One** for the locks allowed on each revision. The following figure illustrates multiple locks on a single revision.



No user can have more than one lock in an archive, but multiple locks on a revision are allowed.

- Select either **One** or **More Than One** for the locks allowed by each user. The following figure illustrates multiple locks per user.



- Select the **Remove lock on unchanged revision** check box to tell Version Manager to remove the lock on an unchanged revision during check in and to not increment the tip revision number. If you do not select this option, Version Manager cancels the check in and the lock remains in force.
- Select **Require Checkout for Editing** to enforce pessimistic locking on clients that are otherwise capable of optimistic locking. Under pessimistic locking, a user must check out a file before checking in changes to it.

NOTE The rich integrations to both Visual Studio and Eclipse can use optimistic locking. However, at present (Version Manager 8.2), pessimistic locking can be enforced only on Visual Studio 2005.

5 Click **OK** if you are finished defining options, or click **Apply** to save these settings and continue to define other options.

Command-line interface

The directives that define the archive locking options are:

- `MultiLock | NoMultiLock`. The default is NoMultiLock, which means multiple users can place locks on different revisions, but not on the same revision.
- `ForceUnlock | NoForceUnlock`. The default is NoForceUnlock, which means Version Manager does not check in an unchanged revision and the lock remains in force.

Refer to the *Serena PVCS Version Manager Command-Line Reference Guide* for complete information about these directives.

Promotion Options

The Promotion option lets you define a promotion model by defining each promotion group in relation to the next higher group. The directive for this option is Promote.

By default, there is no promotion model defined in the desktop client or the command-line interface.

For complete information about promotion models and how to define them, see "Using Promotion Models" on page 247.

Access Control Database Options

The access control database options let you specify an access control database for Version Manager to use and let you enable the specified access control database. The directives for these options are AccessDB and AccessControl | NoAccessControl, respectively.

By default, there is no access control database enabled in the desktop client or the command-line interface. In the desktop client, there is an access control database created when you create a project database, but it is disabled.

For complete information about access control databases and how to specify one, see "Using Security" on page 203.

Journal File Options

A journal file contains a record of the actions that users perform on archives. Each time a user performs an action that updates an archive, Version Manager records information about the action in the journal file. The journal file can be used to generate a journal report. See "Generating Journal Reports" on page 297.

You have three options for journal files:

- Do not keep a journal file. The directive for this option is `NoJournal`.
- Keep a journal file named journal.vcs for each archive and store it in the archive directory. The directive for this option is `Journal`.
- Keep a single journal file for all of the archives in a project, and specify its name and where to store it. The directive for this option is `Journal = Path\File_Name`.

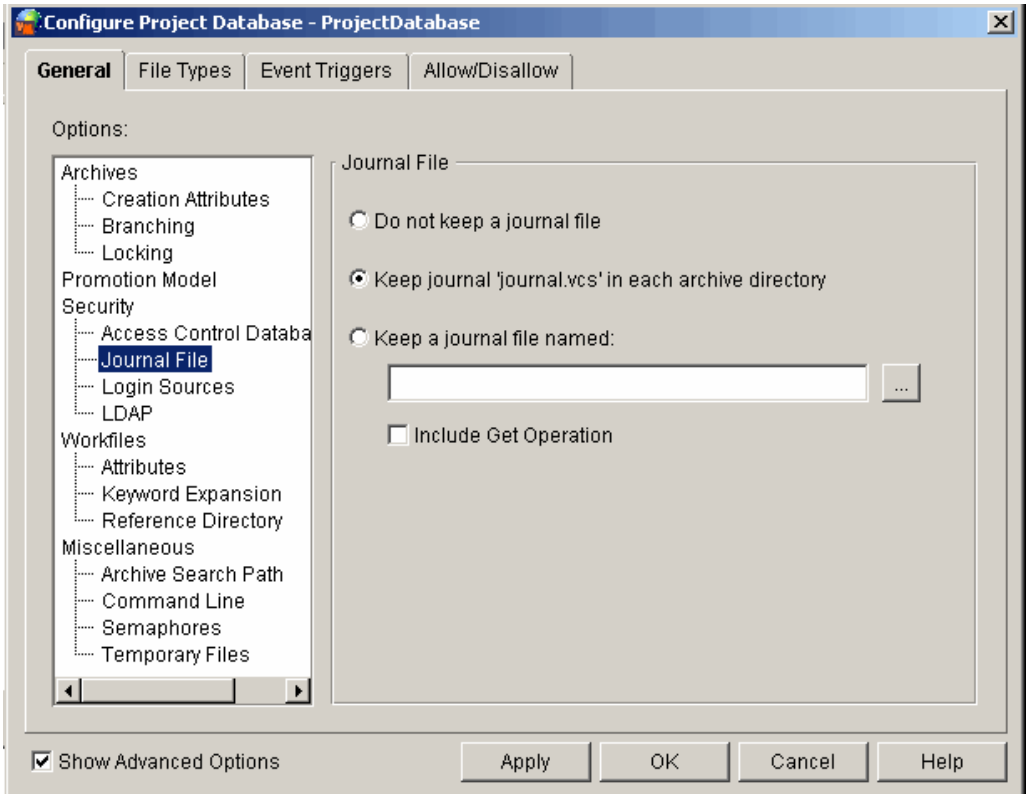
NOTE If you do not specify a path, a journal file of the specified name will be created in each archive directory.

Desktop client

To set these options in the desktop client:

1. Select Admin | Configure Project. The Configure Project dialog box appears.
2. If not already selected, select the **Show Advanced Options** check box.

3 Select **Journal File** beneath Security. The Journal File pane appears on the right. The values that display for each option are the settings that are currently defined in the configuration file associated with the project database or project.



4 Select one of the three **Journal File** options.

If you select **Keep a journal file named**, you must specify a name for the journal file. If you do not specify a path with the file name, Version Manager places the journal file in the archives directory. This option tells Version Manager to keep a single journal file for all of the archives in a project.

5 Select the **Include Get Operation** checkbox to include non-locking Get operations. Locking Get operations (checkouts) are included by default.

6 Click OK if you are finished defining options, or click Apply to save these settings and continue to define other options.

Command-line interface

The `Journal` directive allows you to set the journal file option. The default is not to keep a journal file (`NoJournal`).

Refer to the *Serena PVCS Version Manager Command-Line Reference Guide* for complete information about this directive.

Login Sources

Before users begin using Version Manager, you should set up Version Manager to obtain user identification from a *login source*. A login source is an operating system, network, or utility that Version Manager uses to obtain user identification. Version Manager uses the user identification it obtains from the login source as the author for archive operations.

Before taking an action, Version Manager attempts to obtain user identification from the login sources that you specified in the Configure Project dialog box, starting with the first. If Version Manager cannot obtain user identification, it displays an error message and terminates. If a user ID can be obtained and no access control databases are enabled, any user can access the archives.

Valid login sources are:

Host ID	Host operating system. Use this source with systems that provide a user identification mechanism, such as UNIX or Windows, or for environments in which more than one network is in use. The directive for this option is `LogIn=HOST`.
LDAP ID	Lightweight Directory Access Protocol (LDAP). Use this source to authenticate user IDs and passwords against an LDAP server. Once authenticated against LDAP, user IDs are passed to the access control database, if one is in effect. The passwords, if any, in the access control database are ignored. The directive for this option is LogIn=LDAP. LDAP does not work with the command-line interface (CLI). If LDAP is the first login source specified, the CLI will attempt to use the next login source. If no other login sources are specified, the CLI command will fail. **NOTE** Microsoft Active Directory uses LDAP.
Login Dialog	The Version Manager desktop and Web client login utility. This source requires users to enter their user ID and a password, if one is defined, before they can use Version Manager. To use password protection, an access control database must be defined. This login source applies only to the operation of the desktop and Web clients. The directive for this option is `LogIn=VLOGIN`.
Netware ID	Novell NetWare (Windows only). Use this source to obtain user IDs from a Novell NetWare server, rather than Windows. The directive for this option is `LogIn=NETWARE`.
VCS ID	The user's PVCS ID, which Version Manager derives from the value of the VCSID environment variable. The directive for this option is `LogIn=VCSID`. Be aware that using VCSID as a source for user identification is not secure. Users can circumvent security by logging in as another user or resetting the value of the VCSID environment variable.
WNet ID	Microsoft Windows networks. Version Manager obtains the user ID from the Microsoft WNET API. The directive for this option is `LogIn=WNET`.

When you execute Version Manager, it searches for a user ID according to the order in which you specified the login sources. Therefore, you must consider the security limitations of your operating systems and login sources when specifying the login source order to ensure that your most vulnerable systems are protected.

NOTE A Login dialog box appears if VLOGIN or LDAP is set as your login source. Canceling, or failing, the login from this dialog box cancels the login operation. Additional login sources are not searched and you cannot log into the project database or project.

For UNIX users: The UNIX GUI and PCLI support only the Host ID, LDAP ID, Login Dialog (PCLI uses the PCLI_ID environment variable or the -id switch rather than a dialog box),

and VCS ID login sources. The UNIX CLI supports only Host ID and VCS ID. The default value is Host ID.

For desktop client users: When Version Manager obtains a user ID, the program can check the access control database to see whether the user ID exists there. If the user ID does not exist, you can configure Version Manager to automatically create the user ID in the access control database and assign privileges to the user.

This is a useful feature for organizations that have not yet decided to use access control databases. If at some point the organization decides to enable access control, the users are already defined—thus, reducing the time it would take to set up security. Also, this is a simple way to allow guest accounts with restricted privileges.

User Ids can be either case-sensitive or case-insensitive; however, you cannot define duplicate user IDs that differ only in case. The directive that controls case sensitivity is CASE.

For Web client users: The Version Manager Web client works with the following login sources: Host ID, LDAP ID, and Login Dialog. If none of those login sources are enabled, the Web client will default to Login Dialog.

Single Sign-On

The administrator can configure Version Manger projects and project databases to recognize a user who has already logged in to Version Manager. When single sign-on is enabled, a user can login once and access all single sign-on enabled projects without logging in again.

NOTE Your Version Manager user ID and password combination must match across the project databases and projects on which you wish to use single sign-on. Single sign-on is supported only in the Desktop Client.

To enable single sign-on, see the next section.

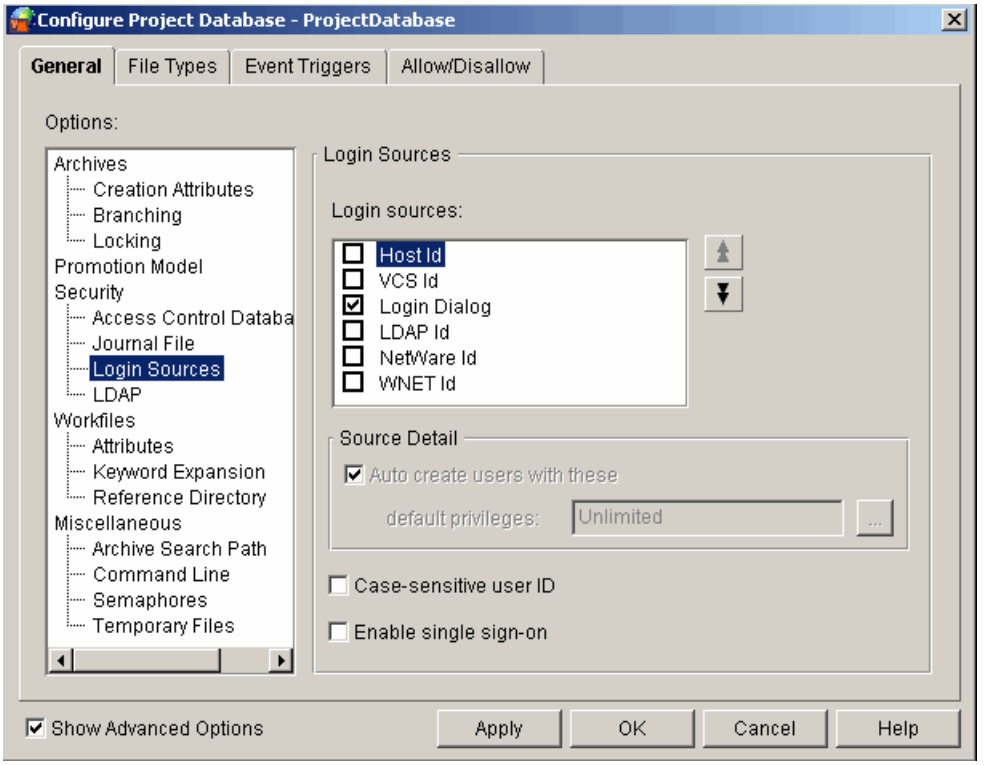
Setting Login Sources

Desktop client

To set these options in the desktop client:

1. Select Admin | Configure Project. The Configure Project dialog box appears.
2. If not already selected, select the **Show Advanced Options** check box.

3 Select **Login Sources** beneath Security. The Login Sources pane appears on the right.



The fields display the current settings as defined in the configuration file associated with the project database or project.

4 Select the appropriate login sources for your configuration.

By default, when you create a project database, Host Id is defined as the login source in the master configuration file.

5 Use the up arrow and down arrow buttons to arrange the login sources in the desired order. Remember that the order you define here is the order in which Version Manager searches for user IDs. The search terminates at the first log in source that works.

NOTE A Login dialog box appears if VLOGIN or LDAP is set as your login source. Canceling, or failing, the login from this dialog box cancels the login operation. Additional login sources are not searched and you cannot log into the project database or project.

6 To configure any of the login sources that you selected to automatically create user IDs in the access control database:

a Select the login source.

b Select the **Auto create users with these default privileges** check box.

c Specify the default privileges that will be assigned to the user. See "About Privileges" on page 208 for a complete discussion of privileges.

By default, the Host Id login source automatically creates users in the access control database and assigns them the Unlimited privilege set.

7 To make user IDs case-sensitive, select the **Case Sensitive User ID** check box. By default, user IDs are **not** case-sensitive in the desktop client.

8 To enable single sign-on access to the selected project or project database, select the **Enable single sign-on** check box. For more information, see "Single Sign-On" on page 75.

NOTE Your Version Manager user ID and password combination must match across the project databases and projects on which you wish to use single sign-on. Single sign-on is supported only in the Desktop Client.

9 Click **OK** if you are finished defining options, or click **Apply** to save these settings and continue to define other options.

Command-line interface

The directives that define the login source options are:

- **LogIn**. Valid values are: HOST, LDAP (not supported in the CLI), NETWARE, VCSID, and WNET. HOST is the default for all platforms.
- **Case**. By default, user IDs are case-sensitive.

Refer to the *Serena PVCS Version Manager Command-Line Reference Guide* for complete information about these directives.

Embedding Login Sources into Version Manager

Embedding login sources into Version Manager ensures that Version Manager uses the same login sources to obtain user IDs for all users. To prevent users from changing login sources, disallow the LogIn directive in all master configuration files and do not give users the Configure Project privilege (desktop client only).

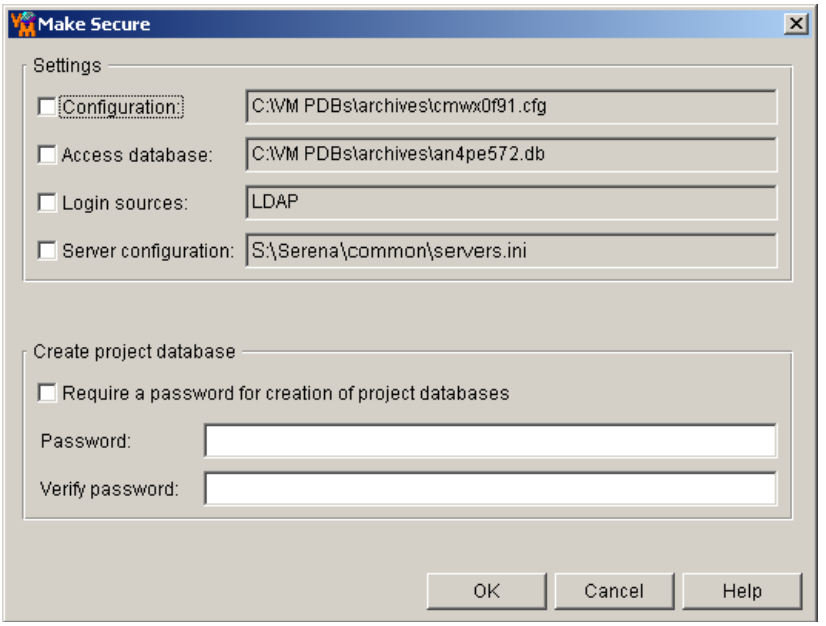
Login sources that are embedded into Version Manager affect all desktop client and command-line users who are using the copy of Version Manager that has the login sources embedded.

Desktop client

To embed login sources:

1 Select the project database associated with the master configuration file that has the appropriate login sources set.

2 Select Admin | Make Secure. The Make Settings Secure dialog box appears.



3 Select the **Login Sources** check box. The field next to this check box displays the login sources that will be embedded. You cannot edit this field.

4 Click OK.

Command-line interface

To embed the name, use the VCONFIG command:

`vconfig -i`*`source`*`[,`*`source...`*`] `*`vm_filename`*

where:

`source` is the name of the login source that you are embedding. Valid values are: `HOST`, `NETWARE`, `VCSID`, `WNET`, and `VLOGIN` (for desktop client operation only).

`vm_filename` is either:

- `VMWFVC.DLL` for Windows
- `vmufvc.a` for UNIX

NOTE vconfig is located in the admin subdirectory of the bin directory.

After you embed the login sources, you should move the file (VMWFVC.DLL or vmufvc.a) to the same location as the Version Manager executable files, if the file is not there already. Also, make sure users do not have write permission to this location. You do not want users to be able to embed login sources into Version Manager.

For more information about the `VCONFIG` command, see the *Serena PVCS Version Manager Command-Line Reference Guide*.

LDAP Configuration Options

To use Version Manager with a Lightweight Directory Access Protocol (LDAP) server you must:

- Have a functional LDAP server.
- Select **LDAP ID** as one of your login sources. For more information, see "Login Sources" on page 73.
- Specify an LDAP server. For more information, see the next section.
- Specify an LDAP configuration. For more information, see "Specifying an LDAP Server Configuration" on page 80.

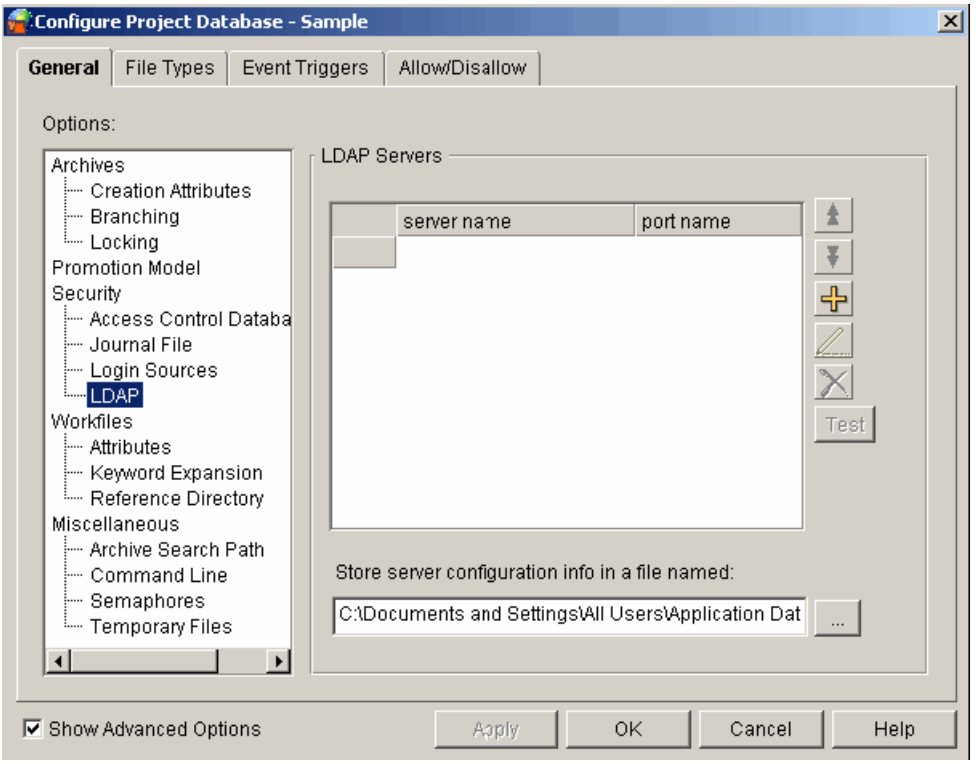
Specifying and Configuring LDAP Servers

To specify LDAP servers:

1 Select Admin | Configure Project. The Configure Project dialog box appears.

2 If it is not already selected, select the **Show Advanced Options** check box.

3 Select **LDAP** beneath Security. The LDAP Servers pane appears on the right.



4 To add a new server, click the Add button. To edit the configuration settings of a server, click the Edit (Writing pen icon) button.

NOTE You can add multiple LDAP servers using the Add button.

- To reposition the selected server in the **LDAP Servers** list, click the Up or Down arrow button. Version Manager searches for login authorization starting from the top of this list.

- To delete the selected server from the list, click the Delete button.
- To test the connection to the selected server, click the **Test** button.

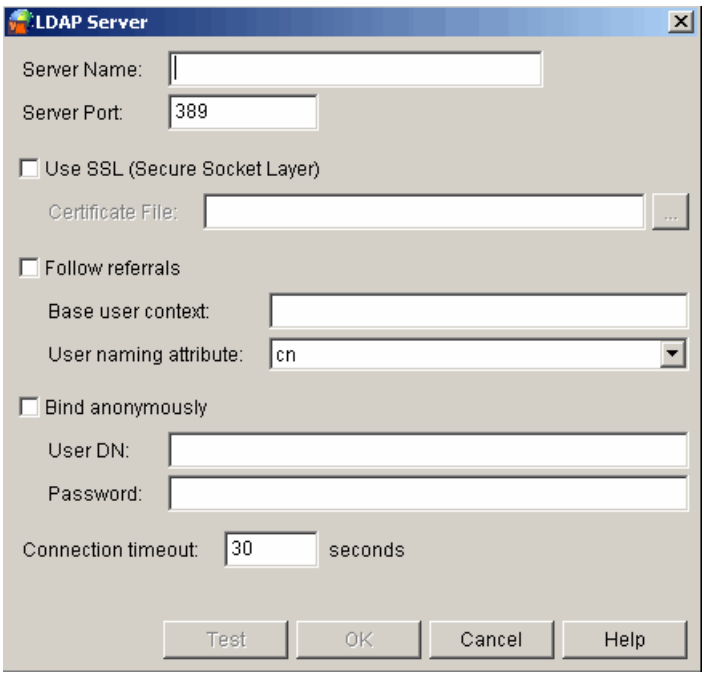
5 Enter a path and file name (`*.ini`) in which to store server configuration information in the **Store server configuration info in a file named** field.

Special Considerations

- This field is not available if the ServerConfigInfoPath directive is embedded.
- LDAP configuration information can be shared between Version Manager and other LDAP enabled Serena applications via this `.ini` file. To share the LDAP configuration file, you must place the file in an accessible location.
- By default, the LDAP configuration file is created in the root of the project database.
- If you used the File Server Administration Utility to assign LDAP protection to a Client Name, the Version Manager File Server will look for configuration information in the following file:

  VM_Install`\vm\common\bin\`*OS*`\pvcsldap.ini`

6 On selection of the Add or Edit button, the LDAP Server Configuration dialog window appears.



Specifying an LDAP Server Configuration

To specify an LDAP configuration:

1 To add the **LDAP Server configuration** details, complete the following fields:

a **Server Name:** Enter the host name or IP address of the LDAP server. For example, `myserver.mydomain.com`.

b **Port:** Enter the port number of the LDAP server. Typically LDAP servers are configured to use port 389; 636 for SSL.

2 To enable Secure Socket Layer, select the **Use SSL (Secure Socket Layer)** check box.

NOTE If there is no certificate database in the *VM_Install_Dir/vm/common/bin/OS* directory or if the database is missing, you will receive errors, such as 'Failed to connect to LDAP Server.". To create and populate a certificate database using **Netscape 4.7x** or **Certutil**, refer to the *Serena PVCS Version Manager Readme File: Usage Cautions*.

3 To allow Version Manager to follow referrals from one LDAP server to another, select the **Follow referrals** check box. Use this feature to support properly configured non-redundant distributed servers.

NOTE See "Using the LDAP Referral Option" on page 82.

4 Specify the base user context (distinguished name) in the **Base User Context** field. This is the base from which to search for users.

5 Select the user naming attribute from the **User Naming Attribute** list. This is the attribute in which the LDAP server holds the user ID value.

6 Select a method of querying the LDAP server to retrieve the list of users:

- To query anonymously, select the **Bind Anonymously** check box. This requires that the LDAP server is configured to allow anonymous users to retrieve a list of users and attributes.

NOTE You cannot bind anonymously to a Microsoft Active Directory Server.

- To require a user ID and password for queries:

  a Specify a full user DN in the **User DN** field.

  b Specify a user password in the **Password** field.

7 Enter a value in the **Connection timeout** field to set the idle time in seconds before the connection to the LDAP server times out.

8 Click **OK**.

9 Click **OK** or **Apply** from the **LDAP Server** main window.

LDAP Configuration Examples

Following are examples of typical LDAP configurations.

Example 1: Microsoft Active Directory Server

Server Name: myserver.mydomain.com

Server Port: 389 (unless using SSL, then 636)

Use SSL: Not selected (or selected)

Base User Context: ou=people, dc=mydomain, dc=com

User Naming Attribute: sAMAccountName

Bind Anonymously: Not selected

User DN: cn=User One, cn=Users, dc=mydomain, dc=com

Password: OpenSesame

Example 2: Netscape Server

Server Name: myserver.mydomain.com

Server Port: 389 (unless using SSL, then 636)

Use SSL: Not selected (or selected)

Base User Context: ou=people, dc=mydomain, dc=com

User Naming Attribute: uid

Bind Anonymously: Selected

User DN: n/a

Password: n/a

Using the LDAP Referral Option

LDAP referrals is a new option available in LDAP version 3 clients. Use this option to find referral information from the same or other directory servers.

Prerequisites: The directory servers must be configured with referral setup and must have one common user for binding. Use this common user to connect to the servers.

NOTE The referrals may not work in a cross-configured environment (for example, Microsoft Active Directory server with Sun Directory One server).

Example 3: LDAP Referrals Configuration

Server Name: myserver.mydomain.com

Server Port: 389 (If you use SSL, then use port 636)

UseSSL=No ((If you use port 636, then enter Yes)

Base User Context: ou=people, dc=referrraldomain, dc=com

BindAnonymously=No

FollowReferrals=Yes

DereferenceAliases=Never

LDAPConnectionTimeoutSeconds=30

UserNameAttribute=sAMAccountName

UserDN=test1@dev.com

Password: OpenSesame

Workfile Attributes Options

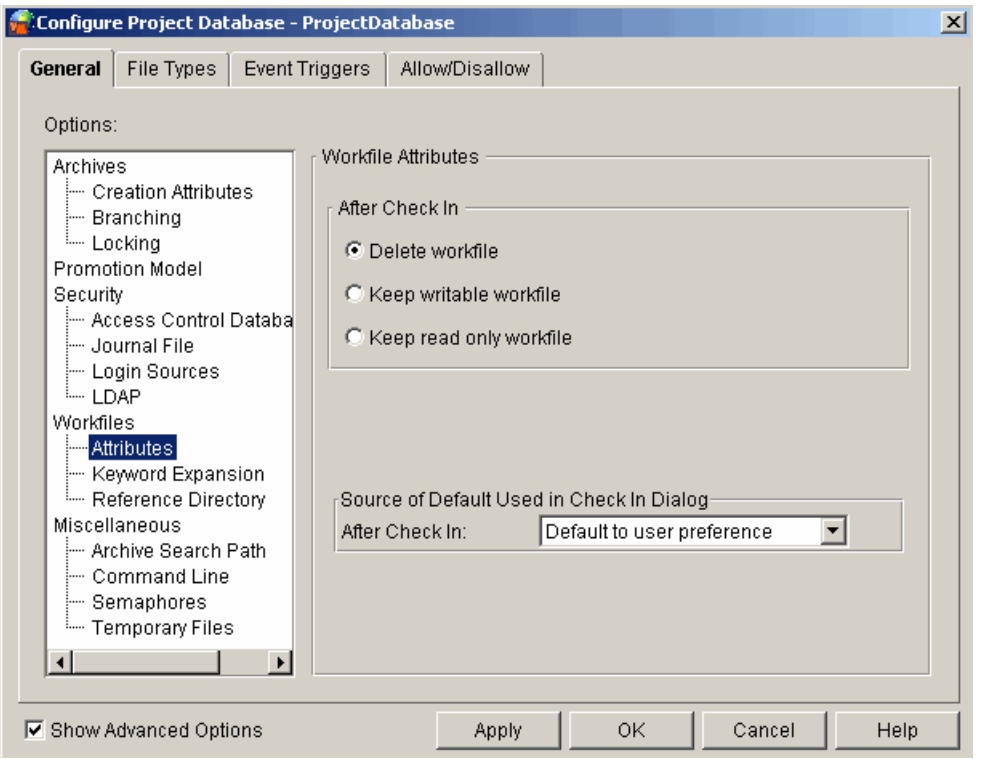
The workfile attributes options let you specify:

- Whether or not a workfile is deleted from the workfile location after check in. Version Manager updates keywords in workfiles that are not deleted after check in. This option can be overridden when you check in workfiles by a setting in the Options dialog box (View | Options). The directive for this option is [No]DeleteWork.
- Whether or not workfiles left in the workfile location are write-protected (read-only). The directive for this option is NoDeleteWork=[No]WriteProtect.

Desktop client

To set these options in the desktop client:

1. Select Admin | Configure Project. The Configure Project dialog box appears.
2. Select **Attributes** beneath Workfiles. The Workfile Attributes pane appears on the right. The values that display for each option are the settings that are currently defined in the configuration file associated with the project database or project.



3. Select one of the three After Check In options:
   - **Delete workfile** to delete the workfile after check in.
   - **Keep writable workfile** to not delete the workfile after check in and to keep the workfile in write mode.

- **Keep read only workfile** to not delete the workfile after check in and to make the workfile read only.

4 Click OK if you are finished defining options, or click Apply to save these settings and continue to define other options.

Command-line interface

The `[No]DeleteWork` directive allows you to set the workfile attributes options. The default is to delete the workfile.

Refer to the *Serena PVCS Version Manager Command-Line Reference Guide* for complete information about this directive.

Keyword Expansion Options

Keywords are special character strings that you can place in workfiles to define information about the archive when expanded. For example, the keyword $Archive$ defines the full path name of the archive. You can set a keyword option to tell Version Manager to expand keywords when the workfile is checked in.

The following table defines each Version Manager keyword. Keywords are case-sensitive.

Keyword	Definition
$Archive$	The full path name of the archive.
$Author$	The user ID of the revision author.
$Date$	The date the workfile was checked in.
$Header$	The archive name, revision number, revision date, and author ID.
Log	Cumulative check-in messages.
$Modtime$	The time of the last modification.
$Revision$	The revision number.
$Workfile$	The file name stored in the Version Manager archive.

CAUTION! Keyword expansion will cause corruption in binary files, so Version Manager disables keyword expansion by default, except for text files with certain file extensions. Do not enable keyword expansion for binary files.

Expanded keywords take the form $*Keyword:text*$, where text is the current value of the keyword. For example, the keyword $Revision$ could have the value of $Revision: 1.0$ when expanded. The Log keyword differs from the other keywords in that its expansion text is not enclosed by dollar signs. Instead, Version Manager inserts the change description after the line that contains the Log keyword. The change descriptions accumulate in the workfile in reverse chronological order.

Fixed Length Keyword Expansion

Certain types of files must maintain a fixed line length to be read correctly. These types of files include column-sensitive languages, non-text files, word processor files, database files, and spreadsheet files. To maintain line length, you can regulate the exact length of

the expanded keyword by placing character "fillers" between the keyword and its terminating dollar sign ($). The syntax is:

$*Keyword*: :$*123456*: :$

Where:

- ***Keyword*** is the keyword.
- ***123456*** is the length delimiter. The length is determined by the number of characters. Any characters (other than **::$**) can be used as the length delimiter.
- **::$** marks the beginning and the end of the length delimiter character string.

If the keyword expands to fewer than the specified number of characters, Version Manager fills the remaining reserved space with spaces. If it expands to more than the specified number of characters, Version Manager includes as many characters as will fit in the reserved space.

CAUTION! The syntax for fixed length keyword expansion has been updated. The old syntax still works (not shown here), but we strongly recommend that you use the new syntax. The old syntax can cause file corruption if the content of the keyword text happens to include a "$" character.

Directives

The options that you can set for keyword expansion are:

- Which path separator is used for expanded keywords—forward slash (/) or backward slash (\). This separator specifies the character Version Manager uses to separate the parts of path specifications that appear in expanded keywords such as $Archive$. The directive for this option is PathSeparator.
- Whether or not the timestamp of the workfile is updated when Version Manager expands keywords after check in. Expanding keywords does not affect the check in and last modified date and time. The directive for this option is ExpandKeywords Touch.

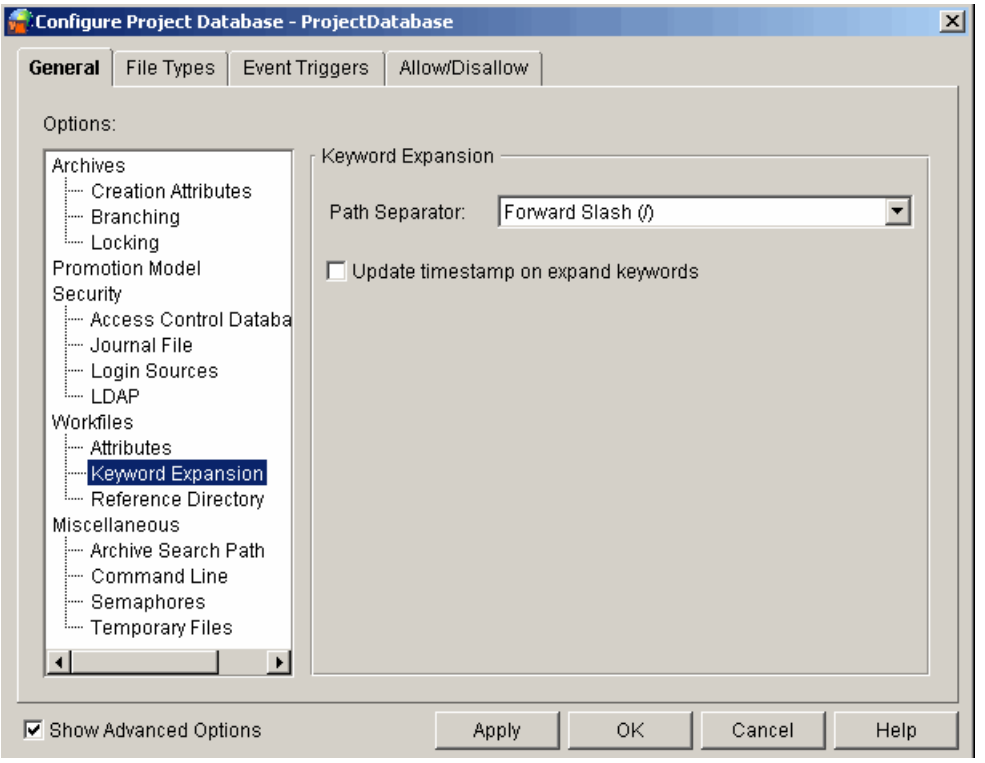
See also "Options Set on a File Type Basis" on page 96 for more keyword expansion options.

Desktop client

To set these options in the desktop client:

1. Select Admin | Configure Project. The Configure Project dialog box appears.
2. If not already selected, select the **Show Advanced Options** check box.

3 Select Keyword Expansion beneath Workfiles. The Keyword Expansion pane appears on the right. The values that display for each option are the settings that are currently defined in the configuration file associated with the project database or project.



4 Select the **Path Separator**—either Forward Slash (/) or Backward Slash (\).

5 Select the **Update timestamp on expand keywords** check box to update the timestamps of workfiles when Version Manager expands keywords after check in.

6 Click OK if you are finished defining options, or click Apply to save these settings and continue to define other options.

Command-line interface

The directives that define the keyword expansion options are:

- `PathSeparator`. The default is a forward slash (/).
- `ExpandKeywords Touch`. The default is to update the timestamp.

Refer to the *Serena PVCS Version Manager Command-Line Reference Guide* for complete information about this directive.

Reference Directory Options

A reference directory is the location where Version Manager automatically stores a copy of each workfile each time it is checked into an archive. A reference directory provides a central repository of workfiles for browsing, printing, or copying. The reference copies enable developers to reuse or examine code without having to check out revisions.

Unless you are using a workbench or integrated development environment that cannot perform any action on write-protected files, including reading them, we recommend that you always maintain read-only workfiles either in reference directories (as described in this section) or in the workfile location (as described in "Workfile Attributes Options" on

page 83). Read-only workfiles remind users not to modify workfiles without locking the archives first.

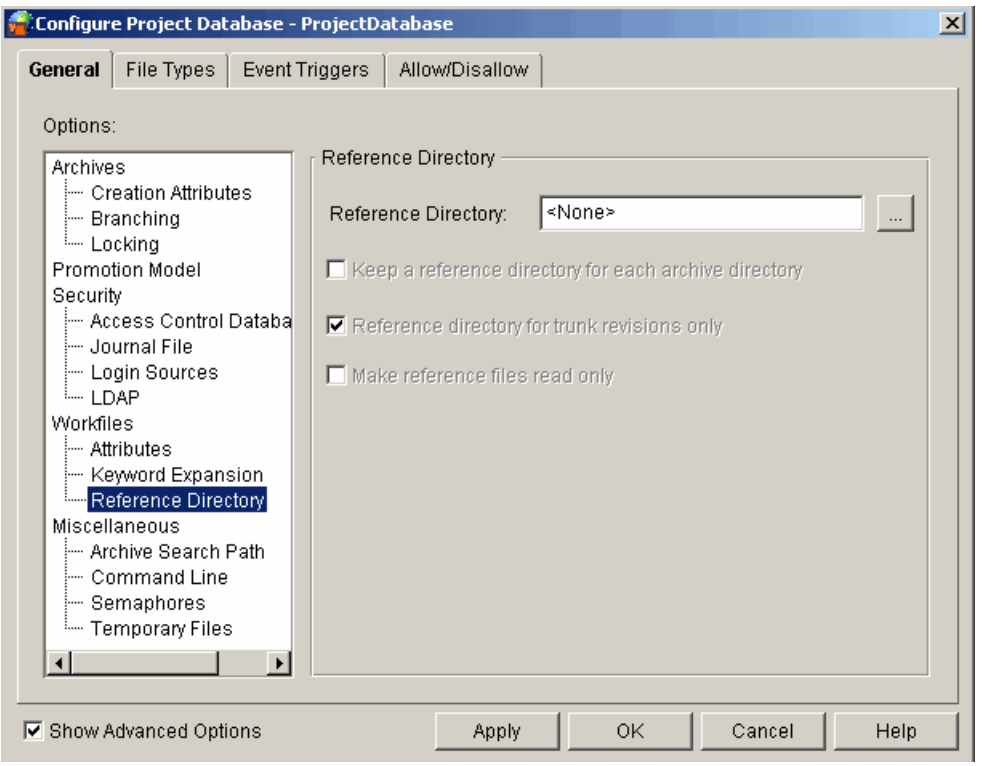
The options that you can set for reference directories are:

- The location of the reference directory. The directive for this option is `ReferenceDir`.
- Whether Version Manager maintains copies of tip revisions from the trunk only or from both the trunk and any branches. The directive for this option is `ReferenceDir=TrunkOnly | All`.
- Whether or not Version Manager write-protects the files in the reference directory. The directive for this option is `ReferenceDir=[No]WriteProtect`.

Desktop client

To set these options in the desktop client:

1. Select Admin | Configure Project. The Configure Project dialog box appears.
2. If not already selected, select the **Show Advanced Options** check box.
3. Select **Reference** Directory beneath Workfiles. The Reference Directory pane appears on the right. The values that display for each option are the settings that are currently defined in the configuration file associated with the project database or project.



4. Enter the location of the reference directory in the **Reference Directory** field.

   You can enter a directory name and make it parallel to the archive directory structure by selecting the **Keep a reference directory for each archive directory** check box. For example, `d:\refdir\project1\foo.c, d:\refdir\project2\bar.c` is parallel to `c:\db\archives\project1\foo.c-arc, c:\db\archives\project2\bar.c-arc`.

5. To maintain copies of tip revisions from the trunk only, select the **Reference directory for trunk revisions only** check box.

6 To write-protect the files in the reference directory, select the **Make reference files read only** check box.

7 Click OK if you are finished defining options, or click Apply to save these settings and continue to define other options.

Command-line interface

The directive that defines the reference directory options is `ReferenceDir`. The default is to make a reference directory in the current working directory and not to write-protect the files in the reference directory.

Refer to the *Serena PVCS Version Manager Command-Line Reference Guide* for complete information about this directive.

Archive Search Path Options

The archive search path options are applicable only to the operation of the command-line interface. The options can be set in either the desktop client or the command-line interface even though they affect only the command-line interface.

The desktop client archive search paths are automatically defined and updated based on the archive locations defined or modified in the project database.

The archive search path options for the command-line interface let you specify:

- Where Version Manager searches for archives when you perform an action on a file. This enables users to perform actions on files without having to know where the archives are stored. See "How Version Manager Searches for Archives" in the *Serena PVCS Version Manager Command-Line Reference Guide* for complete information. The directive for this option is VCSDir.
- Whether or not wildcard specifications match archives in the first archive directory in which they are found. The directive for this option is FirstMatch. This directive is used in conjunction with the VCSDir directive. If only VCSDir is in effect, Version Manager interprets wildcard specifications to match archives in all archive directories.
- How Version Manager searches for archives when you specify only a workfile name. The directive for this option is IgnorePath. When this directive is in effect, Version Manager ignores any path you specify for a workfile and looks in the VCSDir directories for the archive.

By default, when you create a project database in the desktop client, Version Manager sets the VCSDir to the database's archive location in the master configuration file. If you customize and add archive locations in other directories for subprojects, you must update the VCSDir setting so that command-line operations work.

Desktop client

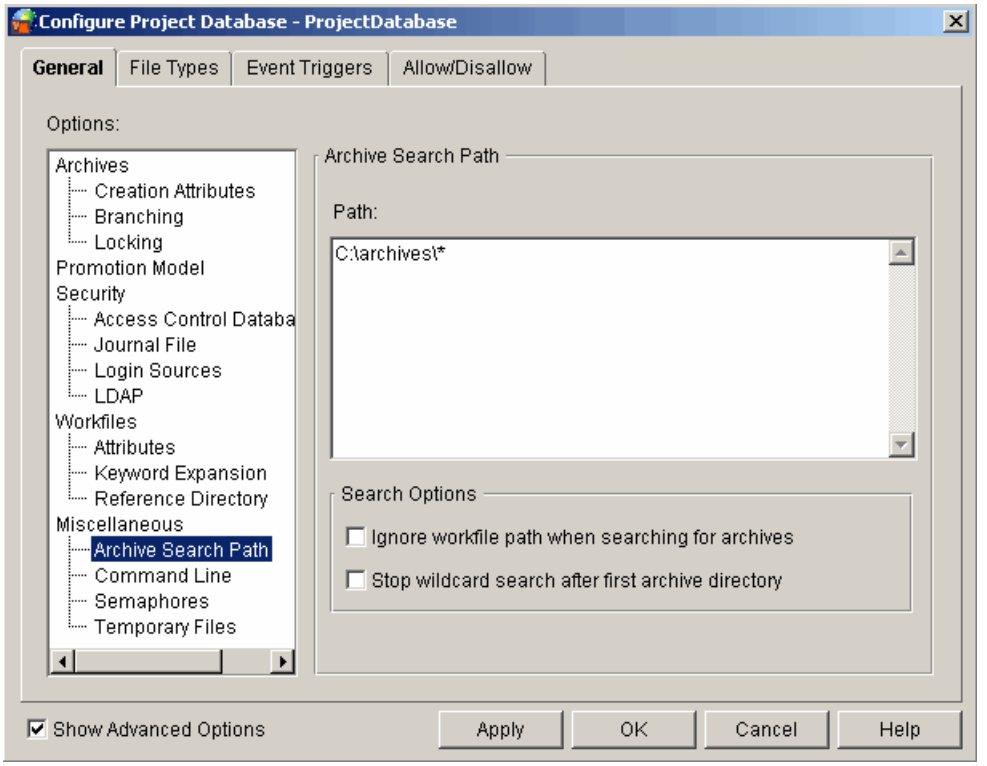
To set these options in the desktop client:

1 Select Admin | Configure Project. The Configure Project dialog box appears.

2 If not already selected, select the **Show Advanced Options** check box.

3 Select **Archive Search Path** beneath Miscellaneous. The Archive Search Path pane appears on the right. The values that display for each option are the settings that are

currently defined in the configuration file associated with the project database or project.



4 In the **Path** box, list the directories in which you want Version Manager to search for archives. Separate each directory path with a semicolon (;). The directories can be absolute or relative path names. For example, for Windows:

```
..\vcs; c:\proj\archives
```

Version Manager searches for archives by looking from left to right in the directories listed.

5 Select the **Ignore workfile path when searching for archives** check box to have Version Manager ignore the workfile path name specified by the user when searching for archives and search only in the directories specified in the Path field above.

6 Select the **Stop wildcard search after first archive directory** check box to have Version Manager stop wildcard searching after the first match. If this is not selected, Version Manager will continue searching all specified archive directories.

7 Click OK if you are finished defining options, or click Apply to save these settings and continue to define other options.

Command-line interface

The directives that define the archive search path options are:

- `VCSDir`, which allows you specify the directories where Version Manager looks for archives. The default is the current working directory.
- `IgnorePath`, which controls how Version Manager searches for archives when you specify only a workfile name. The default is `NoIgnorePath`.
- `FirstMatch`, which determines if Version Manager searches all specified directories when doing a wildcard search or stops searching after the first match. The default is to search all directories.

Refer to the *Serena PVCS Version Manager Command-Line Reference Guide* for complete information about these directives.

Command-Line Options

The command-line options are applicable only to the operation of the command-line interface. The options can be set in either the desktop client or the command-line interface even though they affect only the command-line interface.

The command-line options let you specify:

- Whether or not to display copyright and version information. The directive for this option is SignOn.
- Whether or not to display detailed messages. The directive for this option is Quiet | Verbose.
- The command-line editor to use for entering workfile or change descriptions. The directive for this option is VCSEdit.
- The name and location of the temporary file used for the prompt lines. The directive for this option is VCSEdit.
- The suffix of message files. The directive for this option is MessageSuffix.
- Whether or not to delete message files after Version Manager reads them. The directive for this option is DeleteMessageFile.

Message Files

A message file is a file that you create to provide Version Manager with a change description. Message files enable you to enter change descriptions into a text file. Then, when you check in the workfile, you can use the contents of the message file for the change description instead of typing it in the Change Description field.

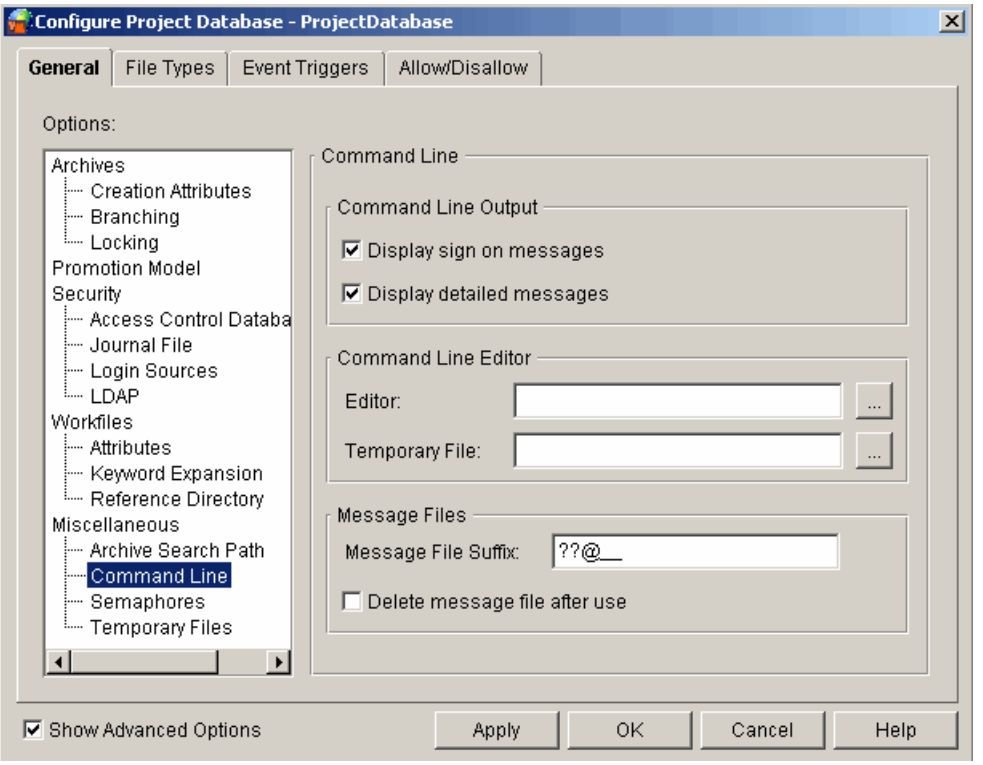
To use the message file when you check in a workfile, use the PUT -M command. You can specify a message file to use (`put-m@`*`message_file`*) or specify to read the message file whose name is computed from the workfile name using the value specified by the message suffix (`put -m@`). The second case is the reason that you will want to specify a message suffix.

Desktop client

To set these options in the desktop client:

1. Select Admin | Configure Project. The Configure Project dialog box appears.
2. If not already selected, select the **Show Advanced Options** check box.

3 Select Command Line beneath Miscellaneous. The Command Line pane appears on the right. The values that display for each option are the settings that are currently defined in the configuration file associated with the project database or project.



4 In the **Command Line Output** group, you can:

- Select the **Display sign on messages** check box to display copyright and version information when certain Version Manager commands are used, for example, the VCS command.
- Select the **Display detailed messages** check box to display details of a command operation such as the workfile and archive names and the current and previous revision numbers.

5 In the **Command Line Editor** group, you can:

- Enter the location and name of the executable for the text editor you want to use for entering workfile or change descriptions. If you do not specify an editor, Version Manager provides a simple internal line editor.
- If you specify a command-line editor to use, you must enter the location and name of a temporary file. Version Manager copies two prompt lines to the temporary file. When the editor runs, these prompt lines remind you what description you are entering. You should not change these prompt lines; Version Manager will delete them for you automatically if you do not change them. The temporary file is deleted after use.

6 In the **Message Files** group, you can:

- Enter the suffix for message files in the **Message File Suffix** field. By default, the Version Manager desktop client uses the suffix +-msg. For example, if the workfile name is windev.c, Version Manager looks for a message file with the name windev.c-msg from which to read the change description.

- Select the **Delete message file after use** check box to delete message files after Version Manager reads them to obtain a change description.

7 Click OK if you are finished defining options, or click Apply to save these settings and continue to define other options.

Command-line interface

The directives that define the command-line options are:

- `SignOn`. The default is to display copyright and version information under Windows and not to display the information under UNIX.
- `Quiet` | `Verbose`. The default is to display details of command-line operations.
- `VCSEdit`. The default text editor is Notepad for Windows and vi for UNIX.
- `MessageSuffix`. The default suffix template for the command-line interface is `??@___` , which causes Version Manager to substitute an at sign (@) for the third character in the suffix of the workfile name. For example, if the workfile name is windev.cpp, Version Manager looks for a message file with the name windev.cp@ from which to read the change description.
- `DeleteMessageFile`. The default is **not** to delete the message file.

Refer to the *Serena PVCS Version Manager Command-Line Reference Guide* for complete information about these directives.

Semaphore Options

The semaphore options let you prevent data corruption by coordinating the activities of programs and processes in multi-user and multi-tasking environments. In these environments, semaphores ensure that Version Manager processes have exclusive access to shared archives, journal files, and the access control database. Semaphores ensure that only one process at a time can write to a file.

Version Manager creates a semaphore when access begins and removes it when access ends. When other users or programs try to access the file, the presence of the semaphore signals Version Manager that the archive is in use.

Version Manager has the following types of semaphores:

- File semaphores are files that Version Manager creates to signal other processes that a file is in use. Version Manager creates file semaphores for files that are being modified and files that are being read. Always use file semaphores for environments composed of heterogeneous network systems. The UNIX version of Version Manager supports only file semaphores.
- **For the command-line interface only:** NetWare semaphores are created using Novell NetWare API calls. NetWare automatically deletes them if a command terminates abnormally. This type of semaphore is created only for files that are being modified.

NOTE For command-line interface users: You must set up Version Manager so that all users of an archive use the same type of semaphore. This is not an issue in the desktop client because only one type of semaphore is available, file semaphores.

The options that you can set for semaphores are:

- The directory where the semaphore file is created. All users of an archive should use the same directory for semaphores. If two users specify different directories, they will

lose all semaphore protection. For this reason, you should set the semaphore directory in the master configuration file, and then disallow the option. The directive for this option is `SemaphoreDir`.

- The delay time between attempts to gain exclusive access to an archive. The directive for this option is `SemaphoreDelay`.
- The number of times Version Manager will attempt to gain exclusive access to an archive. The directive for this option is `SemaphoreRetry`.
- The suffix (extension) of semaphore files. The directive for this option is `SemSuffix`.
- **For the command-line interface only:** The type of semaphore to use for archives stored on a local drive and on a network drive, if any. The directive for this option is `Semaphore`.

NetWare Semaphores in the Command-Line Interface

Version Manager creates NetWare semaphores only for archives that are being updated and only checks for a semaphore on the archive if the archive is being updated.

If Version Manager is not updating an archive, it does not create a semaphore. When other non-update processes need to read the archive, Version Manager opens the archive in a file sharing mode.

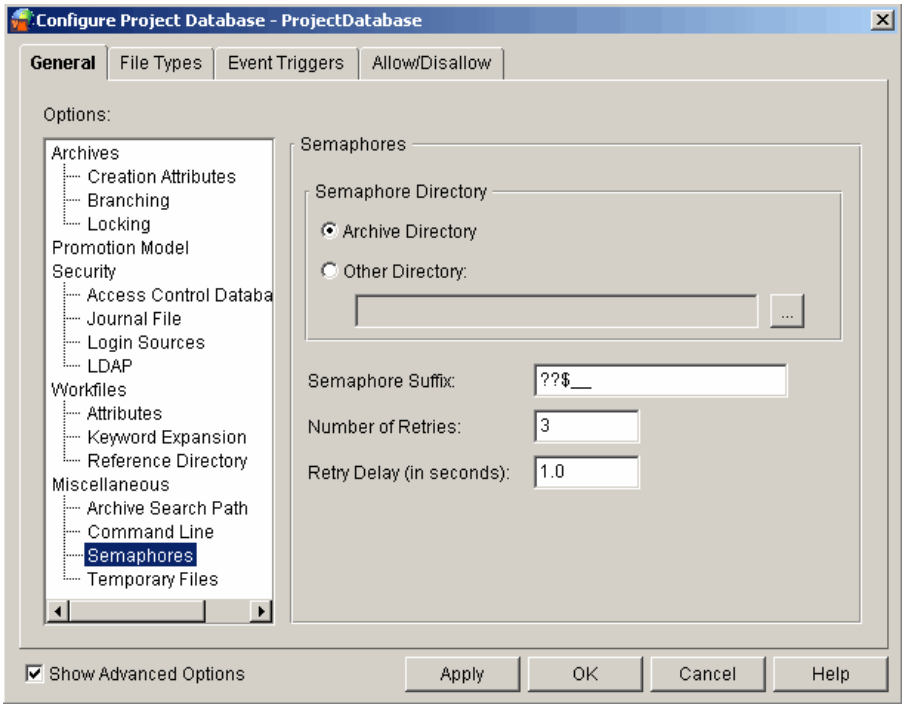
If Version Manager is updating an archive, it creates a semaphore and opens the archive in read-only mode. Version Manager copies the archive to the archive's temporary file directory and closes the archive. At this point, the command that is updating the archive has a private copy of the archive. The presence of the semaphore signals all other Version Manager commands that an update is in progress and no other update operations can take place.

When Version Manager finishes the update to the private copy of the archive, it replaces the original archive with the private copy. If any other process is reading the archive at the time, a sharing violation error is returned to the process that is trying to update the archive. If you have set the retry options for semaphores, the update process waits until the archive is available and then updates it.

Desktop client **To set these options in the desktop client:**

1 Select Admin | Configure Project. The Configure Project dialog box appears.

2 Select Semaphores beneath Miscellaneous. The Semaphores pane appears on the right. The values that display for each option are the settings that are currently defined in the configuration file associated with the project database or project.



3 In the **Semaphore Directory** group, you can either:

- Select **Archive Directory** to place the semaphore files in each archive directory.
- Select **Other Directory** and specify the path of the directory to place the semaphore files in a directory other than the archives directory.

4 Specify the suffix for semaphore files in the **Semaphore Suffix** field. By default, the Version Manager desktop client uses the suffix "+-sem". For example, if the archive file name is windev.c-arc, the semaphore file name is windev.c-arc-sem.

5 In the **Number of Retries** field, specify how many attempts Version Manager will make to gain exclusive access to an archive or journal file in networked environments.

6 In the **Retry Delay (in seconds)** field, specify the delay between attempts to gain exclusive access to archives, journal files, and the access control database.

7 Click OK if you are finished defining options, or click Apply to save these settings and continue to define other options.

Command-line interface

The directives that define the semaphore options are:

- `Semaphore`. The default is to use file semaphores.
- `SemaphoreDelay`, which is used in conjunction with the Semaphore directive to set the delay (in tenths of a second) between attempts to gain exclusive access to archives, journal files, and the access control database. The default is 10 (one second).
- `SemaphoreDir`. By default, Version Manager creates semaphore files in the archive directory.
- `SemaphoreRetry`. The default is three attempts.

- `SemSuffix`. The default suffix template for the command-line interface is ??$___ , which causes Version Manager to substitute a dollar sign ($) for the third character in the suffix of the filename. For example, if the archive file name is windev.c_v, the semaphore file name is windev.c_$.

Refer to the *Serena PVCS Version Manager Command-Line Reference Guide* for complete information about these directives.

Temporary File Options

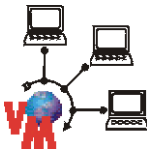
A temporary file is a backup file that Version Manager creates and then deletes to protect an archive and its internal data. For example, if the network goes down while you are checking out a file from an archive stored on the network, and the network crash corrupts the archive, you can retrieve an uncorrupted copy of the archive from the temporary file when the network becomes available.

Version Manager creates two types of temporary files:

- One that contains information about delta processing and other archive manipulations. By default, Version Manager places the temporary files for delta processing in the current working directory. The temporary files are named `PV`*`nnnnnn`*`.TMP`, where *nnnnnn* is a base 32 number; you cannot specify temporary file names.
- One that contains a temporary archive file. By default, Version Manager creates temporary archive files and places them in the current working directory. The temporary files are named `PVCS`*`nnnn`*`.TMP`, where *nnnn* is a hexadecimal number; you cannot specify temporary file names. You can specify a different directory in which to store temporary archive files.

The temporary file options let you set:

- The directory where Version Manager creates temporary files that are generated during delta processing. The directive for this option is `WorkDir`.
- The directory where Version Manager stores the temporary copies of archives before updating them. The directive for this option is `ArchiveWork`.



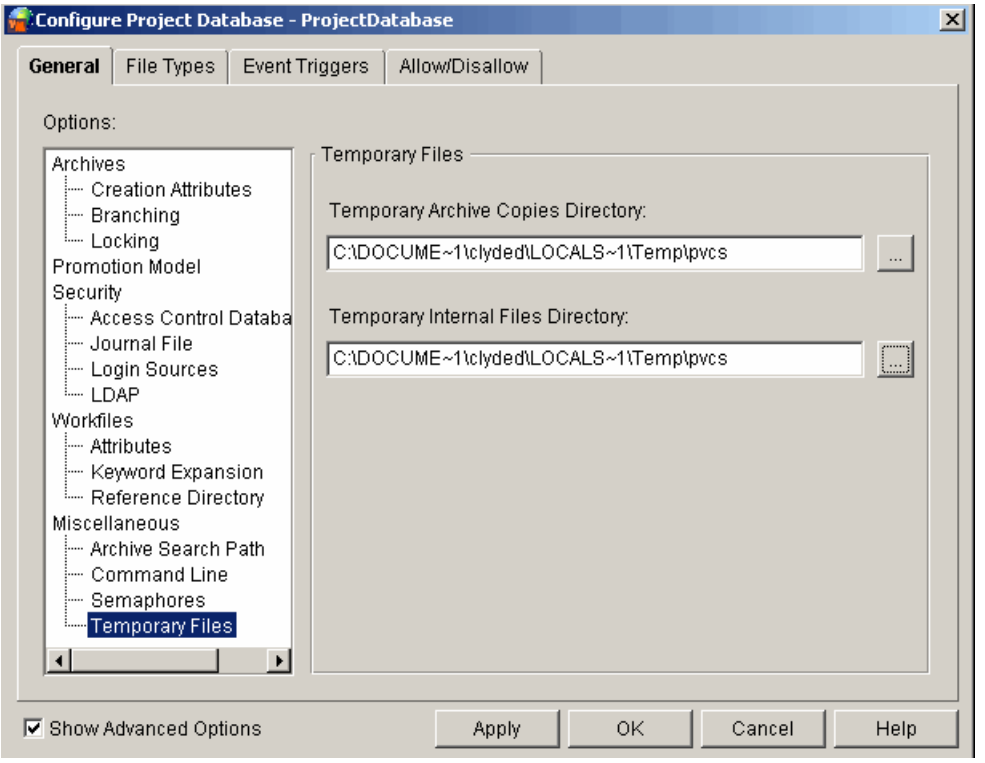
Desktop client

IMPORTANT! If you are using a Version Manager File Server, do not set `WorkDir` or `ArchiveWork` to a directory that is mapped to the file server--unless that location can be accessed via an existing O/S drive mapping. The file server itself cannot be used to access the directories.

To set these options in the desktop client:

1 Select Admin | Configure Project. The Configure Project dialog box appears.

2 Select Temporary Files beneath Miscellaneous. The Temporary Files pane appears on the right. The values that display for each option are the settings that are currently defined in the configuration file associated with the project database or project.



3 Enter the directory in which Version Manager will store the temporary archive copies.

4 Enter the directory in which Version Manager will store the temporary internal files.

5 Click OK if you are finished defining options, or click Apply to save these settings and continue to define other options.

Command-line interface

The directives that define the temporary files options are:

- `WorkDir`, which allows you to specify the directory where Version Manager creates temporary files that are generated during delta processing and other archive file manipulations. This directory is not where the temporary backup archive files are saved. The default is the current working directory.
- `ArchiveWork`, which allows you to specify that temporary archive files be created and the directory where they are saved. The default is to create the files and save them in the current working directory.

Refer to the *Serena PVCS Version Manager Command-Line Reference Guide* for complete information about these directives.

Options Set on a File Type Basis

On the File Types tab of the Configure Project dialog box, you can set the following options on a per file type basis:

- Archive suffix. See "Changing the Archive Suffix" on page 97 for an explanation of archive suffixes. The directive for this option is `ArchiveSuffix`.

- Whether or not to store deltas. See "Storing Deltas" on page 98. The directive for this option is `GenerateDelta`.
- Whether or not to translate the end-of-line character. The directive for this option is `Translate`.
- Whether or not to expand keywords. If you choose to expand keywords, there are two additional options you can set: the comment prefix to insert before lines in the Log keyword expansion and the end-of-line indicator in the Log keyword expansion. The directives for these options are `ExpandKeywords`, `CommentPrefix`, and `NewLine`. See "Keyword Expansion Options" on page 84 for an explanation of the Log keyword.

CAUTION! Keyword expansion will cause corruption in binary files, so Version Manager disables keyword expansion by default. Do not enable keyword expansion for binary files.

- The column masking/renumbering options, which are discussed in "Column Masking/ Renumbering" on page 98. The directives for these options are `ColumnMask` and `Renumber`.
- Record length of fixed-length files so Version Manager can generate deltas based on logical lines. Using this option improves Version Manager's performance when processing fixed-length records. Do **not** use this option for variable length files. The directive for this option is `RecordLength`.

Changing the Archive Suffix

The default archive suffix for the desktop client is to add the characters -arc to the end of the workfile name (for example, myfile.txt-arc), which creates unique archive names. However, the default archive suffix for the command-line interface is to substitute the letter V for the third character in the suffix (extension) of the workfile name (for example, myfile.txv), which can result in identical archive names. For example, using the command-line interface default archive suffix, two workfiles named myfile.cpp and myfile.cpy end up sharing the same archive name, myfile.cpv.

There are two ways that you can define an archive suffix. First you can define an archive suffix that adds characters to the end of the workfile name. To do this, you begin the archive suffix with a plus sign (+). The characters after the plus sign are added to the end of the workfile name. For example, if the archive suffix is +-arc and the workfile name is main.c, the archive name would be main.c-arc.

Second, you can define an archive suffix that consists of two parts—a suffix mask and a default suffix, for example, ??V___. The first three characters are the suffix mask, which determines which characters Version Manager uses for the archive extension if the workfile extension consists of three characters. For example, the questions marks in the default suffix mask direct Version Manager to use the first and second characters of the workfile extension, if they exist. The letter V directs Version Manager to substitute the third character in the workfile extension, if it exists, with a V. If you check in the workfile main.asm, Version Manager would create the archive main.asv.

The second three characters are the suffix default, which determines which characters Version Manager uses for the extension if any of the characters in the workfile extension do not exist. For example, the underscores in the default suffix direct Version Manager to substitute an underscore character (_) whenever a character in a workfile extension does not exist. If you check in the workfile main.c, Version Manager would create the archive main.c_v.

Note that the archive suffix definition of ??V___ should not be used for files with the letter V as the third character in the suffix because the workfile and archive names will be the same (for example, the workfile test.java would have an archive name of test.java).

Storing Deltas

Deltas are changes that define each revision, resulting in smaller archives. By default, if Version Manager determines that a file is a text file (based on the extension), the tip revision of the file is saved in its entirety, while each non-tip revision is saved as a delta.

If workfiles are in binary format, the set of deltas is often as large as the workfile, which makes delta computation time-consuming. In this case, you might want to consider storing complete copies of the workfiles. By default, Version Manager does not store deltas for binary files.

Column Masking/Renumbering

You should set the column masking options when the files that you are creating and modifying contain line numbers, such as COBOL source code files. Masking specifies the columns that should be converted to spaces upon check in and the columns that should be treated as spaces when comparing or merging files. The directive for this option is `ColumnMask`. This directive only affects archives when they are created.

When you add a line to or delete a line from a COBOL file, the lines of the file are automatically renumbered. To generate deltas or a difference report, Version Manager compares files line-by-line to determine what has changed.

Without masking, Version Manager processes every line in the new file as changed because of the difference in line numbers. This results in large deltas and difference reports.

You can also set a renumbering option that tells Version Manager to insert line numbers when checking out revisions. Version Manager only renumbers masked columns beginning with column 1. It fills line numbers with zeroes beginning on the left (000010, 000020, etc.). The directive for this option is `Renumber`.

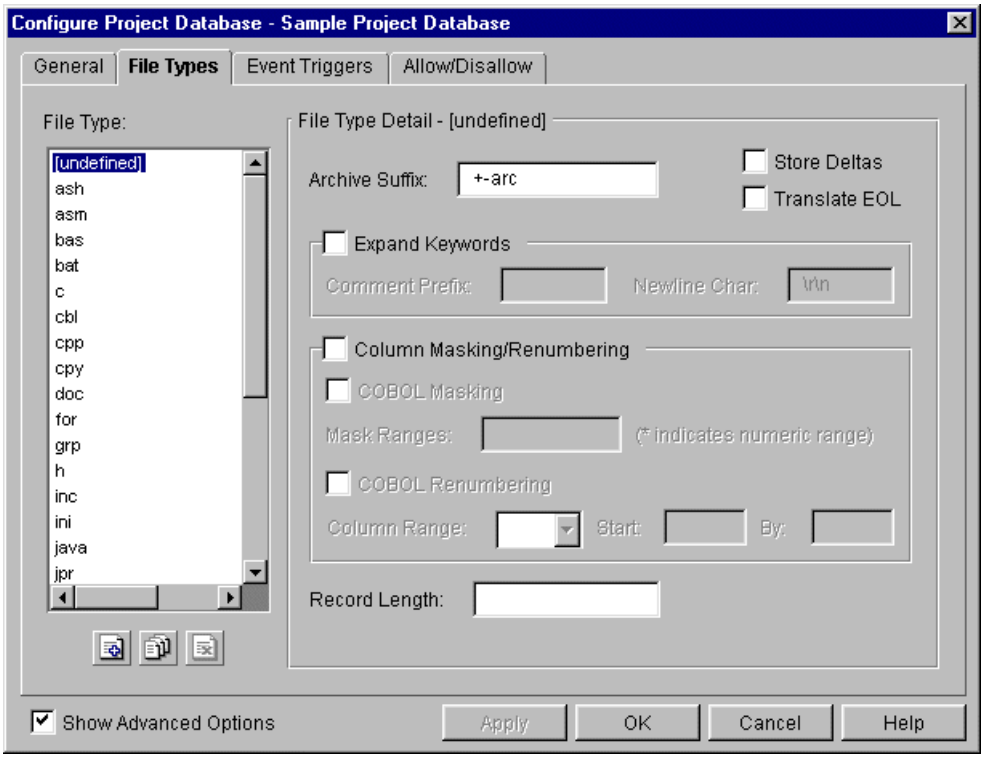
NOTE If you specify columns to be masked that extend beyond the end of a record, Version Manager does not pad the lines with spaces.

Desktop client

To set these options in the desktop client:

1. Select Admin | Configure Project. The Configure Project dialog box appears.
2. If not already selected, select the **Show Advanced Options** check box.

3 Select the File Types tab. The values that display for each option are the settings that are currently defined in the configuration file associated with the project database or project.



4 In the File Type list on the left, select the file type for which you want to define options. **[undefined]** is the default file type; it applies to all file types that are not specifically defined. You can modify the settings of the **[undefined]** file type as you can any other.

5 Do any of the following to manage the File Type list:

- Add a new file type from scratch. The new file type inherits the default settings as defined in the **[undefined]** file type.

- Add a new file type by copying an existing file type. The new file type inherits the settings of the file type you copied.

- Delete a file type.

NOTE You cannot delete the following file types from Version Manager: .c, .h, .pas, .mak, .for, .bas, .prg, .asm, .bat, .obj, .lib, .doc, .y, .l, .o, .a. The desktop client will allow you to delete these file types from the list, but when you reenter the dialog box, the file types will be listed again.

6 Enter the archive suffix in the **Archive Suffix** field. Read "Changing the Archive Suffix" on page 97 for an explanation of how to define an archive suffix.

7 Select the **Store Deltas** check box to store all revisions other than the tip revision as a set of deltas. See "Storing Deltas" on page 98 for an explanation of when to store deltas.

8 Select the **Translate EOL** check box to translate the end-of-line character. You should always select this option for text files.

CAUTION! Do not select this option for binary files; it may cause archive corruption.

9 Select the **Expand Keywords** check box to expand keywords when workfiles are checked in. See "Keyword Expansion Options" on page 84 for an explanation of keywords.

CAUTION! Do not select this option for binary files; it may cause archive corruption.

If you select this option, you can do the following:

- Enter a value in the **Comment Prefix** field to define a comment prefix for the Log keyword. The comment prefix is cosmetic, and it is the responsibility of the developer to enter the Log keyword within comments in the source file.
- Enter a value in the **Newline Char** field to define a new line character for the Log keyword. By default, Version Manager ends lines added for the Log keyword with a carriage return/line feed combination. If your operating system requires a different end-of-line character, you must specify it here.

10 Select the **Column Masking/Renumbering** check box if the files that you are creating and modifying contain line numbers. Then, you can do any of the following:

- Select the **COBOL Masking** check box to use the default COBOL masking definition for COBOL files, which is: Mask columns 73-80 and mask columns 1-6 only if column 1 is numeric.
- If you did not select COBOL Masking, enter the column to mask from and the column to mask to. For example, you could enter 1 and 6; then, Version Manager would ignore columns 1 through 6. To restrict column masking to numeric fields only, enter an asterisk (*) beside the range, for example, 1-6* masks the columns only if column 1 is numeric. You can enter multiple ranges, for example, 1-6*,10-12. Separate ranges with a comma.
- Select the **COBOL Renumbering** check box to use the default COBOL renumbering definition for COBOL files, which is: Renumber columns 1 through 6, start with the number 10 and increment by 10.
- If you did not select COBOL Renumbering, enter the following:
  - The column range to renumber in the **Column Range** field. For example, to renumber columns 1 through 6, you would enter 1-6.
  - The number to start numbering with in the **Start** field.
  - The number by which to increment each line number in the **By** field. For example, if you enter 20 as the start number and 5 as the number by which to increment each line, the first line would be numbered 000020, the second 000025, the third 000030, and so forth. This assumes you are renumbering six columns.

See "Column Masking/Renumbering" on page 98 for an explanation of column masking and renumbering.

11 Enter the length of records in fixed-length files in the **Record Length** field so that Version Manager can generate deltas based on logical lines.

CAUTION! Do not select this option for variable length files. If you do, this can potentially corrupt an archive or cause other unexpected results.

12 Click OK if you are finished defining options, or click Apply to save these settings and continue to define other options.

Command-line interface

The directives that support the file types options are:

- `ArchiveSuffix`. The default is ??V___.
- `GenerateDelta`. The default is to generate deltas.
- `Translate`. The default is to **not** translate end-of-line characters.
- `ColumnMask`. The default is to perform column masking in columns 1-6 for COBOL files. This directive only affects archives when they are created, not existing archives. To change column masking for existing archives, use the VCS command.
- `Renumber`. The default is do not renumber. This directive only affects archives when they are created, not existing archives. To change the renumbering options for existing archives, use the `VCS -XRenumber` command.
- `RecordLength`. There is no default record length.

Refer to the *Serena PVCS Version Manager Command-Line Reference Guide* for complete information about these directives.

Event Triggers

The event triggers tab lets you identify a program that will be executed before or after a specified Version Manager event occurs. An event is a particular action that occurs during Version Manager processing, for example, checking in a file, assigning a version label, and adding an entry to the journal file. The directive for these options is `EventTrigger`.

By default, there are no event triggers defined in the desktop client or the command-line interface.

For complete information about event triggers and how to set them up, see "Using Event Triggers" on page 275.

Allowing and Disallowing Options

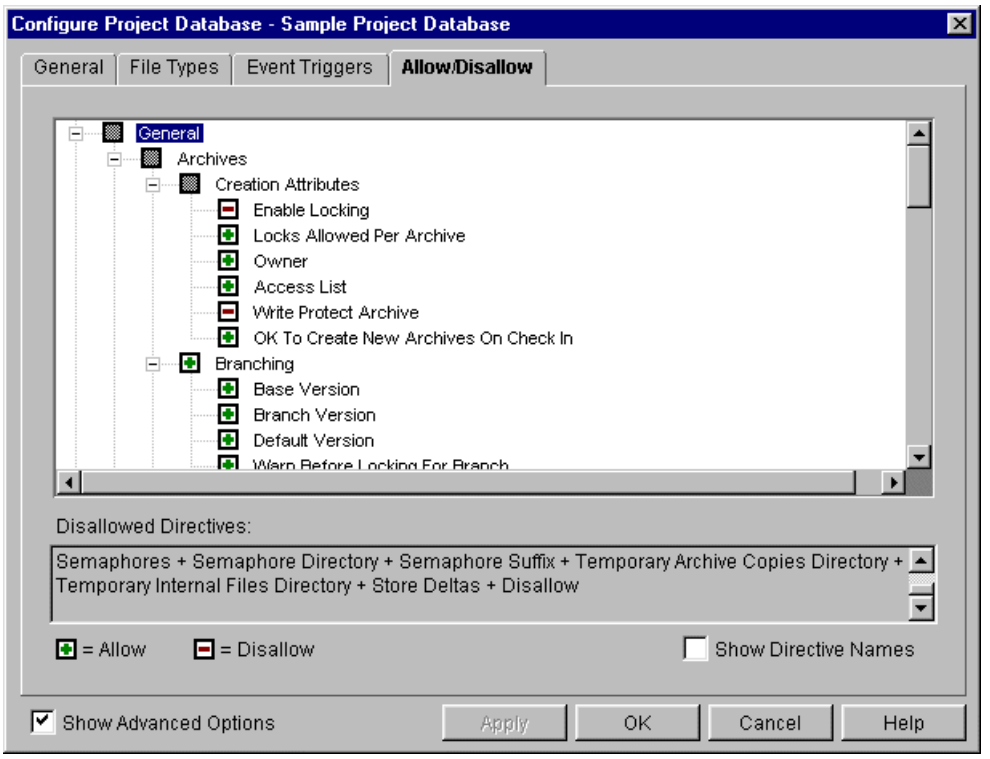
Allowing and disallowing configuration options in the master configuration file controls whether or not users can change the options in other configuration files. Disallowing an option prevents users from changing the option. You can only disallow options in the master configuration file. The Allow/Disallow tab is only available when configuring a project database.

For some options, it is recommended that all projects maintain identical settings, such as the list of sources that Version Manager uses to obtain user IDs, the access control database, the semaphore directory, locking, and if auditing is important, the journal option.

Desktop client

To set these options in the desktop client:

1. Select the project database associated with the master configuration file in which you want to disallow options.
2. Select Admin | Configure Project. The Configure Project Database dialog box appears.
3. If not already selected, select the **Show Advanced Options** check box.
4. Select the Allow/Disallow tab.



5. This tab displays a check box tree of configuration options that you can allow and disallow. The options with a – beside them are disallowed and the options with a + beside them are allowed.

   To change the status of an option, click the check box beside the option. You can allow or disallow an entire set of related options by clicking the parent option. For example, you could disallow all of the Branching options by clicking the check box beside Branching to place a – in the check box.

6. To show the directive name for each option, select the **Show Directive Names** check box. When selected, the directive name appears to the right of the configuration option in parentheses.
7. Click OK if you are finished allowing/disallowing options, or click Apply to save these settings and continue to define other options.

Command-line interface

The directive to disallow configuration options is `Disallow`. To completely disallow a directive in the master configuration file, you must use `Disallow` on both the directive and the `No` form of the directive (if one exists), for example, `MultiLock` and `NoMultiLock`. If you only disallow the directive, then subsequent configuration files can

specify the `No` form of the directive. For example, to prevent the use of `MultiLock`, specify:

```
DISALLOW MULTILOCK NOMULTILOCK
```

NOTE The desktop client sets both the positive and negative forms of a directive when you allow or disallow it.

Embedding a Master Configuration File into Version Manager

Embedding a configuration file into Version Manager ensures that all users will be using the same configuration for Version Manager and that users cannot use a different master configuration file. A master configuration file that is embedded into Version Manager affects all desktop client and command-line users who are using the copy of Version Manager that has the file embedded.

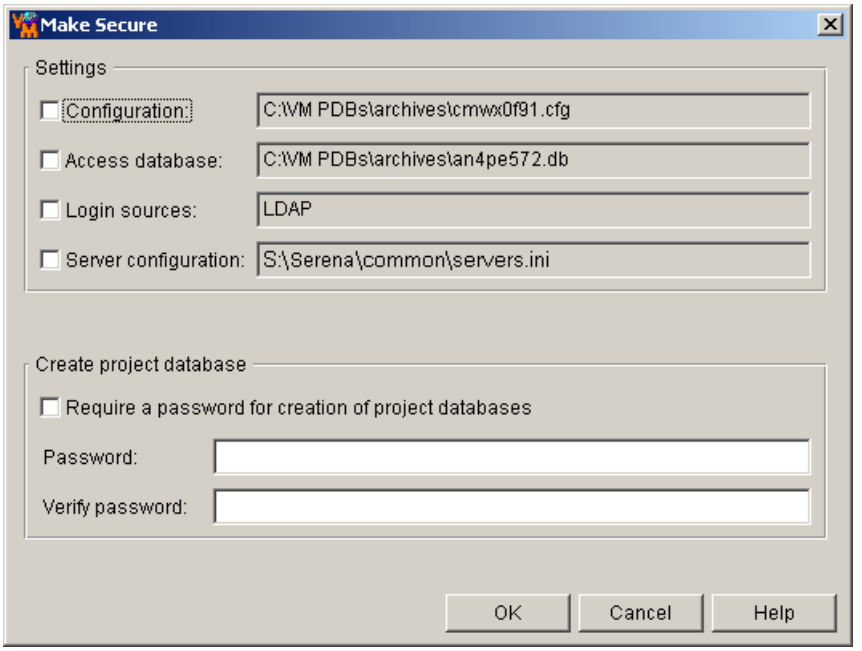
Using the Desktop Client

You can embed one master configuration file into Version Manager, and this configuration file will control how Version Manager operates for all project databases. When you embed a master configuration file, the configuration file that you have associated with each project database becomes a project configuration file. The project configuration file is no longer a master configuration file; therefore, any Disallow options in the configuration file are ignored.

To embed a master configuration file:

1 Select the project database associated with the master configuration file you want to embed into Version Manager.

2 Select Admin | Make Secure. The Make Settings Secure dialog box appears.



3 Select the **Configuration** check box. The field next to this check box displays the location and name of the configuration file you are embedding. You cannot edit this field.

NOTE After you embed the name of the master configuration file, you should move the file (VMWFVC.DLL or vmufvc.a) to the same location as the Version Manager executable files, if the file is not there already. Also, make sure users do not have write permission to this location. You do not want users to be able to embed a different master configuration file into Version Manager or to modify the master configuration file.

4 Click OK.

Using the Command-Line Interface

In the command-line interface, for Version Manager to use a configuration file that you create as a master configuration file, you must embed the name of the configuration file into Version Manager.

To embed the name, use the VCONFIG command:

`vconfig -c`*`config_filename vm_filename`*

where:

`config_filename` is the name of the configuration file that you are embedding.

NOTE vconfig is located in the admin subdirectory of the bin directory.

vm_filename is either:

- VMWFVC.DLL for Windows
- vmufvc.a for UNIX

After you embed the name of the master configuration file, you should move the file (VMWFVC.DLL or vmufvc.a) to the same location as the Version Manager executable files, if the file is not there already. Also, make sure users do not have write permission to this location. You do not want users to be able to embed a different master configuration file into Version Manager or to modify the master configuration file.

For more information about the VCONFIG command, see the *Serena PVCS Version Manager Command-Line Reference Guide*.

Setting Up TrackerLink and SourceBridge

SourceBridge integrates Serena's version control and issue management solutions. This allows you to associate issues created in TeamTrack and Tracker with versioned files when you check files in and out from source control.

SourceBridge setup varies depending on which Version Manager clients and issue tracking system you use (see the following table).

NOTE A specific TeamTrack user privilege is required in order to run TeamTrack SourceBridge. See the TeamTrack SourceBridge documentation.

VM Client	TeamTrack	Tracker
Desktop Client	You must install a SourceBridge component to your Version Manager client. You can download this component using the TeamTrack web client. To do so, click the **About TeamTrack** button in the web client's toolbar. From the About dialog, select the Product Information tab. Click on the Install SourceBridge link. For complete instructions, see the *SourceBridge User's Guide*. Once the SourceBridge component is installed, you must specify a TeamTrack server and login information. To do so, select Tools | SourceBridge (The first time you do this, a SourceBridge Settings dialog box will appear. Subsequently you must click the Settings button on the intervening SourceBridge dialog box). For complete instructions, see the *SourceBridge User's Guide*.	You must install and configure TrackerLink. See the Tracker documentation for details.
Web Client	Your Version Manager administrator must configure each web servlet to work with issue management. For more information, click the **Help** button on the Servlets tab of the Version Manager Application Configuration utility or see the *Version Manager Installation Guide*.	

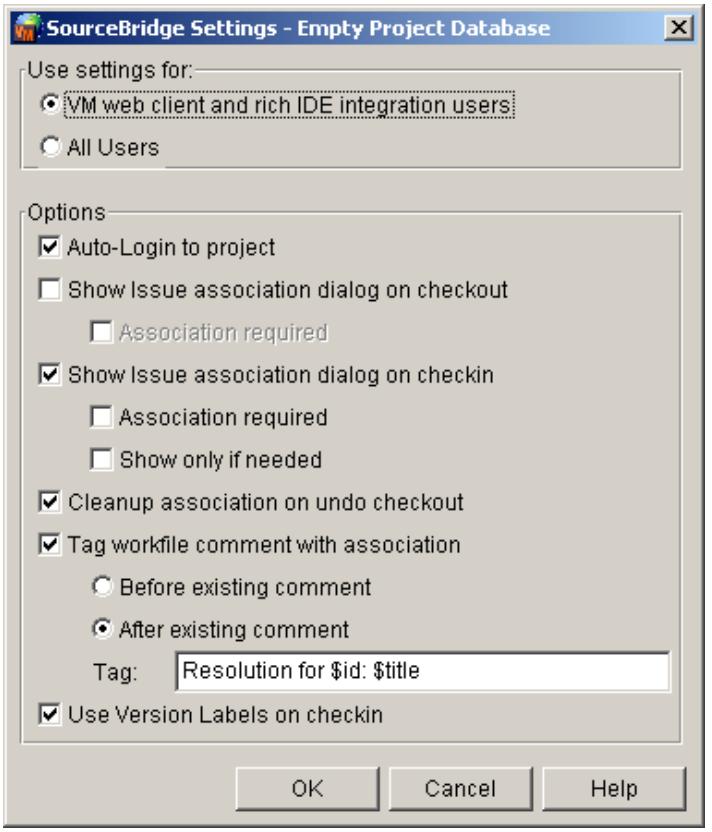
VM Client	TeamTrack	Tracker
SCC IDE's	You must install a SourceBridge component to your Version Manager client. You can download this component using the TeamTrack web client. To do so, click the **About TeamTrack** button in the web client's toolbar. From the About dialog, select the Product Information tab. Click on the Install SourceBridge link. For complete instructions, see the *SourceBridge User's Guide*. Once the SourceBridge component is installed, SourceBridge will be set as your Source Code Control (SCC) provider. By default, it will use TeamTrack as its issue management provider and Version Manager as its source control provider. You can specify what source control provider SourceBridge will use from the System Settings tab of the SourceBridge dialog (but only if it is invoked from an SCC/IDE environment, such as the Test button of the Serena SCC Admin utility). You can change SCC providers via the Version Manager SCC Admin utility (from the Serena folder of the Windows Start menu, select Version Manager | Version Manager IDE Client | Version Manager SCC Admin). See the *Serena PVCS IDE Client Implementation Guide* for more information.	You must install TrackerLink and configure your IDE to use TrackerLink as its Source Code Control (SCC) provider. See the Tracker documentation for details.
Rich IDE Integrations	The rich IDE integations to Eclipse and .NET integrate Version Manager and TeamTrack features into the IDE. See the *Serena PVCS IDE Client Implementation Guide* for more information.	N/A The rich integrations do not work with TrackerLink.

Setting SourceBridge User Options

SourceBridge options are configured in Version Manager and set on the Version Manager project database by administrators with Superuser privileges. The following settings are optional. The settings you choose to set will affect the amount of process control you wish to enforce on your users when associating workfiles with issues.

NOTE The TrackerLink Settings dialog is now named the SourceBridge Settings dialog. It applies to both Serena issue management integrations: Tracker TrackerLink and TeamTrack SourceBridge.

You can access the SourceBridge Settings dialog box from Admin | SourceBridge Settings.



To set up SourceBridge for Version Manager desktop and web client users:

SourceBridge settings reside in the project database. You need to set up the SourceBridge settings for each project database you want to use with issue management. These settings can apply to all Version Manager users, or you can configure them for just Version Manager web client and rich IDE integration users. Note, only three fields affect the rich IDE integrations: **Use Version Labels on checkin**; **Tag workfile comment with association**; and **Association required** (on checkin).

- To enforce SourceBridge settings for just web client and rich IDE integration users, select the **VM web client and rich IDE integraion users** option.
- To enforce SourceBridge for all Version Manager users, select the **All Users** option.

NOTE You may have users who set up individual settings on their own systems. The **All Users** setting will override any individual settings with the ones you set from this dialog.

To set up auto-login for users (TrackerLink only):

Select **Auto-Login to project** to automatically log users in to Tracker Projects. If this option is not selected, users will be prompted to log into Tracker each time they associate workfiles.

To use SourceBridge when checking out or locking:

- Select the **Show issue association on checkout** check box to automatically open the association dialog box when checking out or locking workfiles.

- You can require issue associations with files you are *checking out* or *locking*, by selecting the **Association required** check box beneath the **Show Issue association dialog on checkout** check box. Versioned files or revisions will not be locked or checked out until you associate the file with one or more issues.

To use SourceBridge when checking in or adding workfiles:

- Select the **Show issue association dialog on checkin** check box to automatically open the association dialog box when checking in and adding workfile.
- You can require issue associations with files you are *checking in* or *adding*, by selecting the **Association required** check box beneath **Show issue association dialog on checkin** check box. When this check box is selected, Version Manager automatically brings up the association dialog box when you click OK in the Add Workfiles and Check In dialog boxes. The workfiles will not be added or checked in until the user associates the files with one or more issues.
- (SCC IDEs only) You can limit display of the associations dialog to instances when one or more of the files are not already associated with issues. To do so, select the **Show only if needed** check box.

To clean up associations when unlocking workfiles:

Select the **Cleanup association on undo checkout** check box to clean up all the assigned associations when you choose to unlock workfiles. Associations aren't complete unless the workfile is checked in. Using this "clean up" option helps you to clean up half-done workfile associations.

To tag a workfile comment with an association:

Use tags to record issue-workfile associations in the workfile comment. When adding files, you can associate files with issues. The default text "Initial Description" that Version Manager uses when creating an archive is appended or pre-pended with the issue's information. You can select the format of the comment and which issue information will be included by entering the format in the **Tag workfile comment with an association** field.

- Choose **Before existing comment** to include the text before the comment.
- Choose **After existing comment** to include the text after the comment.

To use issue management to apply version labels:

When the **Use Version Labels on checkin** feature is enabled, a version label containing each issue number associated with the revision is created. For example, a user checks in foo.b, which creates revision 1.4. The user associates the file with two issues: DEF762 and DEF833. When Version Manager creates the revision, two version labels are generated and applied to revision 1.4.

Note that the format of this version label cannot be configured from within Version Manager; it must be set by the issue management administrator.

Complete help for TrackerLink and SourceBridge is available from the Help buttons on the associate dialog boxes.

Working with Both TeamTrack and TrackerLink

You may wish to retain Tracker for some project databases while implementing TeamTrack for others. Version Manager has provisions for users working across both issue

management integrations, but the details vary depending upon which Version Manager client you use. For client specific details, see the following sections.

Desktop Client

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.

Web Client

The servlet for each project database determines which, if any, issue management integration is used. There is nothing to change or configure on the client system when switching between project databases that use a different issue management integration.

SCC IDE's

To use SourceBridge or TrackerLink with an SCC-based IDE, you set one or the other as your SCC provider.

1. Launch the Serena SCC Admin utility (from the Serena folder of the Windows Start menu, select Version Manager | Version Manager IDE Client | Version Manager SCC Admin).
2. Select either **Merant TrackerLink** or **Serena SourceBridge** from the list of SCC providers.
3. Click **OK**.
4. Restart your IDE.

Rich IDE Integration to Eclipse and .NET

The rich integrations to Eclipse and Visual Studio .NET are based on TeamTrack. It is not possible to use TrackerLink with the rich integrations.

NOTE The SCC integrations to Eclipse and .NET can use either TrackerLink or TeamTrack.

For more information

See the Tracker and TeamTrack documentation for more information about installing, configuring, and using issue management.

Setting Up Serena Mover

What is Serena Mover?

The Serena® Mover™ application bridges the gap between your code and application development on one hand, and server distribution and publishing on the other. With

Serena Mover, you can automate the retrieval and deployment of your files and folders across your entire network. Based on the criteria of your choice, Mover extracts files from your Version Manager or Dimensions projects, and deploys them to any mixture of servers.

With Mover managing your deployment processes, you can also automate the transfer of any other files and folders on your development servers, and trigger any command on your network (such as build scripts and server commands). In this way, once you have set up Mover, all code compilation and distribution is handled for you.

Configuring a Connection to a Mover Server

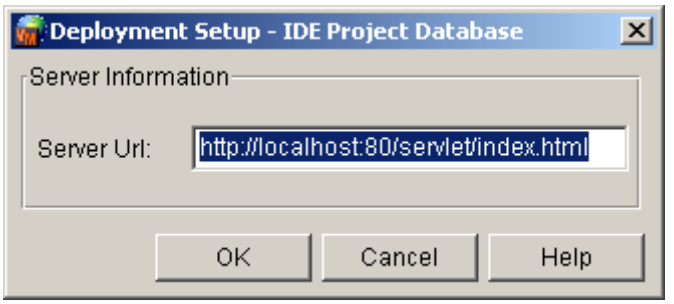
Complete the following procedure to define and store connection settings for the Serena Mover server.

Special Considerations

- Connection settings are stored for each project database.
- Once a connection is defined for a project database, any user who can log in to the project database can log in to Mover.
- You must have Superuser privileges to configure a connection to a Mover server.

To define Mover connection settings:

1. In Version Manager, open the project database for which you want to define the Version Manager URL.
2. Select Admin | Mover Setup. The Deployment Setup dialog box appears.



3. Enter the Version Manager URL in the **Server URL** field as follows:

   `http://server:port/servlet`

   For example:

   `http://Deployment:80/servlet`

4. Click **OK**.

   If Version Manager is unable to establish a connection to the Version Manager server, an error appears. Otherwise, you are returned to the desktop client and can now log in to Version Manager.

Using Version Manager with Mover

You can access the Version Manager Login page from Version Manager, from Dimensions, or directly from your web browser.

NOTE No particular Version Manager or Dimensions permissions are required to log in to Version Manager.

To log in to Version Manager:

1 Do one of the following:

If you are logged in to...	Then...
A browser	Go to the following URL: `http://`*server:port*`/servlet` For example, if Version Manager is installed to port 80 (the default) on a server called Deployment, go to: http://Deployment:80/servlet
Version Manager desktop client	Select Admin | Mover.
Version Manager web client	Select Actions | Mover.

2 At the Version Manager login page, enter your user name and password.

NOTE An administrator must have already set up a user account for you in Version Manager.

If you are logging in for the first time, enter `admin` in both the **Username** and **Password** fields.

3 Click the **Login** button.

4 In the Login Options dialog box, do one of the following:

If you want to...	Then...
Set up user access to Version Manager or create a project	Choose **Administration**.
Create a project	Choose **New project**.
Log in to a project	Choose **Open project** and select the project you want to log in to. **IMPORTANT!** An administrator must have already created a project.

5 Click **OK.**

For more information on setting up and using Mover with Version Manager, see the *Serena Mover Installation and Setup Guide* and the *Serena Mover User's Guide*.

Troubleshooting

This section discusses situations you may encounter when using configuration files.

Configuration File Locked by Another User

If you attempt to open a project database or project and receive an error message stating that the configuration file is locked by another user, and you know that this is not the case, do the following:

1 Delete the configuration file's .lck file, which is located in the same directory as the configuration file.

2 Delete the configuration file's semaphore file, if one exists. Whether or not a semaphore file exists depends on the settings in the configuration file. If the configuration file directs Version Manager to create semaphore files, then one exists for the configuration file. It will be located in the location defined in the configuration file for semaphore files. The extension used for semaphore files is also defined in the configuration file. The default extension suffix in the desktop client is +-sem, which means if the name of the configuration file is cb006nf.cfg, then the name of the semaphore file is cb006nf.cfg-sem.

Chapter 3

Setting Up Version Manager Using the Command-Line Interface

Introduction

Using the command-line interface, you can adapt Serena PVCS Version Manager to a wide variety of project management styles and project sizes. To start this chapter, we provide a section about using the command-line interface. This chapter also provides a few project organization scenarios to help you better understand the different ways you can set up Version Manager, a planning section, procedures for setting up a multiple-person network-based project, and procedures for setting up Version Manager for cross-platform use.

For information about how to set up Version Manager using the desktop client, see "Setting Up Version Manager Using the Desktop Client" on page 13.

About the Version Manager Command-Line Interface

Like the Version Manager 5.3/6.0 command-line interface, the current Version Manager command-line interface enables you to work with archives directly.

However, because the Version Manager desktop client accesses archives through project databases, there are some functions available in the desktop client that are not available in the command-line interface. In addition, to ensure that the same default archive suffix template is used for any archive that you access via both the desktop client and command-line interface, you must use the same configuration file regardless of which interface you use to modify an archive. Information about these issues is provided in this section.

Default Archive Suffix Template

The same archive suffix template must be used for archives that are accessed via both the Version Manager desktop client and the command-line interface. The *archive suffix template* is the pattern Version Manager uses to derive the extension of an archive. The default archive suffix template of the Version Manager desktop client differs from the default archive suffix template of the Version Manager command-line interface:

Version Manager Interface	Archive Suffix Template	Example of Archive Extension
desktop client	+-arc	readme.txt-arc
command-line	??V___	readme.txv

With the desktop client template it is easier to identify the name of the file based on the name of the archive. However, if an archive is accessed from both the desktop client and command-line interface, both interfaces must apply the same archive suffix template to the archive.

The easiest way to ensure that the same template is used in each interface is to use the same configuration file in both the command-line interface and the desktop client. For more information, refer to the *Serena PVCS Version Manager Command-Line Reference Guide*.

Guidelines for Use

Follow these guidelines when using the command-line interface to access archives created with the Version Manager desktop client:

- Always use the same configuration file that was used in the Version Manager desktop client.
- The Version Manager desktop client provides a GET function that extracts a temporary, read-only copy of the default revision to a temporary directory (`\temp\pvcs` in Windows and `/tmp/pvcs` on UNIX). In Windows, the temporary directory is defined by the TEMP environmental variable. On UNIX, it is defined by the pvcsvmux script.

  In the command-line interface, you use the GET command with a lock to check out a writable copy of the latest version of the default revision. Note that although these two commands are similar and share the same name, the behavior is slightly different depending on the interface in which you are working.

Restrictions

The following restrictions apply when working with archives in the command-line interface that were created using the Version Manager desktop client:

- You cannot issue a command for any function related to project databases, such as creating, opening, renaming, or closing project databases. The command-line works with archives directly.
- You cannot issue a command for any function related to projects and subprojects, such as creating, opening, renaming, or closing projects. The command-line works with archives directly.
- You cannot issue a command for any function related to workspaces, such as setting, creating, deleting, or renaming workspaces. For the command-line interface, the workfile location is usually the current working directory. You can specify a different workfile location by identifying an alternative path along with the file. For example: `get foo.c-arc(e:\temp)`
- You can also use a list file, which is explained in the *Serena PVCS Version Manager Command-Line Reference Guide*.

Project Organization Scenarios

This section contains project organization scenarios that range from small single-person projects to multiple-person network-based projects.

Small Single-Person Projects

In this type of project, a single developer maintains up to 25 files. The default configuration of Version Manager works well with projects of this size.

All source and object files, build scripts, and other files are typically located in a single directory to make backing up the files a simple task. The archives are maintained in the

same directory as the source files and are write-protected for security against errors; this is a flat (not hierarchical) structure.

The programmer must check out files with a lock to modify them and then check them back in again.

Larger Single-Person Projects

In this type of project, a single programmer maintains many files. Related files are grouped into directories according to function; this is a hierarchical structure.

To reduce the number of files in each directory, you should place related files together in a directory, and create a subdirectory containing file archives in each directory. We also recommend that you write-protect the archives for security purposes.

Multiple-Person, Network-Based Projects

If developers use a network, project files can be located on a net drive so that all members of the project team have access to them. Related files should be grouped into directories according to function; this is a hierarchical structure.

Team members typically duplicate the project directory structure on their workstation. Because developers generally modify only a small number of the project source files at any one time, they usually keep only the files they are using on their workstation.

Planning a Version Manager Setup

Before you define a Version Manager setup, you need to know the following:

- Where the workfiles, or source files, that you want to place under source control are located
- Where you want the archives for the files to be located
- Where you want users to edit the workfiles
- Whether or not your environment is cross-platform

Workfiles

For large multiple-person projects and large single-person projects, if the workfiles that you want to place under source control are already grouped into directories according to function, you are a step ahead. We recommend that you group the files in directories according to function and mirror this directory structure for your archive and workfile locations.

For small single-person projects, all of the files can be in the same directory, including the archives.

Archive Locations

The archive location is where the archives for the workfiles are located. Typically, for large multiple-person projects, this location is a shared disk storage device on a network; a location where all authorized users have access. It is not required, but we recommend that the archive locations mirror the workfile locations.

NOTE If you are using the Version Manager File Server, the archives may be located on the server. See "Configuring and Using the Version Manager File Server" on page 129.

Workfile Locations

The workfile location is the location where Version Manager places a revision that is checked out of an archive. When you check out a revision, it becomes a workfile.

For large multiple-person projects, we recommend that users create workfile directories that mirror the archive structure on their local workstations.

Cross-Platform Environment

Using the Version Manager command-line interface, you can share configuration files, access control databases, and archives between Windows and UNIX platforms.

You can store your archives on UNIX and access them from Windows, and vice versa.

Setting Up a Multiple-Person, Network-Based Project

The following basic steps provide an overview for setting up a multiple-person, network-based project in a single-platform environment. Following this list are sections that provide detailed procedures.

NOTE All users must use the same Version Manager installation for this set up to work properly. This can be achieved by all users using a network install of the product or using a workstation install of the network installation.

1. On the network, create the archive directories for the project. The directory hierarchy should reflect the organization of the project.
2. On the local workstations, create directories that mirror the archive directories you created in Step 1. These directories will be the workfile locations. Move the workfiles to these directories on your workstation (as the Administrator) so that you can check in the workfiles from these location.
3. In each workfile directory you created in Step 2, create a local configuration file (vcs.cfg) that defines where the archive directory is for the workfiles. The directive that defines the archive directory is VCSDir.
4. Check in the initial revisions of the workfiles to the archive directories to make an archive for each file.

5 Create a master configuration file on the network, and in this file, set the WorkDir and ArchiveWork directives to local locations, rather than network locations. Doing so improves performance by reducing network traffic and also avoids conflicts between the names of temporary files created by different users.

   Also, use the Semaphore directive to prevent simultaneous update operations on archives and the SemaphoreDir directive to set the shared semaphore directory. Everyone must use the same networked semaphore directory.

6 Use the VCONFIG command to embed the name of the master configuration file into Version Manager to set the location of the master configuration file for all users.

Creating Archive Directories

1 Create an archive directory structure that forms a hierarchical relationship that reflects the organization of the project. Related files should be grouped together in a directory according to function.

   For example, if you have a project that has three types of files (for example, include, resource, and source), create a top-level directory named "MyApp" with a subdirectory for each of the file types.

2 Make sure developers have read/write (and execute for UNIX) permissions for the archive directories.

NOTE If you are using the Version Manager File Server for all archive access, you do not need to enable read/write or execute permissions for the archive directories. See "Configuring and Using the Version Manager File Server" on page 129.

Creating Workfile Directories on Local Workstations

As the Administrator, you can perform Step 1 of this task on each developer's workstation, or provide the information to each developer and have them do it.

1 Create a workfile directory structure on local workstations that mirrors the archive directory structure on the network.

2 As the Administrator, on your workstation, move the files that you want to place under source control to these directories so that you can check the files in from this location. You will check them in after you create the local configuration files.

Creating Local Configuration Files

To direct Version Manager to find the archives on the network server, you must create a local configuration file (vcs.cfg) for each workfile directory. This file must contain the VCSDir directive, which specifies the directories that Version Manager uses to look for archives.

1 With a text editor, create a new file. Enter the following line in the file:

   `VCSDir = archive_path`

   where *archive_path* is the directory path of the archive, including the drive letter (in Windows) if it is not the same as the drive you are on. For example, `f:\myapps\source`.

2 Save the file as vcs.cfg in the corresponding workfile directory.

3 Repeat Steps 1 and 2 for each workfile directory.

4 Exit the text editor.

For complete information about configuration files, see "Understanding Configuration Files in the Command-Line Interface" on page 62.

Checking In Workfiles

1 Navigate to a workfile directory.

2 Check each file into the appropriate archive directory, which is defined in the vcs.cfg file. Enter the following command:

   `put workfile_name`

   where:

   workfile_name is the name of a file in the current directory. If you are checking in all of the files from the current directory, you can specify * to check in all of the files at once.

   For example, to check in all of the files from the current directory, enter:

   ```
   put *
   ```

   Another example, to check in a file named retry.c from the current directory, enter:

   ```
   put retry.c
   ```

Creating a Master Configuration File

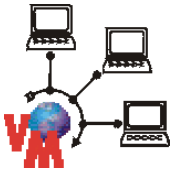
A master configuration file contains common configuration options that define how Version Manager operates globally. Version Manager reads the master configuration file before any other configuration file to ensure that all users share the same settings for certain directives. You can prevent other configuration files (for example, the vcs.cfg file) from overriding common settings in the master configuration file by using the Disallow directive so that users cannot reset the configuration. See "Understanding Configuration Files in the Command-Line Interface" on page 62 for further information.

On a multiple-person network-based project, set the WorkDir and ArchiveWork to local locations. This improves performance by reducing network traffic and also avoids conflicts between the temporary file names created by different users. The users can change these settings in their local configuration files (vcs.cfg) to suit their local environment.

Everyone must use the same type of semaphore to prevent simultaneous update operations on archives. In this case, we tell you to use File semaphores. You must also define a semaphore directory. To do this, you need to set the SemaphoreDir directive to a shared directory. Everyone must use the same networked semaphore directory. Then, you

need to disallow both the Semaphore and SemaphoreDir directives so that users cannot redefine them.

There are many other configuration options you can set in a master configuration file. Read "Configuring Version Manager" on page 51 for complete information about configuration options and configuration files. There is a default master configuration file shipped with the product. This file is on the lib subdirectory of the install directory (for example, `C:\Program Files\Serena\vm\common\pvcsprop\pvcs\vm` in Windows and `/usr/serena/vm/common/pvcsprop/pvcs/vm` on UNIX) and is named `default.cfg`.



NOTE For project databases created on a Version Manager File Server, the default configuration file is named `defaultfs.cfg`.

Add the following lines to the master configuration file:

```
WorkDir = path_name
ArchiveWork = path_name
Semaphore = File
SemaphoreDir = network_path_name
Disallow Semaphore NoSemaphore
Disallow SemaphoreDir
```

where *path_name* is any local directory. For example, for WorkDir, you could define `C:\PVCSTMP`, and for ArchiveWork, you could define `C:\ARCTMP`. Note that these directories must exist on each users' machine.

Embedding the Name of the Master Configuration File

For the Version Manager command-line interface to use a configuration file that you create as a master configuration file, you must embed the name of the configuration file into Version Manager.

NOTE A master configuration file that is embedded into Version Manager affects all desktop client and command-line users that are using the copy of Version Manager that has the file embedded.

To embed the name, use the VCONFIG command:

`vconfig -c`*`config_filename vm_filename`*

where:

config_filename is the name of the configuration file that you are embedding.

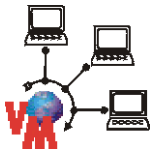
vm_filename is either:

- VMWFVC.DLL for Windows
- vmufvc.a for UNIX

For more information about the VCONFIG command, see the *Serena PVCS Version Manager Command-Line Reference Guide*.

Setting Up Version Manager for Cross-Platform Use

This section describes how to configure Version Manager and assign privileges to directories and files for sharing archives, configuration files, and access control databases between UNIX and Windows users.



NOTE If you are using the Version Manager File Server for all archive access, only certain parts of this section apply. These parts are denoted by the graphic to the left. For more information on the Version Manager File Server, see "Configuring and Using the Version Manager File Server" on page 129.

The instructions in this section assume that you:

- Have installed Version Manager on both your Windows and UNIX systems
- Are using NFS or Samba for sharing file systems between UNIX and Windows

NOTE If users are running Version Manager from Windows but are storing archives on UNIX, all user IDs for Windows users must belong to the same UNIX primary group because of Windows NFS translation.

The configuration files you create for your Version Manager implementation must contain Windows paths. The nfsmap file will map the Windows drive letters to their corresponding UNIX path names so that the same configuration files can be used by both UNIX and Windows.

For information on using setuid on UNIX within a cross-platform environment, see the *Serena PVCS Version Manager Installation Guide*.

The following basic steps provide an overview for setting up Version Manager for cross-platform use. Following this list are sections that provide detailed procedures.

1. On UNIX, we suggest that you create a Serena Administrator user ID such as *pvcs* for you to use and create (if not already created) a group that contains all of the PVCS users, such as *pvcsgrp*. Make this group your primary group or change the group ownership of each PVCS file and directory to the PVCS group.

   The Serena Administrator should own all of the directories that contain configuration files, access control databases, the nfsmap file, and the Version Manager executables, as well as the files within these directories.

   Refer to the documentation for your operating system for more information on creating user IDs and groups.

2. In order for Version Manager to run in a cross-platform environment, you must turn off setuid mode.

3. On UNIX, edit the nfsmap file to map the Windows drive letters to the corresponding UNIX path names so that users on both Windows and UNIX systems can share configuration files. See "Editing the nfsmap File" on page 41.

NOTE

- This step must be performed before you create archives.
- There is a limit of 26 active (uncommented) entries in the nfsmap file. Additional entries will be silently ignored.

4 On UNIX, you, as the Administrator, and each user must set the PVCS_BINDIR environment variable to the location of the directory that contains the nfsmap file and the VMWFVC.DLL for Windows or the vmufvc.a for UNIX.

5 Assign the proper privileges on UNIX and Windows:

- On UNIX:

For...	Assign these privileges...	To...
PVCS users and admin	read, write, execute	Directories that are specified in the configuration files
PVCS users	read, execute	Directories that contain the master configuration file, the access control databases, and the nfsmap file
PVCS users	read	The master configuration file, access control database, and nfsmap file
NOTE If you want users to have the ability to change their own passwords, PVCS users must have read and write privileges to the access control database.		
PVCS admin	read, write, execute	Directories that contain the master configuration file, the access control databases, and the nfsmap file
PVCS admin	read, write	The master configuration file, access control database, and nfsmap file

- In Windows:

For...	Assign these privileges...	To...
PVCS users and admin	full permissions	Directories that are specified in the configuration files
PVCS users	read	Directories that contain the master configuration file, the access control databases, and to these files
NOTE If you want users to have the ability to change their own passwords, PVCS users must have read and write privileges to the access control database.		
PVCS admin	full permissions	Directories that contain the master configuration file, the access control databases, and the nfsmap file and to these files

6 Configure Version Manager to:

- Make archives writable by using the NoWriteProtect directive for new archives and the `vcs -pw` command for existing archives.
- Translate the EOL sequence for text files by using the Translate directive for new archives and the `vcs +pt` command for existing archives.

- **NOT** translate the EOL sequence for binary files by using the NoTranslate directive for new archives and the `vcs -pt` command for existing archives.
- Make user IDs case-insensitive so that the UNIX and Windows systems will read the user IDs the same way by using the `NoCase=VCSID` directive.

7 Make the case of file and directory names consistent for files and directories shared between Windows and UNIX systems.

Turning Off Setuid

On UNIX, use the following command to turn off setuid permissions for Version Manager executables:

`chmod u-s` *`executable`*

where *`executable`* is the following:

■ get	■ regen	■ vjournal
■ ident	■ vcompres	■ vlog
■ makedb	■ vconfig	■ vmrg
■ put	■ vcs	■ vpromote
■ pvcsvmsuid	■ vdel	■ vsql
■ readdb	■ vdiff	

Editing the nfsmap File

Version Manager provides the `nfsmap` file to enable users to share configuration files between Windows and UNIX systems. The `nfsmap` file is a text file that maps Windows drive letters or UNC path names to their corresponding UNIX path names. The file is stored on UNIX. When this file is available, Version Manager uses the MS-DOS path format when reading or writing path names from or to a configuration file.

NOTE You must complete this procedure before you create your archives.

On UNIX, to edit the nfsmap file:

1 Open the `nfsmap` file in a text editor. The default location for the `nfsmap` file is:

`Install_Location``/vm/common/bin/`*`OS`*`/nfsmap`

Where *`OS`* is the name of the operating system. For example, on Solaris, the path would be:

`Install_Location``/vm/common/bin/solaris/nfsmap`

2 Enter each mapping as two columns on its own line, with the drive letter or UNC path name in the left column and the corresponding UNIX directory in the right column. If the UNC path name contains spaces, then it must be enclosed in double quotes (").

Example 1: To map the `M:` drive to the UNIX directory `/product/dev`, enter:

```
M    /product/dev
```

Example 2: To map the UNC path name `"\\myserver\vol2 abc"` to the UNIX directory `/product/dev`, enter:

```
"\\myserver\vol2 abc"    /product/dev
```

IMPORTANT! Each drive letter, UNC path, and UNIX directory must be unique. You cannot repeat a drive letter, UNC path, or UNIX directory in another mapping.

3 Save your changes and exit the editor.

Setting PVCS_BINDIR

On UNIX, set the PVCS_BINDIR environment variable to the location of the nfsmap file and the access control database.

NOTE It is best to either have this variable set in `/etc/profile and /etc/.login` or in each user's .profile and .cshrc files.

To set the PVCS_BINDIR environment variable:

Depending on the user's shell, enter one of the following commands:

For csh: `setenv PVCS_BINDIR` *`directory`*

or

For sh, ksh, bash: `PVCS_BINDIR=`*`directory; export PVCS_BINDIR`*

where *`directory`* is the directory where the nfsmap file and access control database are located. Typically, these files are located in the Version Manager installation directory on UNIX; the default is `/usr/pvcs`.

NOTE Six command-line commands do not read the nfsmap file: vcompres, vdel, vdiff, vsql, readdb, and makedb.

Assigning Privileges

You must assign the proper privileges to users on both UNIX and Windows so that:

- Users can read and write archives but only read the master configuration file, access control databases, and the nfsmap file (the nfsmap file is on UNIX only). Users should **not** have write access to the master configuration file, access control databases, and the nfsmap file because they could reconfigure Version Manager.

NOTE If you want users to have the ability to change their own passwords, PVCS users must have read and write privileges to the access control database.

- Administrators can read and write archives, the master configuration file, access control databases, and the nfsmap file (the nfsmap file is on UNIX only).

To assign privileges on UNIX and Windows:

1. On UNIX, assign read, write, and execute privileges to PVCS users and the Serena Administrator for directories (and applicable subdirectories) that are specified in the configuration files, which could include: the archive directory (VCSDir), the semaphore directory (SemaphoreDir), the work directory (WorkDir), the archive work directory (ArchiveWork), the reference directories (ReferenceDir and VCSDir), and the journal file directory (Journal).

   For each directory, enter the following command:

   `chmod ug+rwx` *`directory`*

2. On UNIX, assign the read and execute privileges to PVCS users for directories that contain Version Manager configuration files, access control databases, and the nfsmap file. Also, assign read, write, and execute privileges to the Serena Administrator for these directories.

   For each directory, enter the following command:

   `chmod ug+rx,u+w` *`directory`*

3. On UNIX, assign only the read privilege to PVCS users for the configuration files, access control databases, and the nfsmap file. Also, assign read and write privileges to the Serena Administrator for these files.

   Navigate to each directory and enter the following command for each of these files:

   `chmod ug+r,u+w` *`file_name`*

4. In Windows, assign full permissions to PVCS users and the Serena Administrator for directories that are specified in your configuration file, which could include: the archive directory (VCSDir), the semaphore directory (SempahoreDir), the work directory (WorkDir), the archive work directory (ArchiveWork), and the reference directories (ReferenceDir and VCSDir).

5. In Windows, assign only the read privilege to PVCS users for directories that contain Version Manager configuration files and access control databases as well as the files in these directories. Users should not be allowed to change the configuration files and access control databases.

6. In Windows, assign full permissions to the Serena Administrator for the files and directories specified in Step 5; the Administrator must be allowed to changes these files.

Configuring Version Manager

In this section, you will configure Version Manager to:

- Make archives writable.
- Translate the EOL sequence for text files and **not** translate the EOL sequence for binary files.
- Make user IDs case-insensitive.

Making Archives Writable

When sharing archives between Windows and UNIX, the archive files must be made writable.

1 Configure Version Manager to make new archives writable by placing the following line in the master configuration file:

```
NoWriteProtect
```

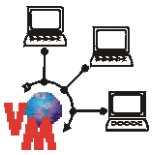
2 Disallow the WriteProtect and NoWriteProtect directives in the master configuration file by placing the following line in the master configuration file:

```
Disallow WriteProtect, NoWriteProtect
```

3 Make all existing Version Manager archives writable by using the following command for each archive:

```
vcs -pw archive_name
```

Translating the EOL Sequence



To share archives on UNIX and Windows, you must configure Version Manager to translate the EOL sequence on UNIX from line feed to carriage return plus line feed when you check in a revision **for text files**, and to make the same translation in reverse when you check out a revision.

CAUTION! You must not translate the EOL sequence for binary files; doing so may corrupt your workfiles.

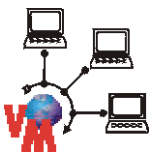
Place the following lines in the master configuration file:

```
NoTranslate
Translate .ext
```

where *.ext* is the file extension of the text file you are archiving. You must enter a Translate line for each type of text file you are archiving; this directive takes only one file extension per line. See the *Serena PVCS Version Manager Command-Line Reference Guide* for complete information about the Translate directive.

The Translate directive only specifies the default to use when creating an archive; it has no effect on existing archives. After an archive has been created use the `vcs +pt` command to enable translation for text files and the `vcs -pt` command to disable translation for binary files.

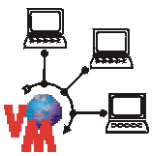
Making User IDs Case-Insensitive



For Version Manager to read user IDs the same on UNIX and Windows systems, you must place the following line in the master configuration file:

```
NoCase=VCSID
```

Making the Case of File and Directory Names Consistent



To make the case of file and directory names consistent for files and directories shared between Windows and UNIX, do one of the following:

- Create all new archives on UNIX so that the default workfile names contain the proper case.

- Use the VCS command on archives created in Windows to change the workfile names in the archives to reflect the proper case. We recommend that you use lowercase.

IMPORTANT! In a cross-platform environment, you **cannot** place files into the same folder if they differ only by case. Such usage is possible only in UNIX-only environments.

What's Next

After you set up Version Manager, you can:

- Configure Version Manager. See "Configuring Version Manager" on page 51.
- Add security. See "Using Security" on page 203.
- Define a promotion model. See "Using Promotion Models" on page 247.
- Define event triggers. See "Using Event Triggers" on page 275.

Chapter 4

Configuring and Using the Version Manager File Server

About the Version Manager File Server

The Version Manager File Server is a server that accesses archives, metadata archives, revisions, and other files on behalf of Version Manager users.

This optional feature is new as of Version Manager 8.

Why Use the Version Manager File Server?

The Version Manager File Server allows you to configure Version Manager for improved security and performance.

Improved Security on Windows

Previously, Version Manager users needed write/delete access to the archive and metadata files on the file system. This approach left open the possibility that a user could maliciously or accidentally delete archive or metadata files with tools such as Windows Explorer or the command prompt of the O/S.

With the Version Manager File Server, you can eliminate this vulnerability since only the file server itself needs write/delete access to the archive and metadata files.

NOTE The Version Manager web client has always offered the security advantage of secure archives, but only if you could limit who had access to the CLI, PCLI, and desktop client. All clients on UNIX could always be made secure.

Improved Performance

With the Version Manager File Server, you can increase the speed of most source control operations. Your net performance gain depends on the characteristics of your work and work practices as well as how closely you model your configuration on the ideal, as outlined below:

- Locate all archive and metadata files on the system running the file server, rather than having the server access them via mapped drives. This saves time and reduces network traffic by allowing:
  - Files to be streamed across the network instead of being read one block at a time.
  - Certain sub operations to be performed on the file server, avoiding the need to transfer files over the network.
- Split your archives into separate revision and metadata files. This allows faster:
  - Metadata operations since metadata archives are quite small in comparison to an archive that also includes revisions.

NOTE Metadata includes version labels, promotion groups, check out dates, change descriptions, etc.

  - Revision operations since revisions can be stored as individual flat files.
  - Access to revisions in a Delta store since revisions can be cached outside of the Delta store for faster retrieval.

How Compatible is the Version Manager File Server?

Implementing the Version Manager File Server is not an all-or-nothing undertaking. In short you can:

- Try it out on just some of your project databases without affecting the others.
- Allow users to access the same projects and archives via the file server (using Version Manager 8 or later) and directly (using Version Manager 5.3 or later).

NOTE Revision libraries (split archives) cannot be accessed directly or with releases of Version Manager prior to 8. They can be accessed only via the file server.

- Recombine (unsplit) revision libraries and metadata archives to return them to the original (inflated) archive format which can be accessed directly by Version Manager 5.3 or later.

The following table maps the compatibility of Version Manager releases and archive formats with the File Server and Revision Library features of the Version Manager File Server.

Client Release	Archive Format	File Server	Revision Library	Compatibility Notes
8 +	8 + (split)*	X	X	All security and performance features are available.
8 +	5.3 - 8 +	X	n/a	All security and *some* performance features are available.
6.5 - 7.5	5.3 - 8 +	X	n/a	These clients can directly access unsplit archives on the file server via a mapped drive. However, since they do not actually use the file server itself, there are no performance gains for these clients. Except for the web client, these clients require write/delete access to the archive files. Allowing such access defeats the key security improvement in the file server.
5.3 - 7.5	8 + (split)*	n/a	n/a	These clients cannot be used with the file server so they cannot read split archives. However, you can unsplit the archives to return them to the inflated format.
* **NOTE** Version Manager 8 + can use the same archive format as Version Manager 5.3 - 7.5. However, to create a Revision Library, you must split archives into separate revision and metadata files; these split archives are not backward compatible with Version Manager 5.3 - 7.5 clients. You can recombine (unsplit) revision libraries and metadata archives to return them to the original (inflated) archive format.				

How Does the Version Manager File Server Work?

The Version Manager File Server performs two main functions:

- **File Server:** Accesses archive and metadata files on behalf of Version Manager users.

  This allows increased security since users no longer need file system write/delete access to the archive files. It also provides some performance advantages.

- **Revision Library:** Stores revision data separately from metadata (split archives).

This allows increased performance since metadata operations need not open revision data files. Also, some operations can take place directly on the server, reducing network traffic and operation times.

In a Revision Library, revisions are stored as:

- **Delta Files (delta.d):** Stores the entire tip revision and describes all other revisions as changes from it. (Also known as reverse deltas.)

  By default, Version Manager stores only text files as Deltas. For more information, see "Options Set on a File Type Basis" on page 96.

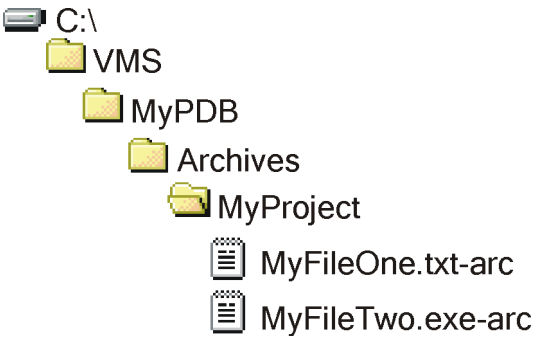
- **Flat Files (*RevisionNumber*u):** Stores each revision as a separate file. For example, 1.4u

NOTE You may occasionally see files named `delta-todo.d`. Such a file may be created when you check in a revision on a busy server. It serves as a note to the file server that there is a new revision--currently in the form of a flat file--that needs to be added to the delta file. Once the server adds the revision to the delta file, it deletes the `delta-todo.d` file.

Since the file server handles all access to Revision Libraries (split archives) and everything works as it always has, the end user need not worry about these server-side details. However, the Revision Library function is not compatible with inflated (unsplit) archives or releases of Version Manager prior to 8.

File Server Feature Only

If you implement the file server feature of the Version Manager File Server but not the revision library feature, the metadata and revision data are stored together in a single (inflated) archive for each file. This is how Version Manager archives have always worked, but now the file server will control access to them. This might look like the following:



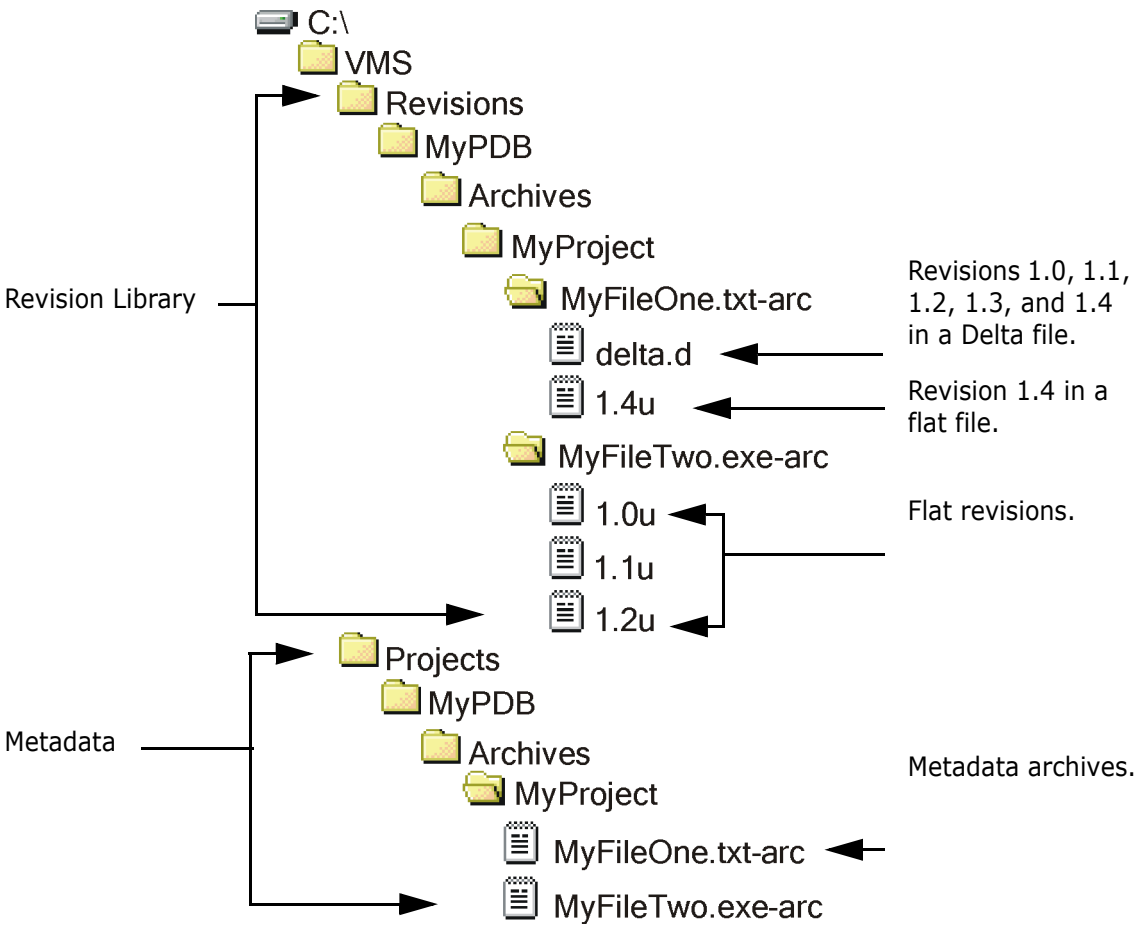
Where the VMS directory is mapped to a Version Manager File Server. Any project databases at or below the VMS directory would be controlled by the file server.

File Server and Revision Library Features

If you implement the revision library and file server features of the Version Manager File Server, the archives are split into separate metadata and revision data components and stored on the file server.

- The metadata is stored in metadata archives.
- Each revision is stored in a Revision Library as a flat file or as an entry in a delta file.

This might look like the following:



In the above example the:

- Revision Path is mapped to `C:\VMS\Revisions`. This is the root of the Revision Library. The directory structure of the Revision Library is based on the structure of the Metadata Store and the names of the archive files. This structure should not be manually edited.
- Project DB Path is mapped to `C:\VMS\Projects`. This is the root of the metadata store. This directory structure is related to the Revision Library and should not be manually edited.
- Name of the project database is `MyPDB`. If there were other project databases in this example, they would be represented by directory structures similar to the structure that starts with the MyPDB directory.
- Name of the project is `MyProject`.

IMPORTANT! If you manually move a metadata or revision data file you will break the relationship between them. See "Exporting, Importing, Moving, Renaming, and Fixing Archives" on page 162.

Installing the Version Manager File Server

The Version Manager File Server is an optional feature you can select during Version Manager installation. For installation instructions, see the *Serena PVCS Version Manager Installation Guide*.

About Administering the Version Manager File Server

The Version Manager File Server includes the Version Manager File Server Administration Utility. With this browser-based utility, you can:

- Manage the list of users authorized to use the Version Manager File Server Administration Utility. See "Managing Administrative Users" on page 136.
- Map project databases to the server. See "Managing Project Database and Revision Library Paths" on page 137.
- Map revision libraries to the server. See "Managing Project Database and Revision Library Paths" on page 137.
- Configure the server. See "Configuring the File Server" on page 141.
- View the status of operations on the server. See "Viewing Server Status" on page 145.
- View a log of server errors. See "Viewing the Server Log" on page 147.

Starting and Stopping the File Server

The Version Manager File Server is a servlet that runs on the Version Manager Application Server. This servlet is automatically installed and configured when you install the Version Manager File Server. To start or start the Version Manager File Server, you start or stop the Version Manager Application Server.

Once you have started the file server, it will continue to run until you shut it down manually or shut down the machine on which it is installed.

Starting or Stopping the File Server on Windows

IMPORTANT! This starts or stops the Version Manager Application Server, which starts or stops the Version Manager File Server, the Version Manager Web Server application, and the WebDAV Server, if installed.

NOTE If you are using Microsoft Vista as the operating system, you must be an administrative user to start the **Version Manager Application Server Admin Services**.

To start the file server on Windows:

1. Select Serena | Version Manager | Version Manager Application Server from the Start menu on the system to which you installed the file sever. The Version Manager Application Server Admin appears.
2. Click the **Start** button on the Servers tab.

NOTE You can also install the Version Manager Application Server as an NT service by clicking the **Install NT Service** button. This service is named Serena VM Web Application Server.

To stop the file server on Windows:

1. Select Serena | Version Manager | Version Manager Application Server from the Start menu on the system to which you installed the file sever. The Version Manager Application Server Admin appears.
2. Click the **Stop** button on the Servers tab.

Starting or Stopping the File Server on UNIX

Once you have started the file server, it will continue to run until you shut it down manually or shut down the machine on which it is installed.

Special Considerations

- Start and stop the Version Manager Application Server as the user who owns the project databases. Do not run it as root.
- This starts or stops the Version Manager Application Server, which starts or stops the Version Manager File Server, the Version Manager Web Server, and the WebDAV Server, if installed.

To start the file server on UNIX:

1. Change to the following directory:

   `/`*`VM_Install_Dir`*`/vm/common/bin`

2. Enter the following command:

   ```
   ./pvcsstart.sh
   ```

To stop the file server on UNIX:

1. Change to the following directory:

   `/`*`VM_Install_Dir`*`/vm/common/bin`

2. Enter the following command:

   ```
   ./pvcsstop.sh
   ```

Launching the Version Manager File Server Administration Utility

Special Considerations

- The Version Manager File Server must be running in order for you to launch the Version Manager File Server Administration Utility. See "Starting and Stopping the File Server" on page 134.
- You can configure the file server to allow remote administration. By default, the Version Manager File Server Administration Utility can be run only on the server itself. See "Configuring the File Server" on page 141.

To launch the Version Manager File Server Administration Utility:

1 Enter the following URL into your browser:

`http://`*`ServerName`*`:`*`Port`*`/serenafs/Admin`

Where:

- *`ServerName`* is the name or IP address of the computer on which the file server is installed.
- *`Port`* is the port number assigned to the file server. By default, the Version Manager File Server uses port 8080.

2 Press ENTER. A login page appears.

TIP Bookmark the login page in your browser.

3 Enter a valid administrative ID and password.

NOTE To login the first time, use the default user ID and password, both of which are **Admin**.

4 Click the **Login** button.

Managing Administrative Users

A regular Version Manager user ID is not sufficient to login to the Version Manager File Server Administration Utility; you must use an ID and password that has been defined in the Administrators pane of the utility.

To manage users:

1 Launch the Version Manager File Server Administration Utility.

NOTE To login the first time, use the default user ID and password, both of which are **Admin**.

2 Select **Administrators** in the left pane. The Administrators pane appears.

3 Do any of the following:

Add a user

To add a new user:

a Click the add ( ) button. The Add Administrative User dialog box appears.

b Enter a new user ID in the **Administrator Name** field. The user ID must not already exist and it must not contain a non-breaking space character (_). User IDs are case sensitive.

c Enter a password in the **Password** field. Passwords are case sensitive and must be at least five characters long.

d To verify that you typed the password correctly, enter it again in the **Retype Password** field.

e Click **OK**.

Change your password

To change your password:

a Select a user ID.

b Click the edit ( ) button. The Edit Administrative User dialog box appears.

c Enter a new password in the **New Password** field.

d To verify that you typed the new password correctly, enter it again in the **Retype New Password** field.

e Click **OK**.

Delete a user

To delete a user ID:

a Select a user ID.

b Click the delete ( ) button. A confirmation dialog box appears.

NOTE You cannot delete the user ID you are logged in as, but you can delete any other user whether or not they are currently logged in.

c Click **OK**.

Managing Project Database and Revision Library Paths

You must specify the paths to the project databases and revision libraries you want the file server to access.

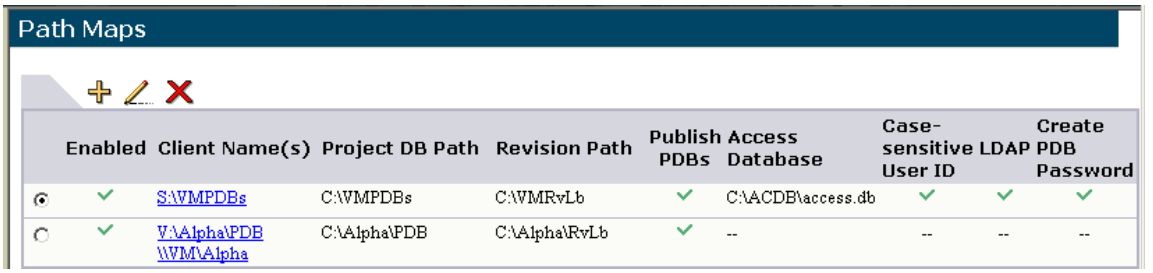
Special Considerations

- Before specifying paths, be sure you understand the relationship between the Revision Path and the Project DB Path. See "How Does the Version Manager File Server Work?" on page 131.
- If you have a large number of users, large archives, or a large number of archives, you may benefit from dividing the work load among multiple file servers. See "Assigning Work to Multiple File Servers" on page 141 .

To manage paths:

1 Launch the Version Manager File Server Administration Utility.

2 Select **Path Maps** in the left pane. The Path Maps pane appears.



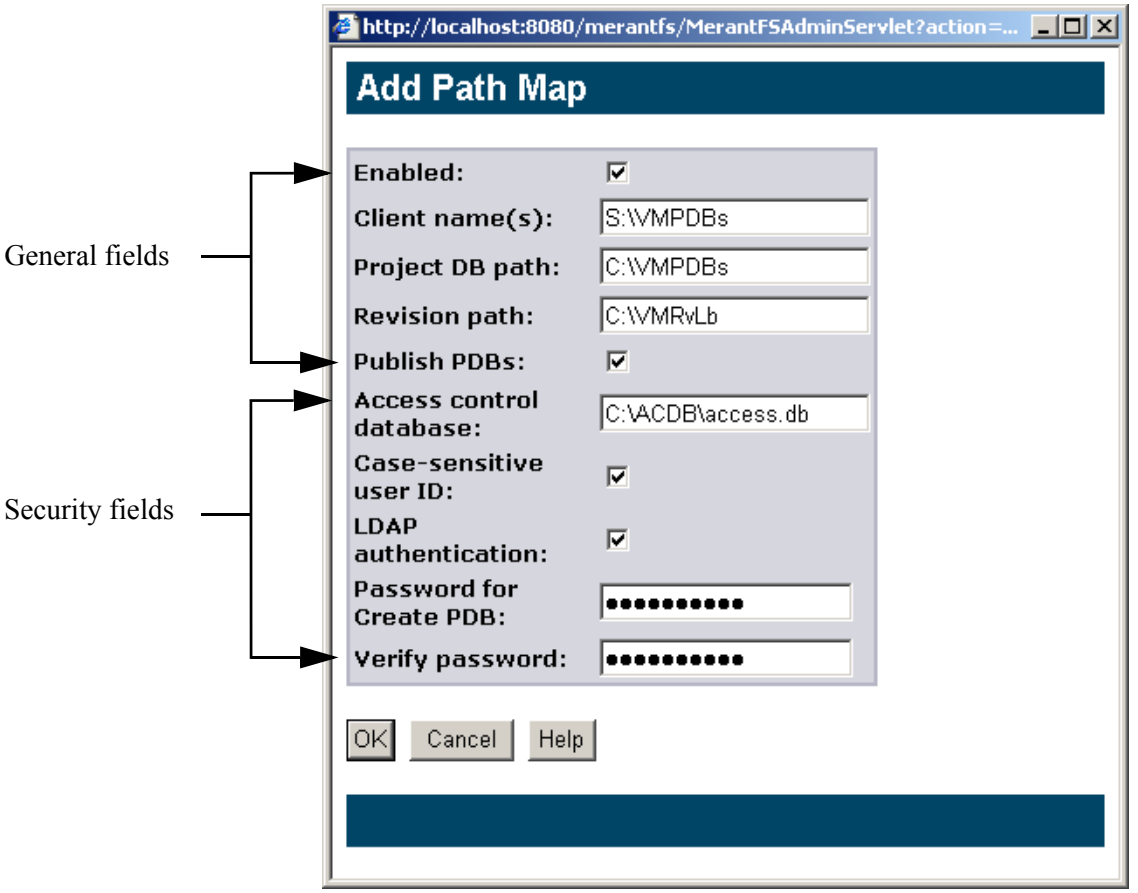
3 Do any of the following:

- To add or edit a path map, see "Adding or Editing a Path Map" on page 138.
- To configure security options for a path map, see "Configuring Path Map Security Options" on page 140.
- To delete a path map, see "Deleting a Path Map" on page 141.

Adding or Editing a Path Map

To add or edit a path map:

1 To add a new path, click the add ( ) button; to edit an existing path, select it and click the edit ( ) button. The Add Path Map or Edit Path Map dialog box appears.



NOTE This section describes the general fields. For information on the security fields, see "Configuring Path Map Security Options" on page 140.

2 Select **True** from the **Enabled** list to activate a server path and make it available to clients.

3 Enter a virtual path in the **Client Name** field. This is the path as it will appear in clients. The path must be unique. Use semicolons (;) to separate multiple virtual paths (as when you specify one for UNIX clients and one for Windows clients).

Special Considerations

- The easiest way to create a path map for an existing project database is to set the **Client Name** to match the existing path to the project database. For example, if your existing project databases are located in a directory on a mapped network drive, `S:\VMPDBS`, you would set the **Client Name** to `S:\VMPDBS`.
- The Client Name need not correspond to an actual path (unless you are mapping a pre-existing project database, as above), but it must look like a genuine O/S or UNC path to the clients. UNC paths must start with two backslashes (\\).
- A Client Name can be the parent or child of another Client Name only if the associated Project Database Paths are related to each other in the same way. For example, the following paths would be valid:

Client Names	Project Paths
\\client\parent	C:\project\one
\\client\parent\child	C:\project\one\child

- If a path must be accessed by both UNIX and Windows clients, you can enter a client name in the correct form for each O/S, using a semicolon to separate them, or you can access the path via a Windows-style path from both Windows and UNIX systems.

  For more information on working in a cross-platform environment, see "Using the File Server in a Cross-Platform Environment" on page 164.

4 To specify a path to a project database or to the root of a directory structure that contains many project databases, enter the path in the **Project DB Path** field. The path must not contain semicolons (;).

5 In the **Revision Path** field, enter the path to a directory under which you want to create a Revision Library. The path must not contain semicolons (;).

IMPORTANT! Do not specify long **Project DB Path** or **Revision Path** values or the complete path to your archives may exceed the path-length limitation inherent in your operating system.

6 To search the **Project DB Path** for project databases and make the resulting list available to clients, select **True** from the **Publish PDB** field. Users can then select project databases from a list displayed in the client. If you do not enable this feature,

only users who know the path and manually enter it will be able to access the project database.

NOTE See "Configuring the File Server" on page 141 for information on limiting how much of the directory structure is searched when building the Published PDB list.

7 Click **OK**.

IMPORTANT! These steps alone do not add a new or existing project database to the Version Manager File Server. See "Adding Project Databases to a Version Manager File Server" on page 148.

Configuring Path Map Security Options

NOTE This section describes the security fields of the Add/Edit Path Map dialog box. For information on the general fields, see "Adding or Editing a Path Map" on page 138.

Configuring path map security is optional, but recommended.

To configure path map security options:

1 To add a new path, click the add ( ) button; to edit an existing path, select it and click the edit ( ) button. The Add Path Map or Edit Path Map dialog box appears.

2 **Access control database:** Specify an access control database to associate with the path map. Once an access control database is specified, all access to the path map will be validated against it. You will not be able to create project databases for the path map without a valid user ID. If you specify a non existent access control database, no one will be able to access the path map.

NOTE

- This access control database controls only the access to the path map. It does not replace or override any existing access control databases in the project databases, nor does it pass privilege or user group information to the clients.
- Your account name and password must match both the path map access control database and the access control database in effect for the project database, if any.

3 **Case-sensitive user ID:** Specify whether the access control database expects case-sensitive user IDs. By default, this option is enabled.

4 **LDAP authentication:** Specify whether LDAP authentication is enabled for the path map. By default, it is not. The Version Manager File Server will look for configuration information in the following file:

VM_Install\vm\common\bin*OS*\pvcsldap.ini

NOTE To enable LDAP authentication for a path map, you must configure LDAP authentication for the server. See "LDAP Configuration Options" on page 79.

5 **Password for Create PDB:** To require a password for create project database operations, specify it here.

6 **Verify password:** Re-enter the password to verify that you entered it correctly.

More information

For more information on configuring Version Manager security, see the following table:

For information on...	See...
Configuring file server security options	"Configuring the File Server" on page 141
File server security considerations	"Security Considerations" on page 165
Configuring Version Manager security	"Using Security" on page 203

Deleting a Path Map

To delete a path map:

1 Select the path map you wish to delete.

2 Click the delete () button. A confirmation dialog box appears.

3 Click **OK**.

Assigning Work to Multiple File Servers

If you have a large number of users, large archives, or a large number of archives, you may benefit from dividing the work load among multiple file servers.

Place each project database, or collection of project databases, on its own file server. Use the normal procedure to set up and map projects to a file server, but consider size and usage patterns when deciding which project databases to place on a given file server so as to best balance the load across your servers.

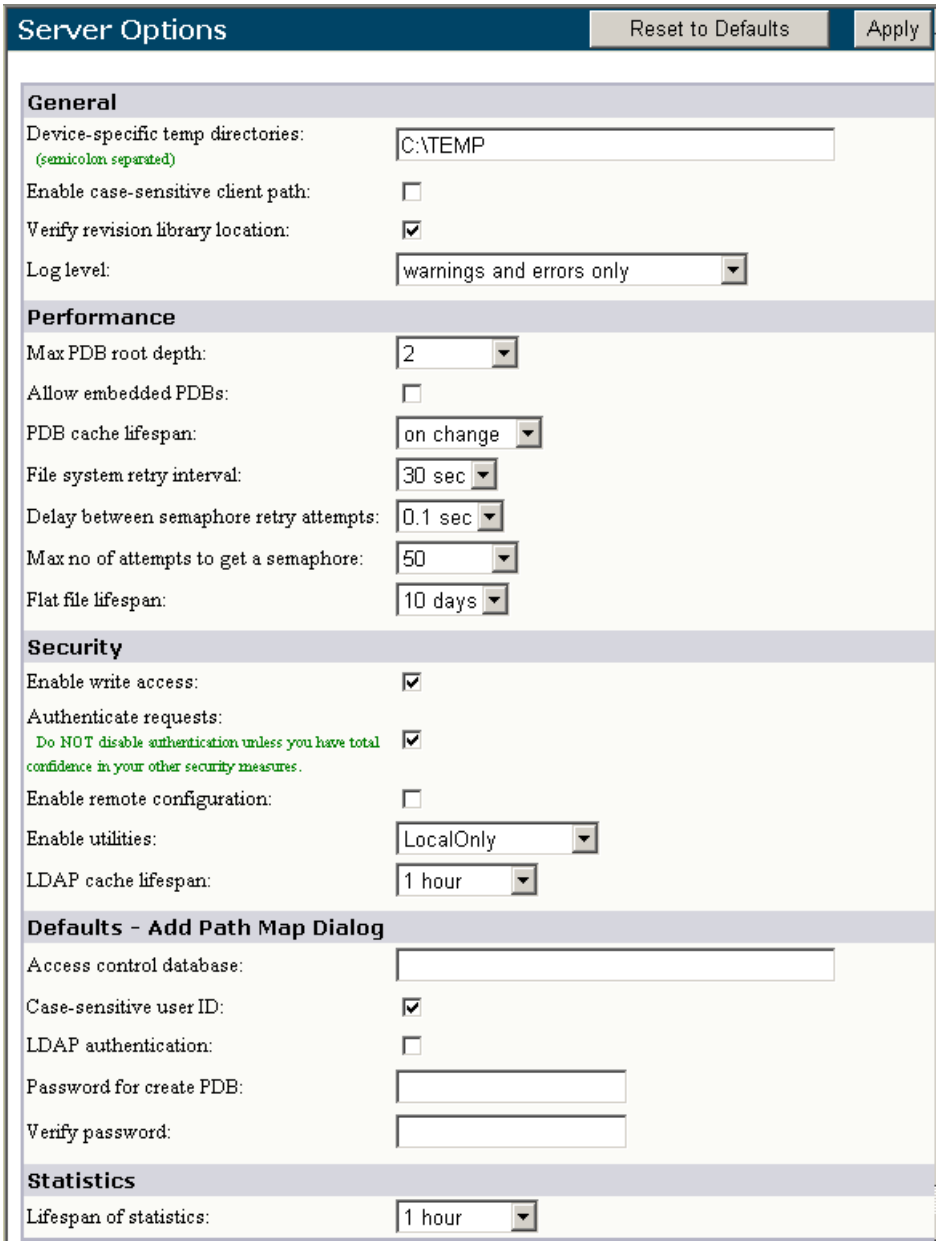
Configuring the File Server

You can enable, disable, and configure file server features to optimize performance and security based on your hardware, O/S, and usage.

To configure the file server:

1 Launch the Version Manager File Server Administration Utility. See "Launching the Version Manager File Server Administration Utility" on page 136.

2 Select **Options** in the left pane. The Server Options pane appears.



3 Modify any of the following:

General options

- **Device-specific temp directories:** A list of the temporary directories to use when transferring files. The first entry in the list is the default temporary directory. Subsequent entries specify the directories to use when accessing or writing to particular drives. For best performance, specify a temporary directory on each drive/file system that contains project databases or archives published through the Version Manager File Server.

  Separate multiple entries with semicolons (;), for example `C:\temp;D:\temp;T:\VMtemp`.

- **Enable case-sensitive client path:** Allows case sensitive client path names. By default, this option is disabled. In UNIX environments, enabling this option can

result in increased performance. Do NOT enable this option unless ALL of the following are true:

- The file server is on a UNIX system.
- ALL clients are on UNIX systems.
- ALL client names, as defined in the Version Manager File Server Administration Utility, are in the form of UNIX-style paths (no Windows-style or UNC-style client paths).

- **Verify revision library location:** Instructs clients to check metadata archives to see if they have been moved or renamed since the archives were split. By default, this is disabled.
- **Log Level:** Specifies which events are written to the log. The options are:
  - **warnings and errors only** (the default).
  - **greater detail**, which includes warnings, errors, and information messages. Every API call for which SUCCESS is not returned will generate an information message, including calls for which the caller did not expect SUCCESS to be returned. For example, the caller issues a PUT command for a directory to ensure the directory has been created. A response of a FsDuplicatePathException in that case is expected.

Performance options

- **Max PDB root depth:** The depth, measured in directories, to search published Project DB Paths for project databases. These project databases are then displayed in the client so users can select them. The options are **0**, **1**, **2** (the default), **3**, **4**, **5**, **6**, **7**, **8**, **9**, **10**, and **unlimited**.

  Select **0** only if the last component of the Project DB Path is itself the project database directory.

NOTE This feature is affected by the **Allow embedded PDBs** option.

- **Allow embedded PDBs:** By default, Version Manager searches published Project DB Paths to the depth specified by the **Max PDB root depth** option or until a project database is found, which ever comes first. If you wish to publish project databases that are nested below other project databases, enable this feature. The search will then continue past any project databases that are found until the depth specified by the **Max PDB root depth** option is reached.

Special Considerations

- Searching deeply through large directory structures may impair performance.
- We recommend that you do **NOT** nest project databases.
- You **cannot** create a project database **beneath** an existing project database.
- You can create a project database **above** an existing project database.

- **PDB cache lifespan:** Specifies how long the list of published project databases is cached before it expires. In addition to the time constraint, the cache expires if the path map is modified or the value of the **Max PDB root depth** or **Allow embedded PDBs** option is changed**.** When the cache expires, the next client request for the list of published project databases causes the server to generate an updated list.

The options are:

- **refresh now:** Immediately refreshes the cache and resets the value of this field to **on change**.
- **0 seconds:** No cache. The list of published project databases is regenerated by the server each time a client makes a request.
- **30 seconds:** Cache lasts for 30 seconds.
- **1 minute:** Cache lasts for 1 minute.
- **5 minutes:** Cache lasts for 5 minutes.
- **10 minutes:** Cache lasts for 10 minutes.
- **daily at *HH*:*MM*:** Cache is refreshed daily at the specified hour, between **00:00** and **23:00**.
- **on change:** Cache expires whenever a project database is created or deleted (the default).

- **File system retry interval:** The maximum time in seconds that the server will wait before failing when trying to recover from an I/O error. The options are **0 sec**, **1 sec**, **5 sec**, **10 sec**, and **30 sec** (the default).
- **Delay between semaphore retry attempts:** The time in seconds between attempts to get a semaphore. The options are **0.1 sec** (the default), **0.2 sec**, **0.5 sec**, **1 sec**, **5 sec**, and **10 sec**.
- **Max no of attempts to get a semaphore:** The maximum number of attempts to get a semaphore before returning an error. The options are **1**, **3**, **5**, **10**, **20**, **50** (the default), **100**, **200**, and **unlimited**.
- **Flat file lifespan:** The number of days to retain unaccessed revisions as flat files if they are also in the Delta store. The options are **0 days**, **1 day**, **5 days**, **10 days** (the default), and **30 days**.

Security options

- **Enable write access:** Allows users to perform operations that write to the revision and metadata archives. Use this control to disable write access during backup operations. By default, write access is enabled.
- **Authenticate requests:** Authenticates each request to the Version Manager File Server to verify that the request is from a valid instance of a Version Manager client. This is done to ensure the security of your data. Without such protection, a malicious entity could gain access to, and/or damage, your data. There is some overhead associated with this security feature, but we strongly recommend that you leave it in place. Do NOT disable authentication unless you have total confidence in your other security measures.
- **Enable remote configuration:** Allows users to access the Version Manager File Server Administration Utility from other computers. By default, this is disabled.
- **Enable Utilities:** Controls access to the VSPLIT, VTRANSFER, ReadDB, and MakeDB utilities. The options are:
  - **True:** Enables all utilities.
  - **False:** Disables all utilities.
  - **ReadOnly:** Disables VSPLIT and MakeDB, enables ReadDB, and enables VTRANSFER for export only.

- **LocalOnly:** Enables all utilities for users who are local to the file server (the default).
- **RemoteReadOnly:** Enables all utilities for users who are local to the file server and, for remote users, enables ReadDB and enables VTRANSFER for export only.

- **LDAP cache lifespan:** Specifies how long the LDAP user ID and password are cached before they expire. The options are **1 minute**, **5 minutes**, **30 minutes**, and **60 minutes** (the default).

Defaults - Add Path Map dialog box

- **Access control database:** Populates the field in the Add Path Map dialog box with a default access control database to associate with client paths.
- **Case-sensitive user ID:** Enables the option in the Add Path Map dialog box by default. That option specifies whether the access control database expects case-sensitive user IDs.
- **LDAP authentication:** Enables the option in the Add Path Map dialog box by default. That option specifies whether LDAP authentication is enabled for the path map.
- **Password for Create PDB:** Populates the field in the Add Path Map dialog box with a default password for create-project-database operations.
- **Verify password:** Populates the field in the Add Path Map dialog box with a default password to verify the entry made above in the **Password for Create PDB** field.

Statistics options

- **Lifespan of statistics:** The length of time data is persisted in memory for viewing in the Operation Times table of the Status pane. The options are **10 minutes**, **1 hour** (the default), **1 day**, and **2 days**.

  The status of all completed operations will be purged if the data exceeds the memory allocated for the status log.

4. To restore the default settings, click the **Reset to Defaults** button.
5. To apply your changes, click the **Apply** button.

Setting WorkDir & ArchiveWork Directives

Do not set the WorkDir or ArchiveWork directives to a directory that is mapped to the file server--unless that location can be accessed via an existing O/S drive mapping. The file server itself cannot be used to access the directories. For more information on setting these directives, see "Temporary File Options" on page 95.

Viewing Server Status

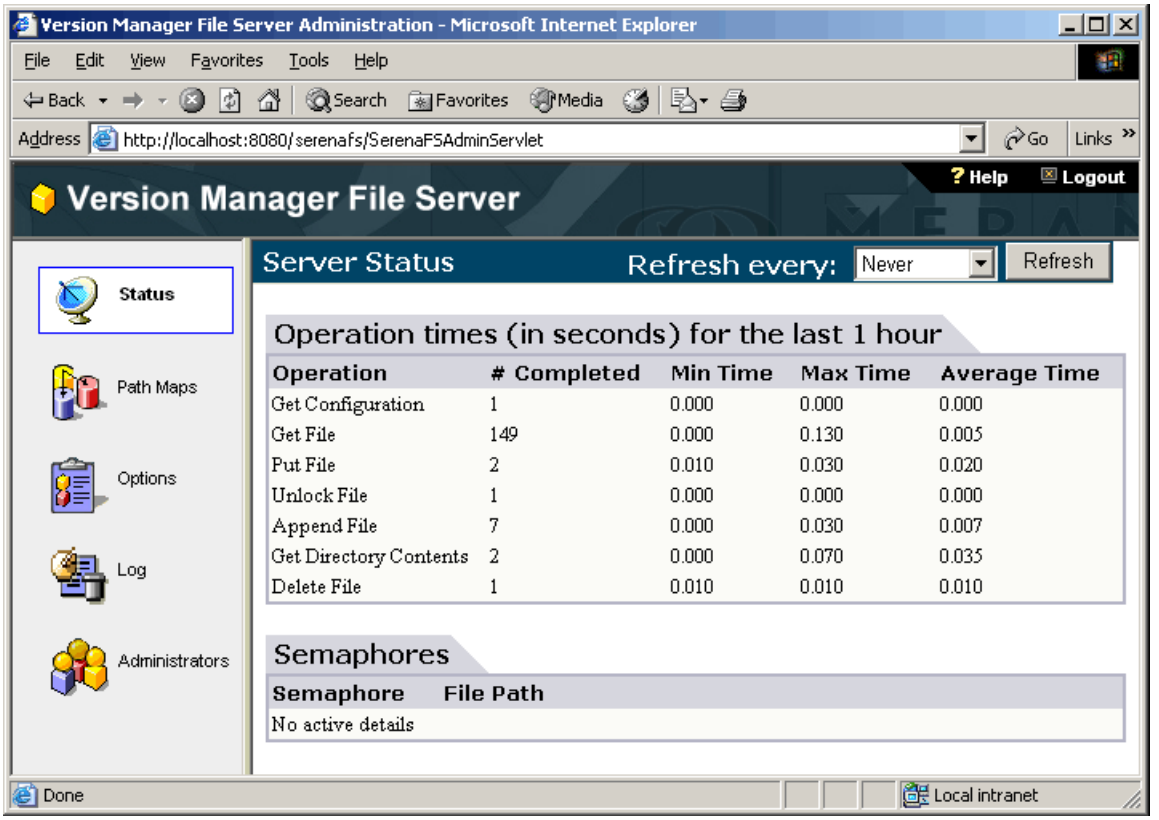
From the Version Manager File Server Administration Utility, you can view two types of file server status information: a record of operations performed over a defined time period, and a list of active semaphores by file path. From the desktop client, you can view the connection status to file servers.

Viewing Server Status from the Version Manager File Server Administration Utility

To view server status:

1 Launch the Version Manager File Server Administration Utility.

2 Select **Status** in the left pane. The Server Status pane appears.



3 The following information is displayed:

- The **Operation times** table displays the following information for the time period indicated in its title:
  - **Operation:** The names of operations that ran.
  - **# Completed:** The number of times a given operation ran.
  - **Min Time:** The minimum time an instance of a given operation ran.
  - **Max Time:** The maximum time an instance of a given operation ran.
  - **Average Time:** The average time all instances of a given operation took to run.

  For information on setting the lifespan of statistics in the Operation Times table, see "Configuring the File Server" on page 141.
- The **Semaphores** table displays the file paths of the active semaphores.

4 To specify how often the Server Status pane is refreshed, select one of the following options from the **Refresh every** field: **Never** (the default), **1 minute**, **5 minutes**, **10 minutes**, **15 minutes**, **30 minutes**.

5 To refresh the Server Status pane, click the **Refresh** button.

6 To release a stuck semaphore, select the radio button next to the appropriate **File Path** entry and click the Delete ( ) button. To release all semaphores, select the radio button next to the **All semaphores** entry and click the Delete button.

Viewing Server Status from the Desktop Client

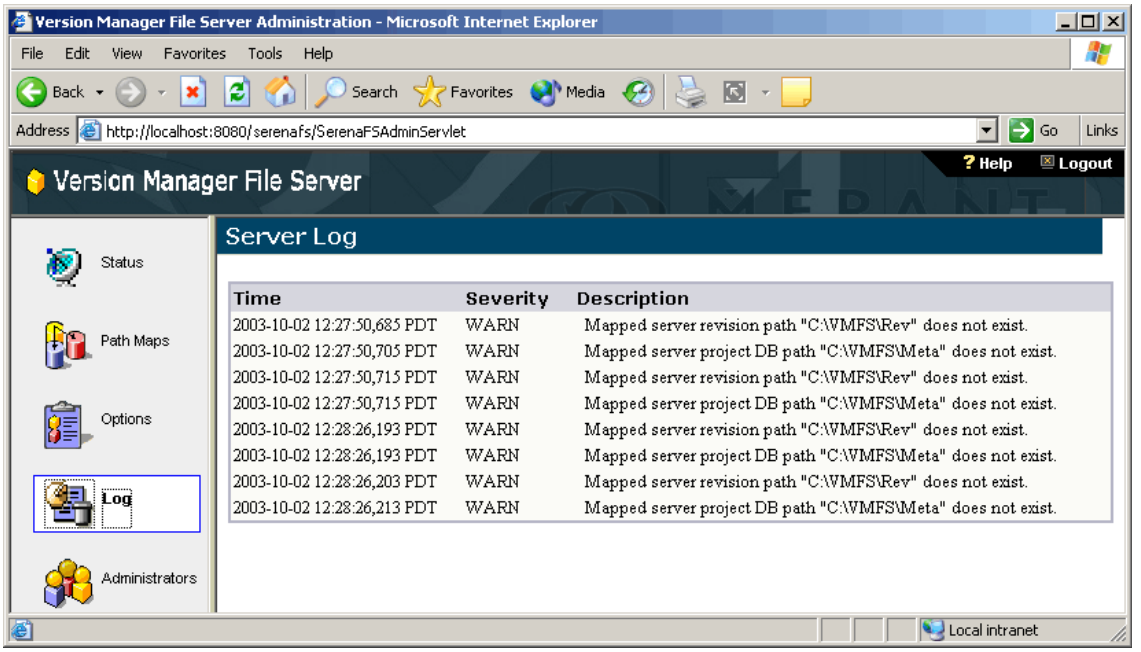
In the Version Manager desktop client, server status is displayed in the left corner of the status bar (click to refresh status and see more information):

- No file server is configured.
- One or more configured file servers are not responding.
- All configured file servers are working correctly.

Server status is updated whenever the client attempts to access files on the server.

Viewing the Server Log

From the Log pane of the Version Manager File Server Administration Utility, you can view a log of server events, such as errors and warnings.



The following information is displayed for each event:

- **Time:** The date and time the event occurred. The time is based on the time zone of the server.
- **Severity:** Denotes the nature of the event, such as error, warning, information, etc.
- **Description:** Describes the event and notes the entities involved.

The information displayed in the Log pane is stored in the `pvcsfs.log` file which is located at:

InstallLocation`\Serena\vm\common\tomcat\logs\`

Once the log file reaches 100KB in size, it is renamed and a new `pvcsfs.log` file is started. Up to ten log files are retained by the server, but only the current log file is displayed in the Log pane. To view old log files, use a text editor.

Adding Project Databases to a Version Manager File Server

The optimum method of adding project databases to the Version Manager File Server depends on your answers to the following questions:

#	Do you want to...	Yes	No
1	Add your existing project databases to a Version Manager File Server?	Go to Question 2.	See "Creating New Project Databases on a File Server" on page 157.
2	Keep your project databases where they are and retain the drive mappings or UNC paths that reference them?	See "Adding Existing Project Databases to a File Server Without Moving Them" on page 148.	Go to Question 3.
3	Use the Desktop Client to copy project databases to a file server?	See "Using the Desktop Client to Copy Project Databases to a File Server" on page 152.	See "Using PCLI to Copy Project Databases to a File Server" on page 154.

Shared archives

For help in determining which archives are shared, see Serena KnowledgeBase article 14160.

Adding Existing Project Databases to a File Server Without Moving Them

This is the easiest way to add existing project databases to a file server. All information in your existing project databases will be retained, including workspace definitions and references to shared archives.

Special Considerations

The transition to a Version Manager File Server is a convenient opportunity to move your project databases to a new location and/or to switch existing drive mappings to UNC paths. Do not use this procedure if you wish to make such changes while moving to a file server. For other options, see "Adding Project Databases to a Version Manager File Server" on page 148.

To add existing project databases to a file server:

1 Backup the project databases you intend to add to a file server.

2 Install the Version Manager File Server to the system that contains the project databases.

NOTE Upgrade the Version Manager clients for all users of the project database to Version Manager 8 or later.

3 Use the Version Manager File Server Administration Utility to create a path map for the project databases.

Set the **Client Name** to match the existing path to the project database. For example, if your existing project databases are located in a directory on a mapped network drive, `S:\VMPDBS`, you would set the **Client Name** to `S:\VMPDBS`. See "Managing Project Database and Revision Library Paths" on page 137.

4 Connect the Version Manager desktop client to the file server. See "Configuring Clients for Use with File Servers" on page 167.

5 (Optional, but recommended) Remove or restrict the network mapping through which the project databases were originally accessed.

See the following example for more details.

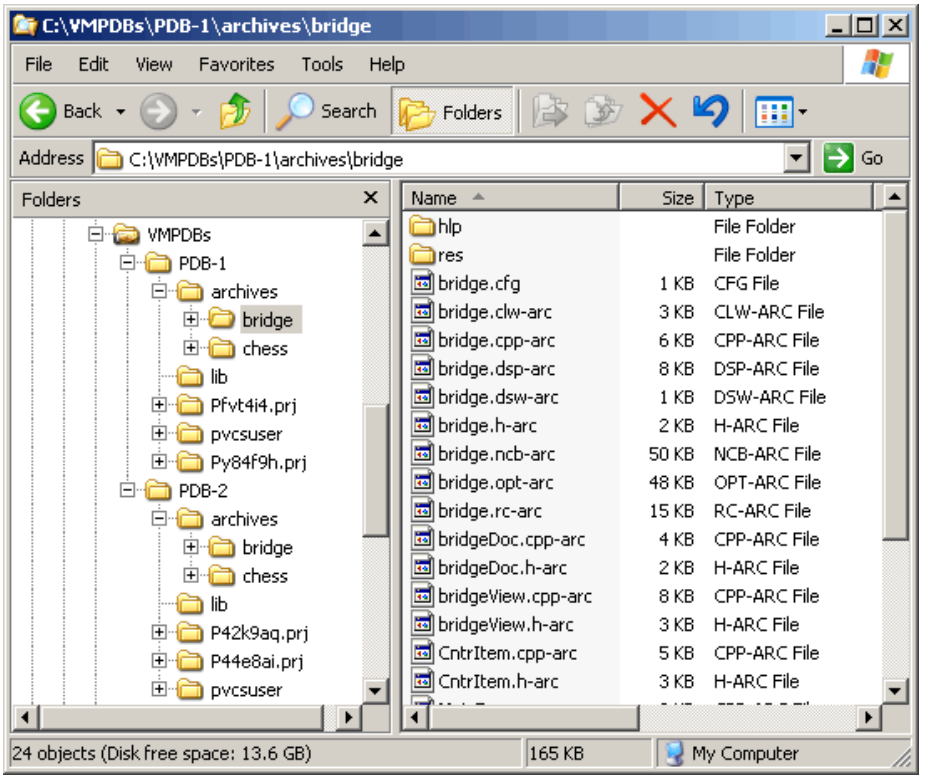
Example: Adding Existing Project Databases without Moving Them

This example shows a hypothetical case where the existing project databases, PDB-1 and PDB-2, are located on the C: drive of a system that is accessed via a network mapping of S:.

Initial state

The following image shows the initial location of the project database files before anything is done to add them to a Version Manager File Server. Actually, nothing about this

structure will change unless you also create a revision library by splitting the existing archives.



Users currently access these project databases over the network as follows:

- S:\VMPDBS\PDB-1
- S:\VMPDBS\PDB-2

Where S: is an O/S drive mapping of the C: drive on the system hosting the project databases.

Backup data

1 Backup the project databases you intend to add to a file server.

Install the file server

2 Install the Version Manager File Server and desktop client to the system that contains the project databases. For information on installing Version Manager, see the *Version Manager Installation Guide*.

Start the file server

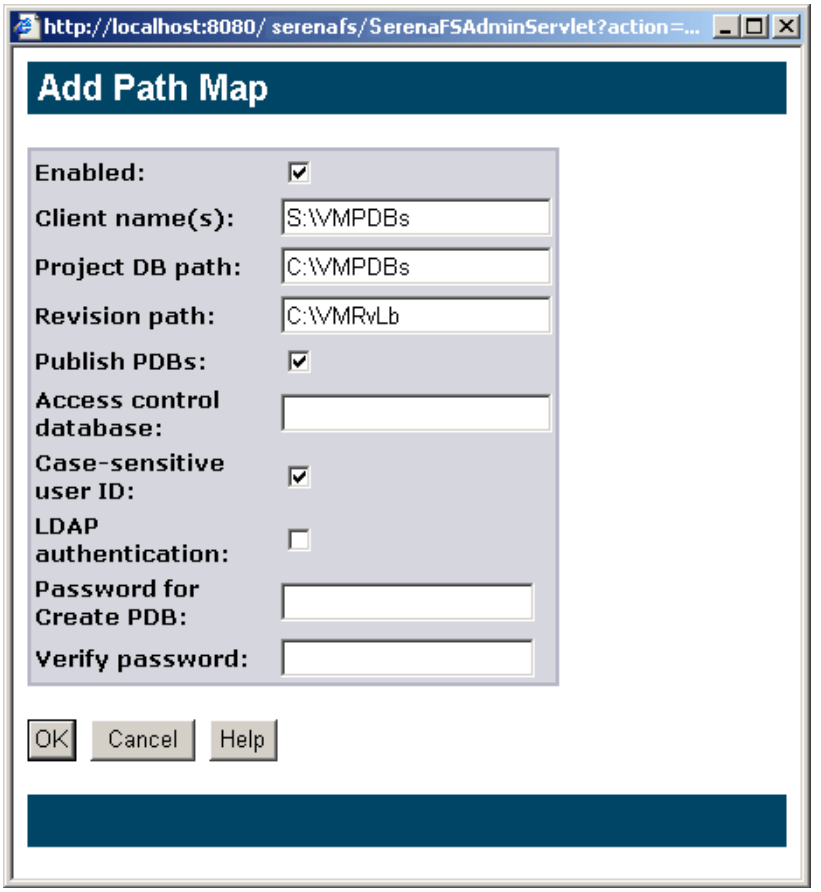
3 Select Serena | Version Manager | Version Manager Application Server from the Start menu on the system to which you installed the file sever. The Version Manager Application Server Admin appears.

4 Click the **Start** button on the Servers tab.

Map project database paths

5 Launch the Version Manager File Server Administration Utility (Serena Version Manager | Serena Version Manager File Server Admin) and click the **Path Maps** button. The Path Maps pane appears.

6 Click the Add ( ) button. The Add Path Map dialog box appears.



a Select **True** from the **Enabled** list to activate the path map.

b In the **Client Name** field, enter the network path to a directory that is above your existing project databases. For this example, that is `S:\VMPDBS`.

NOTE If a path must be accessed by both UNIX and Windows clients, you can enter a client name in the correct form for each O/S, using a semicolon to separate them, or you can access the path via a Windows style path from both Windows and UNIX systems. For more information on working in a cross-platform environment, see "Using the File Server in a Cross-Platform Environment" on page 164.

c In the **Project DB Path** field, enter the O/S native path to a directory that is above your existing project databases. For this example, that is `C:\VMPDBS`.

d In the **Revision Path** field, enter the O/S native path to a directory under which you want to create a Revision Library.

e Select **True** from the **Publish PDB** list so users can select project databases from a list displayed in the client.

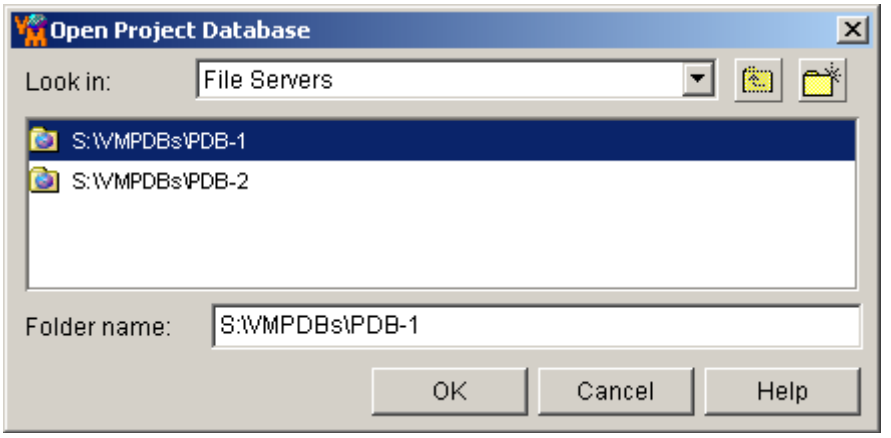
f Click **OK**.

Configure clients

7 Configure each client workstation to connect to the file server. See "Configuring Clients for Use with File Servers" on page 167.

Open project database from client

8 From the desktop client, select File | Open Project Database. The Open Project Database dialog box appears.



9 Select **File Servers** from the **Look in** list--it is the top item in the list. The dialog box is now populated with a list of all the project databases that are located at or below the location you mapped in the Path Maps pane.

10 Select a project database and click **OK**. For more information on accessing the file server from Version Manager clients, see "How to Access File Servers from Version Manager Clients" on page 171.

11 You can now remove the mapped network drive, S: in the example, that was originally used to access the project databases--unless users still need it to access other project databases or workfiles. You can also use O/S permissions to limit access to the project database directories since users will now access the project databases only via the Version Manager File Server.

This example is used as the starting point for "Example: Splitting Archives" on page 160.

Using the Desktop Client to Copy Project Databases to a File Server

This is the easiest way to add existing project databases to a file server on a different system and/or using a different path.

NOTE You can now retain shared archives and workspaces when using the Desktop client to copy project databases. Previously only PCLI could do this.

Prerequisites

Before copying project databases to a file server, you must do the following:

1 Install the Version Manager File Server and desktop client to the system that will host the file server. For information on installing Version Manager, see the *Version Manager Installation Guide*.

NOTE Upgrade the Version Manager clients for all users of the project database to Version Manager 8 or later.

2 Start the Version Manager File Server. See "Starting and Stopping the File Server" on page 134.

3 Use the Version Manager File Server Administration Utility to create a path map for the project databases. See "Managing Project Database and Revision Library Paths" on page 137.

4 Connect the Version Manager desktop client to the file server. See "Configuring Clients for Use with File Servers" on page 167.

Procedure **To copy existing project databases to a file server:**

1 Backup the project databases you intend to copy.

2 Launch the Version Manager desktop client and select Edit | Copy. The Copy Project Database wizard appears.

IMPORTANT! The client must be Version Manager 8 or later.

3 Enter a name for the copied project database in the **Name** field. This is the name that will appear in Version Manager clients.

4 Click the browse button next to the **Location** field. The Select Location for Project Database dialog box appears.

5 Select **File Servers** from the **Look in** list. The dialog box now lists all Client Names that are defined and marked for publication. For information on creating client names, see "Managing Project Database and Revision Library Paths" on page 137.

6 Select the Client Name (path map) to which you want to add the project database.

7 Unless this Client Name is to serve as the root of a single project database, you must create a sub directory for the project database. Otherwise you will be unable to successfully add other project databases to this Client Name.

To create a new subdirectory for the project database, click the New Folder ( ) button and enter a name for the new directory in the resulting dialog box.

8 Click **OK**. The Copy Project Database dialog box reappears with the **Location** and **Archive Location** fields populated.

9 Specify a **Workfile Location** for the project database.

IMPORTANT! The workfile location must NOT be a location mapped to a Version Manager File Server unless it is also available to clients via a network drive mapping or UNC path.

10 Click the **Next** button. The second page of the Copy Project Database wizard appears.

11 Select the **Copy archives to project location** option.

12 To retain associations with shared archives, select the **Retain shared archives** check box.

NOTE Archives are shared, not projects. If you eventually add a new file to the project, it will not be shared with any other project until you copy the versioned file to the desired project or projects.

13 Select the **Include subprojects** check box.

14 Click the **Next** button. The third page of the Copy Project Database wizard appears.

15 Select the **Copy existing Configuration Files** option.

16 Select the **Copy Access Control Database to new location** option.

17 To retain the current workspace definitions, select the **Copy Workspaces** check box.

18 Click the **Finish** button. The progress of the copy process is displayed then the results. Click the **Details** button to examine any errors.

The project database is now on the file server. To create a Revision Library for the project database, see "Creating Revision Libraries" on page 157.

Using PCLI to Copy Project Databases to a File Server

This is the most thorough way to add existing project databases to a file server on a different system and/or using a different path. It provides the ability to deal with more complex issues.

NOTE You can now retain shared archives and workspaces when using the Desktop client to copy project databases. Previously only PCLI could do this.

See "Using the Desktop Client to Copy Project Databases to a File Server" on page 152.

Special Considerations

This procedure is more complicated than the other options. It requires the following steps:

- Use the PCLI ExportPDB command to export project database information to a text file.
- Edit the paths in the text file to reflect the new project database location on the Version Manager File Server.
- Copy the project database files to a directory structure on the file server.
- Edit the paths in all configuration (.CFG) files to reflect the new project database location on the Version Manager File Server.
- Use the PCLI ImportPDB command to import the paths in the modified text file into the project database.

Please review the entire procedure before beginning. For more information on the commands used in this procedure, see the *Version Manager PCLI User's Guide and Reference* and Serena KnowledgeBase article 39987.

For other options, see "Adding Project Databases to a Version Manager File Server" on page 148.

Prerequisites

Before copying project databases to a file server, you must do the following:

1. Install the Version Manager File Server and desktop client to the system that will host the file server. For information on installing Version Manager, see the *Version Manager Installation Guide*.

NOTE Upgrade the Version Manager clients for all users of the project database to Version Manager 8 or later.

2. Start the Version Manager File Server. See "Starting and Stopping the File Server" on page 134.
3. Use the Version Manager File Server Administration Utility to create a path map for the project databases. See "Managing Project Database and Revision Library Paths" on page 137.
4. Connect the Version Manager desktop client to the file server. See "Configuring Clients for Use with File Servers" on page 167.

Procedure

To copy existing project databases to a file server:

1. Backup the project databases you intend to copy.
2. Consider how your embedded configuration options, if any, may affect the export/import operations. Unembed options as necessary before proceeding.
3. From a command prompt, enter the following command:

   pcli exportpdb -pr*PathToPDB* **-id***UserID*:*PW* **-z -f***PathToExportFile*

   Where:

   - *PathToPDB* is the existing path to the root of the project database.
   - *UserID* is a valid Version Manager user ID with the SuperUser privilege and *PW* is the associated password, if any. This information is necessary only if an access control database is in effect.
   - *PathToExportFile* is the path to, and name of, the file that will contain the information exported from your existing project database.
4. Open the resulting file in a text editor and replace the existing paths with the appropriate file-server paths relative to the Client Name you mapped in the Version Manager File Server Administration Utility.

IMPORTANT! Do NOT set `WorkPath` to a location that is mapped to the Version Manager File Server, unless it is also available to clients via a network drive mapping or UNC path.

5. Use a tool or O/S command to copy all of the files whose path references you changed in the previous step. Make sure that you copy the files to the location on the file server that matches the Client Name for which you modified the path references.

   Files to copy normally include:

   - All configuration files (`.cfg`).
   - All access control database files (`.db`).

- And, most likely, all archives (`-arc`).

NOTE The file extensions listed above are the defaults. However, you may have used different file extensions.

Files and directories **NOT** to copy:

- Any `.ser` file (`pvcsroot.ser`, `pvcsid.ser`, etc.).
- Any `.old` files (`pvcsroot.old`, `pvcs.old`, etc.).
- The `pvcsuser` subdirectory.
- The `lib` subdirectory, unless you explicitly placed files there.
- Any subdirectory that matches the P*.prj wildcard. For example, `Pnoprw.prj`.

6 Open all configuration (.CFG) files referenced in the export file and replace the existing paths with the appropriate file-server paths relative to the Client Name you mapped in the Version Manager File Server Administration Utility. Repeat this step for any configuration files referenced by these configuration files, and so on.

IMPORTANT! Do NOT set `WorkPath` to a location that is mapped to a Version Manager File Server, unless it is also available to clients via a network drive mapping or UNC path.

7 (optional) To view past journal entries via a Version Manager client, you must update the paths in the journal files just as you did for the .CFG files. However, this step is not necessary in order for new entries to be viewable in the clients, and you can still view the old entries with a text editor.

8 From a command prompt, enter the following command:

pcli importpdb -pr*PathToNewPDB* **-id***UserID*:*PW* **-f***PathToExportFile*

Where:

- *PathToNewPDB* is the path to the root of the new project database location on the file server. You must specify this path relative to the Client Name.
- *UserID* is a valid Version Manager user ID with the SuperUser privilege and *PW* is the associated password, if any. This information is necessary only if an access control database is in effect.
- *PathToExportFile* is the path to, and name of, the file that contains the information exported from your existing project database.

NOTE If a password is required to create project databases, you must specify the **-pa***password* option.

9 Re-embed any configuration options you unembedded above, as necessary.

Creating New Project Databases on a File Server

To create a new project database on the file server:

1. Launch the Version Manager desktop client and select Admin | Create Project Database. The Create Project Database dialog box appears.
2. Enter a name for the new project database in the **Name** field. This is the name that will appear in Version Manager clients.
3. Click the browse button next to the **Location** field. The Select Location for Project Database dialog box appears.
4. Select **File Servers** from the **Look in** list. The dialog box lists all Client Names that are enabled. For information on creating client names, see "Managing Project Database and Revision Library Paths" on page 137.
5. Select the Client Name that represents the server location (path map) to which you are adding the project database.
6. Unless this Client Name is to serve as the root of a single project database, you must create a sub directory for the project database. Otherwise you will be unable to successfully add other project databases to this Client Name.

   To create a new subdirectory for the project database, click the New Folder () button and enter a name for the new directory in the resulting dialog box.
7. Click **OK**. The Create Project Database dialog box reappears with the **Location** and **Archive Location** fields populated.
8. Complete the remaining fields as you normally would when creating a new project database. For detailed information on the remaining fields, see "Creating a Project Database" on page 22.

   The project database is now on the file server. To create a Revision Library for the project database, see "Creating Revision Libraries" on page 157.

Creating Revision Libraries

Inflated archives

In the past, Version Manager kept revision data and metadata together in the same archives. We now refer to these as *inflated archives* or *unsplit archives*.

Split archives

The Revision Library feature of the Version Manager File Server stores the revision data separately from the metadata, which increases the speed of many operations. Archives used in this way are said to be *split archives* since the revisions are stored in a Revision Library and the metadata is stored in a separate metadata archive.

Split archives are accessible only through the Version Manager File Server using Version Manager 8 or later.

You can *unsplit* archives to restore them to the inflated archive format. These restored inflated archives are backward compatible with earlier versions of Version Manager.

RFSSplitOnCreate directive

The RFSSplitOnCreate configuration file directive causes new archives to be split as they are created. When you create a revision library, you must set this directive if the project

database was added to a file server from existing inflated archives. By default, it is enabled in newly created project databases.

VSPLIT command — The VSPLIT command splits inflated archives into separate revision data (revision library) and metadata.

Prerequisites

Before creating a revision library, you must:

- Use the Version Manager File Server Administration Utility to create a Revision Path map for the project database. See "Managing Project Database and Revision Library Paths" on page 137.
- Add the project database to the file server. See "Adding Project Databases to a Version Manager File Server" on page 148.

Enabling the RFSSplitOnCreate Directive

The RFSSplitOnCreate directive causes new archives to be split as they are created resulting in separate revision data (Revision Library) and metadata. Though this has no effect on existing archives, it will apply to new archives as they are added to the project database.

New archives will be split only if the project database is mapped to a revision path.

To enable the RFSSplitOnCreate directive:

1. Launch the Version Manager desktop client and select a project database.
2. Select Admin | Configure Project. The Configure Project Database dialog box appears with the General tab active.
3. Select **Creation Attributes** from under **Archives** in the Options pane.
4. Select the **Split On Create** check box.
5. Click **OK**.

NOTE For information on enabling/disabling the RFSSplitOnCreate directive from the command-line, see the *Version Manager Command-Line Reference Guide*.

Splitting Existing Archives

VSPLIT is a command-line utility that splits existing inflated archives into separate metadata and revision stores for use with the Revision Library feature of the Version Manager File Server. VSPLIT can also unsplit archives in revision libraries to return them to the original inflated archive format.

Syntax

`vsplit [Options] Path . . .`

Where *Path* is the Client Name that represents the project database as defined in the Path Maps pane of the Version Manager File Server Administration Utility.

Options

@ `@[list_file]`

Use the `@` option to read *list_file* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the `@` option with no parameter to redirect input from another command.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-h Displays help for the command. The command terminates after processing the **-h** option even if you specify other options.

-l Lists the archives but does not split them.

-r Recursive (applies to directory paths).

-u Unsplit recombines revision and metadata from split archives to return them to the original inflated archive format. The reconstituted archive is located in the metadata location. The Revision Library is deleted.

TIP Disable the RFSSplitOnCreate directive or any new files you add will end up in split archives.

-v Report version of this tool.

-xe `-xe`*file_name*

Redirects status, program, and error messages to *file_name*.

-xo `-xo`*file_name*

Redirects standard output to *file_name*.

Special Considerations

- VSPLIT needs to reference the archive path as viewed by the client (as defined by the Client Name in the path map) even if it is run from the system running the file server.
- VSPLIT affects only existing archives. It does not affect the creation of new archives. To split archives when they are created, set the RFSSplitOnCreate directive.
- Other project database files are not used or affected by VSPLIT.
- You can run VSPLIT only if it is enabled by the **Enable Utilities** option. See "Configuring the File Server" on page 141.
- VSPLIT is located in the `vm\`*os*`\bin\admin` directory of your Version Manager installation.

VSPLIT: Usage Suggestions

We recommend that you do the following:

- **Disable User Access:** If the archives to be split are already available to users, disable or limit user access to the archives before running VSPLIT. Once the operation is complete and you have verified the results, you can re-enable user access. If you will run VSPLIT from the file server itself and the archives are local to the server, you can disable user access simply by disconnecting the server from the network.

- **Test:** Set up a test environment that matches your actual production environment as closely as possible. Use this test environment to test the upgrading process against a copy of your project databases.
- **Back Up:** Back up your data before running VSPLIT.
- **Redirect Output:** Redirect the output of the VSPLIT command to a file so you can analyze the output to identify any archives that were not properly split. To redirect the output to a file, use the **-xo** and **-xe** options, which can be combined as **-xo+e**. For example:
  vsplit -xo+e OutPut.txt **-r** S:\VMFS\PDBS

  For information about analyzing this output, see "Analyzing VSPLIT Output" on page 161.

Example: Splitting Archives

This example shows a hypothetical case where the existing project databases, PDB-1 and PDB-2, were located on a system accessed via a network mapping of S:. These project databases were added to the Version Manager File Server as the client name path map S:\VMPDBS.

This example picks up where "Example: Adding Existing Project Databases without Moving Them" on page 149 left off. See that example for the steps that lead up to this point and for more information about the initial state of the project databases.

Users currently access these project databases via the S:\VMPDBS path mapping on the Version Manager File Server as:

- S:\VMPDBS\PDB-1
- S:\VMPDBS\PDB-2

1. Enable the RFSSplitOnCreate directive for the project databases for which you are creating revision libraries. See "Enabling the RFSSplitOnCreate Directive" on page 158.
2. You can run the VSPLIT command only if it is enabled by the **Enable Utilities** option. See "Configuring the File Server" on page 141.
3. Run the following command from the command line of a Version Manager client system:
   vsplit -xo+e OutPut.txt **-r** S:\VMPDBS

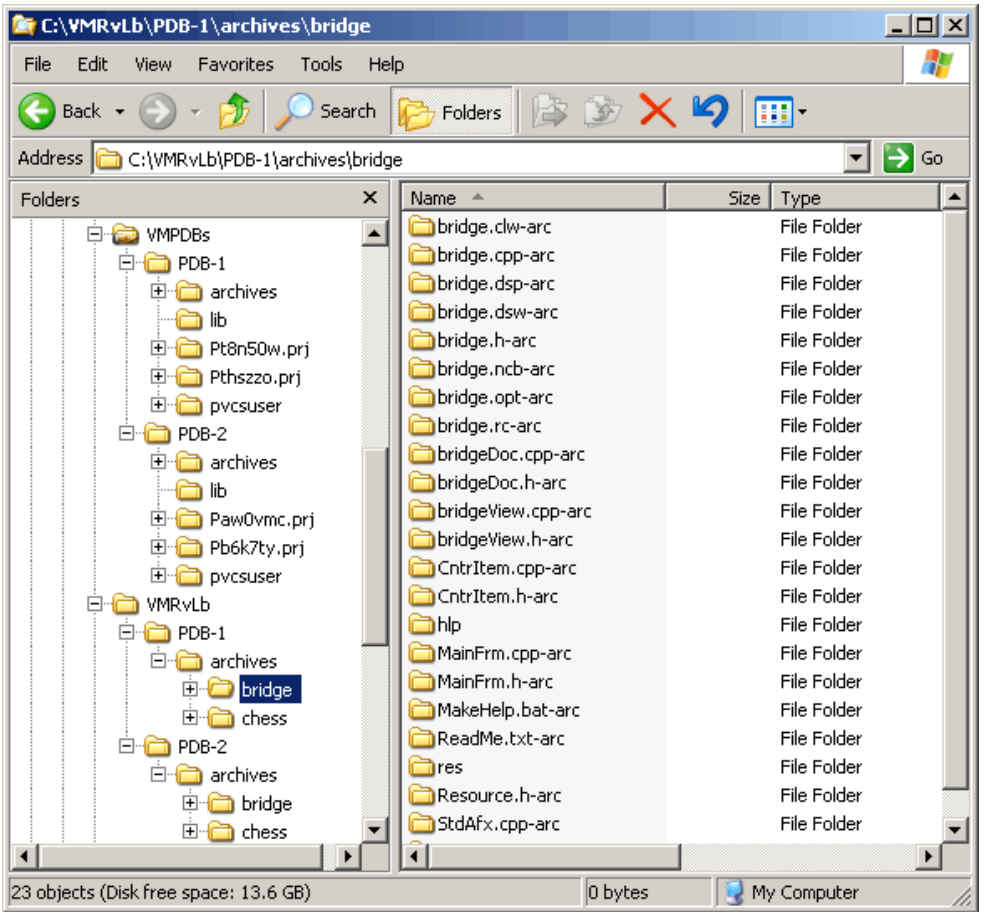
   The VSPLIT command is located in the vm*OS*\bin\admin directory of your Version Manager installation.

   The **-xo+e** option allows you to analyze the results of the operation. See "Analyzing VSPLIT Output" on page 161.

   For information on connecting clients to a file server, see "Configuring Clients for Use with File Servers" on page 167.

The following image shows the location of the project database files after running VSPLIT. The metadata archives are in the original archive location under C:\VMPDBs, while the revisions are under C:\VMRvLb. For information on how these locations were mapped in

the file server, see "Example: Adding Existing Project Databases without Moving Them" on page 149.



Analyzing VSPLIT Output

VSPLIT checks each archive before and after it is split. It will not split corrupted archives or any non-archive files it encounters.

Various non-archive files (access control databases, journal files, and configuration files) will be listed as conversion failures—this is expected and is of no concern. You should examine the output for any archives that failed.

An entry is made for each file. The following example shows an archive that was successfully split and a journal file which failed—as expected—since it is not an archive and cannot be split.

```
convert(S:\VMFS\PDBS\archives\test\env_filt.sed-arc) ok 733/1425 bytes (new/old)
vcheck(S:\VMFS\PDBS\archives\test\journal.vcs)=1 failed, not converting
```

If unexpected failures appear in the VSPLIT output, examine the `pvcsfs.log` file. Errors recorded in this file may help you resolve the underlying problem. For more information, see "Viewing the Server Log" on page 147.

Exporting, Importing, Moving, Renaming, and Fixing Archives

VTRANSFER is a command-line utility for use with the Version Manager File Server that allows you to:

- Export the metadata and revision data of an archive, including split archives, as a single *.zip file. This is a convenient way to gather and package an archive so it can be sent to a Serena support representative for troubleshooting.
- Import the metadata and revision data of an archive, including split archives, from a single *.zip file. This is a convenient way to import changes made to troubleshoot an archive.
- Move an archive from one location to another on the same server.
- Rename an archive.
- Fix the relationship between the metadata and revision data (revision library) if the archive has been manually moved or renamed.
- Delete an archive, both the metadata and revision data.

Syntax

`vtransfer` [*options*] *archivePath* [*secondaryPath*]

Where:

- *archivePath* is the location of the archive you are operating on.
- *secondaryPath* is a parameter specific to the mode in which VTRANSFER is being used, such as the location of the *.zip file or a new name for the archive.

Options

VTRANSFER has six *mode options*, which determine what sort of operation it is to perform, as well as ten general options.

Mode Options

-c Copies an archive or a directory of archives to the name/location specified in the *secondaryPath*. This operation is recursive if the *archivePath* specifies a directory rather than an archive.

-d Deletes the archive or a directory of archives specified in the *archivePath*, including all revisions and metadata.

-f Fixes the relationship between the metadata and revision data by moving the revisions to a location that mirrors the metadata location, and then updates the metadata. This is useful if the metadata has been manually moved or renamed.

-i Imports an archive and its revisions from a zip file specified in the *secondaryPath* to the archive location specified in the *archivePath*. Any existing archive data is overwritten.

-r Renames or moves an archive or a directory of archives to the name/location specified in the *secondaryPath*. This operation is recursive if the *archivePath* specifies a directory rather than an archive.

-x Exports the metadata and revision data of the archive specified in the *archivePath* and writes it to a single *.zip file as specified in the *secondaryPath*. If the *secondaryPath* does not contain a fully qualified path, the file is written to the location from which you ran the command. The revisions are exported from the path implied by the revision path mapped on the file server and the *archivePath*.

Use with the **-m** option to export revisions from the location specified in the metadata. This is useful if the metadata has been manually moved.

General Options

@ **@**[*list_file*]

Use the **@** option to read *list_file* for additional command-line options. Version Manager reads additional options from the file before reading the rest of the command line. Use the **@** option with no parameter to redirect input from another command.

If the file has the extension .GRP, you can omit the extension and the path. Version Manager searches for it in the directories specified by the VCSDir directive.

-h Displays help for the command. The command terminates after processing the **-h** option even if you specify other options.

-id **-id***user_id*:*user_password*

Specifies a user ID and password to authenticate against the access control database and/or LDAP server, if any, that is associated with the relevant file server path map in the Version Manager File Server Administration utility.

NOTE The -id option does not affect the user ID used to perform the CLI operation itself. The user ID for the CLI operation is determined in the normal manner.

-m Use with **-x** or **-f** to export revisions from the location specified in the metadata. This is useful if the metadata has been manually moved

-n Use -n to force 'No' response to all queries. Default is to prompt.

-q Quiet. Displays minimal text output.

-xe -xe*file_name*

Redirects status, program, and error messages to *file_name.*

-xo -xo*file_name*

Redirects standard output to *file_name*.

-y Use -y to force 'Yes response to all queries. Default is to prompt.

-z Copies or moves a directory and all sub-directories. Requires the **-c**, **-d**, or **-r** option.

Examples

In the examples below, the archive is mapped to a Version Manager File Server using the client name S:\VMFS\PDBs.

Example 1: Export an Archive to a Zip File

This example takes the revisions and metadata associated with the `Hello.txt` versioned file and places it in a file named `Hello.zip`. This provides a convenient way to gather and package the various files that can represent a versioned file.

```
vtransfer -x S:\VMFS\PDBs\PDB-1\archives\Project-1\Hello.txt-arc Hello.zip
```

Example-2: Export an Archive to a Zip File Using the Revision Path Stored in the Metadata

This example is the same as Example 1, except that the revisions are obtained from the location indicated in the metadata (**-m**) rather than from the path implied by the revision path mapped on the file server and the *archivePath*. This is useful if the metadata has been manually moved or renamed.

```
vtransfer -x -m S:\VMFS\PDBs\PDB-1\archives\Project-1\Hello.txt-arc Hello.zip
```

Example-3: Import an Archive from a Zip File

This example imports metadata and revisions from an export file, like the ones produced in examples 1 and 2. Any existing archive information is overwritten.

```
vtransfer -i S:\VMFS\PDBs\PDB-1\archives\Project-1\Hello.txt-arc Hello.zip
```

Example-4: Move an Archive

This example moves the metadata and revisions associated with a versioned file to a new location on a file server.

```
vtransfer -r S:\VMFS\PDBs\PDB-1\archives\Project-1\Hello.txt-arc
    S:\VMFS\PDBs\PDB-1\archives\Project-1\Boneyard\Hello.txt-arc
```

Example-5: Rename an Archive

This example renames a versioned file on a file server and updates the metadata and revisions to match.

```
vtransfer -r S:\VMFS\PDBs\PDB-1\archives\Project-1\Hello.txv
    S:\VMFS\PDBs\PDB-1\archives\Project-1\Hello.txt-arc
```

Example-6: Fix an Archive

This example fixes the relationship between the metadata and revision data by moving the revisions to a location that mirrors the metadata location, and then updates the metadata. This is useful if the metadata has been manually moved or renamed.

```
vtransfer -f S:\VMFS\PDBs\PDB-1\archives\Project-1\Hello.txt-arc
```

Using the File Server in a Cross-Platform Environment

To access the same archives from UNIX and Windows systems, you must perform some extra creation and configuration steps to ensure trouble-free cross-platform access.

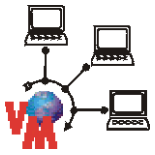
Creating

If the project database is to be on a UNIX server and accessed from Windows clients, you must create it from a Windows client using a UNC path (\\myserver) or Windows-style drive mapping (S:\myserver). Project databases created with UNIX-style paths are not accessible in a cross-platform environment.

Configuring

If you will be accessing the archives:

- Both directly and via the file server, you must perform all steps listed in the section on cross-platform environments.
- Only via the file server, you need perform only the following steps:
  - Translate end-of-line sequence.
  - Make user ID's case insensitive.
  - Make the case of files and directories consistent.



NOTE These steps are denoted in the text by the graphic to the left.

For information on performing these steps using the:

- Desktop client, see "Setting Up Version Manager for Cross-Platform Use" on page 39.
- Command-line, see "Cross-Platform Environment" on page 117.

Security Considerations

By default, the Version Manager File Server is configured to strike a balance between security and performance considerations. We recommend that you carefully evaluate your environment and usage in order to optimize the file server for your situation. Please review the following recommendations.

General recommendations

- As a starting point, regardless of specific considerations:
  - Enable the **Authenticate requests** option (this is the default setting).
  - Disable the **Enable remote configuration** option (this is the default setting).
  - Set the **Enable Utilities** option to **Local Only** (this is the default setting).
  - Enable an access control database for path maps. See "Creating a Default Access Control Database for Path Maps" on page 166.
  - Limit access to the system that hosts the Version Manager File Server. Version Manager clients running on the file server host can bypass file server security for certain metadata read operations. However, all write and revision operations are subject to security.

Internet specific recommendations

- To allow web client access by users outside of your firewall:
  - Place the Version Manager File Server on a separate system located behind your firewall (locate the Version Manager Web Server outside of your firewall).
  - Enable SSL support on the Version Manager File Server. See "Enabling Secure Socket Layer on the File Server" on page 166.

More information For more information on configuring Version Manager security, see the following table:

For information on...	See...
Configuring file server security options	"Configuring the File Server" on page 141
Path map security options	"Configuring Path Map Security Options" on page 140
Configuring Version Manager security	Chapter 6, "" on page 203

Creating a Default Access Control Database for Path Maps

You can specify an access control database to control access to a path map. Once an access control database is specified, all access will be validated against it. You will not be able to create project databases for the path map without a valid user ID. If you specify a non existent access control database, no one will be able to access the path map.

To facilitate the setup of path map security:

1. Create a dummy project database that is not located on a file server path map.
2. Configure an access control database for the dummy project database.
3. Copy the access control database to a location accessible by the file server.
4. In the **Access control database** field of the Options pane, specify the location of the access control database. The specified access control database will be associated with all new path maps created on the file server.

Enabling Secure Socket Layer on the File Server

To increase security, you can enable Secure Socket Layer on the Version Manager Application Server.

NOTE Enabling SSL on the Version Manager Application Server will affect all Version Manager applications running on it: Version Manager File Server, Version Manager Web Server, and WebDAV.

The following procedure allows you to enable SSL, but it does not necessarily address all SSL related considerations you may have. For more information on enabling SSL, see the following document:

Install_Dir\serena\vm\common\tomcat\webapps\tomcat-docs\ssl-howto.html

NOTE Ignore the **Download and Install JSSE** portion of the `ssl-howto.html` file. The JSSE is already installed.

To enable SSL:

1. Run the following command from a command prompt:

On Windows:

```
Install_Dir\serena\vm\common\jre\win32\bin\keytool -genkey -alias tomcat -keyalg RSA
```

On UNIX (where *OS* is: aix, hpux, linux, or solaris):

```
Install_Dir/serena/vm/common/java/OS/jre/bin/keytool -genkey -alias tomcat -keyalg RSA
```

2. When prompted, enter the keystore password. The default is `changeit`.
3. When prompted, enter the information (company name, etc.) that will be displayed to users when they try to access the server.
4. When prompted, enter a password for the key itself. This must be the same as the keystore password you entered above.
5. Open the following file in a text editor (on UNIX, reverse the slashes):

   ```
   Install_Dir\Serena\vm\common\tomcat\conf\server.xml
   ```

6. Find the following section in the file:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<!--
<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
   port="8443" minProcessors="5" maxProcessors="75"
   enableLookups="true" acceptCount="100" debug="0" scheme="https"
   secure="true" useURIValidationHack="false"
   disableUploadTimeout="true">
<Factory
   className="org.apache.coyote.tomcat4.CoyoteServerSocketFactory"
   clientAuth="false" protocol="TLS" />
</Connector>
-->
```

7. Remove the opening and closing comment tags from around the block of text (but not from the section heading). This enables SSL on port 8443. To specify a different port, see the `ssl-howto.html` file.

NOTE The default http port 8080 is still functional. You may want to comment out the block of text that enables port 8080 or otherwise restrict access to the port.

8. For better performance, change `enableLookups` from `true` to `false`.
9. Save the file and restart the Version Manager Application Server.
10. Configure your clients to use https: and the SSL enabled port. See "Configuring Clients for Use with File Servers" on page 167.

Configuring Clients for Use with File Servers

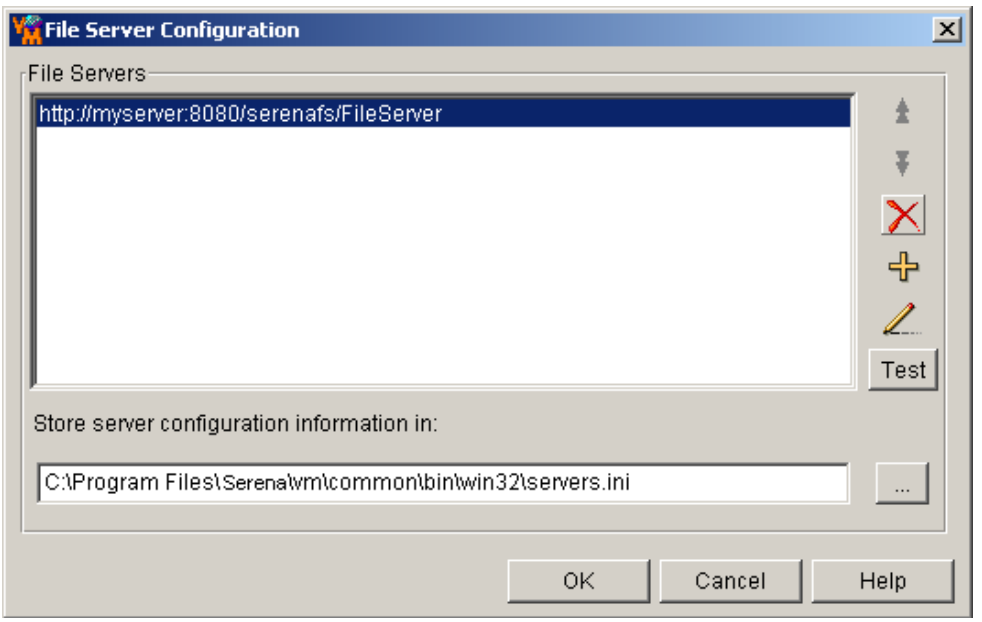
You must enter the paths to your Version Manager File Servers into a server configuration file so all Version Manager clients know where the file servers are located. If you save the server configuration file to a network location, you can point each client system to it rather than creating a new server configuration file on each user's system.

Configuring File Server Access when the Desktop Client Is Installed

To configure clients for use with file servers:

1 Launch the desktop client.

2 Select Admin | File Servers. The File Server Configuration dialog box appears.



3 Do any of the following:

- To reposition a selected server in the **File Servers** list, click the Up or Down arrow button. Version Manager searches starting from the top of this list.

- To delete a selected server from the list, click the Delete button.

- To add a server to the list, click the Add button. The Add File Server URL dialog box appears.

  a Enter the name of the system hosting the server. For example, *MyServer*. Optionally, you may specify a port. For example, *MyServer*:*Port*. By default, port 8080 is used. (If you have configured a web server for use with the Version Manager Web Server Application, you may use its port.)

  By default, your entry is prefaced with `http://`. To specify an SSL server, enter

  `https://`*MyServer*.

  To set the port used by clients local to the file server, configure the File Server Configuration dialog box on the system hosting the file server.

IMPORTANT! If the file server and the web server are located on the same system and there will be more than 50 concurrent users, set the port used by clients local to the file server to 8090. Otherwise, the server may hang. By default, clients local to the server use port 8080.

  b Click **OK**.

- To edit the URL of the selected server, click the Edit button. The Edit File Server URL dialog box appears.
  - **a** Edit the URL in the **Name** field.
  - **b** Click **OK**.
- To test the connection to the selected server, click the **Test** button.

4 Enter a path and file name (`*.ini`) in which to store server configuration information in the **Store server configuration information in** field, or accept the default location.

Automatically populate File Servers list

To ease configuration for all users, set up the list of servers then place the configuration file in a network location where all users can access it. Make this location read only so users cannot accidentally modify the file. Then the users need only specify the shared location of the configuration file to automatically populate their own **File Servers** list.

If you do a workstation installation, the default location would automatically be a shared network location.

5 Click **OK**.

Configuring File Server Access when Only the IDE Client Is Installed

NOTE If the location of the `servers.ini` file is not specified in the `islv.ini` file, the IDE client will create a local `servers.ini` file the first time you attempt to access a file server. If you want all users to access a single `servers.ini` file on the network, see "Specifying the Location of the Servers.ini File when the Desktop Client Is Not Installed" on page 170.

NOTE This procedure does NOT apply to the rich integrations to Eclipse and .NET. Use the desktop client or the manual method to enable file server access for those interfaces.

To configure file server access from the IDE client:

1 From any IDE client dialog box that can open a project database, such as Get or Check Out, click the **Open Database** button. The Select File Server dialog box appears.

Add/edit server

2 To add a new server, double-click an empty cell in the **File Server** column. To edit an existing server entry, double-click the entry for that server.

3 Enter the name of the system hosting the server. For example, *MyServer*. Optionally, you may specify a port. For example, *MyServer*:*Port*. By default, port 8080 is used.

By default, your entry is prefaced with `http://`. To specify an SSL server, enter `https://`*MyServer*.

Delete server

4 To delete a server entry, select the server in the **File Server** column and press the DELETE key.

Order servers

5 To reposition a server in the **File Server** list, drag-and-drop it to the desired position. Version Manager searches starting from the top of this list.

Save changes

6 To save your changes to the `servers.ini` file, click the **OK** button.

Test servers

7 To test the connection to the servers listed in the **File Server** column, click the **Test** button. The status of each server is updated in the **Status** column.

Specifying the Location of the Servers.ini File when the Desktop Client Is Not Installed

You can directly edit the Version Manager `islv.ini` (Windows) or `.islvrc` (UNIX) file for each workstation to specify the location of the `servers.ini` file. You would want to do this if:

- Only the command-line interface is installed.
- Only the IDE client is installed and you want all users to access a single `servers.ini` file on the network.

NOTE If the location of the `servers.ini` file is not specified in the `islv.ini` file, the IDE client will create a local `servers.ini` file the first time you attempt to access a file server.

To manually update client islv.ini files:

1 Open the `islv.ini` (Windows) or `.islvrc` (UNIX) file in a text editor. The file is located in the Windows root directory (usually `C:\WINNT`) or your home directory on UNIX.

2 Look under the `[PVCSGUI_6.5]` heading for the `pvcs.fileserver.path=` entry. By default, it looks like this on Windows:

```
pvcs.fileserver.path=C:\Program Files\Serena\vm\common\bin\win32\servers.ini
```

And like this on UNIX:

```
pvcs.fileserver.path=/usr/serena/vm/common/bin/OS/servers.ini
```

3 Modify the path to reflect the location of a configured `servers.ini` file.

NOTE You can configure the `servers.ini` file from the desktop client and the IDE client.

4 Save the file.

TIP If you plan to use the workstation install feature to automate the set up of your workstations, you can save time by modifying the `islv.ini` or `.islvrc` file in the workstation install. For more information about workstation installs, see the *Version Manager Installation Guide*.

How to Access File Servers from Version Manager Clients

Once a user's system is configured, the user can access a server-based project database from the:

- **Desktop client** by selecting File | Open Project Database and then selecting **File Servers** from the **Look in** list.
- **IDE client** via any dialog box that can open a project database, such as Get or Check Out, by clicking the **Open Database** button.
- **Command line (CLI**) by specifying an archive, configuration file, or project database path relative to the Client Name (as defined in the Path Maps pane of the Version Manager File Server Administration Utility). For example:

  ```
  \\ClientName\Project-1\Foo.txt-arc
  ```

NOTE If an access control database or LDAP authentication is associated with the Client Name, you must use the `-id` option, or set an environment variable, to present a user ID and password for validation (**`-id`***`user_id:user_password`*). See the *Version Manager Command-Line Reference Guide*.

- **Project command line (PCLI)** by specifying an archive, configuration file, or project database path relative to the Client Name (as defined in the Path Maps pane of the Version Manager File Server Administration Utility). For example:

  ```
  \\ClientName\Project-1\Foo.txt-arc
  ```

- **Web client** by configuring the web server to access a project database located on a Version Manager File Server. You do this by specifying the location of the project database relative to the Client Name when you set up the servlet. No set up is required on the web client itself.

IMPORTANT! If the file server and the web server are located on the same system and there will be more than 50 concurrent users, set the port used by clients local to the file server to 8090. Otherwise, the server may hang. By default, clients local to the server use port 8080. See "Configuring File Server Access when the Desktop Client Is Installed" on page 168.

Chapter 5

Managing Your Environment

Introduction

This chapter discusses tasks that you can perform to manage your Serena PVCS Version Manager environment. As an Administrator, you may need to define public workspaces, add custom tools to the tool bar and Tools menu, change the attributes of existing archives, or change the locations of archives.

Using Workspaces

A workspace is a collection of work settings defined for a project database. Specifically, workspaces store:

- The workfile locations defined for the project database, and the projects, subprojects, and versioned files contained within the project database. A workfile location is the directory to which you check out files and from which you check in files. You define workfile locations when you create a project database, create a project, or add workfiles.
- The default version, which specifies the revision Version Manager will automatically operate on (for actions such as checking out) when you don't specify a revision number or version label. This is usually a floating label which selects the tip of a particular branch.

IMPORTANT! The rich IDE integration (Eclipse; Visual Studio) uses the default version (label) to determine which files are visible in a given Version Manager workspace. To avoid confusion, it is important that you understand how this works.

If you apply a default version or change the existing one for a project database or for a workspace, only files that have the version label will appear in the IDE client. If the project and solution files do not have these labels, you will see no files.

To avoid the potential for confusion:

- Create a Version Manager workspace for any user or group of users who may need their own default version (label).
- Define default versions on a workspace-by-workspace basis (File | Properties | Workspace Settings tab), rather than for the entire project database.

- The base version and branch version used to facilitate automatic branching.
- 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.

Workspaces can be created for project databases only in the desktop client. You can create multiple workspaces, but only one can be set on a project database at a time.

Even though workspaces are associated with project databases only, they contain the settings for all of the projects and subprojects within the database. For example, the workfile locations defined for a project database and all of the projects within the database, are stored in a workspace. Although only one workspace can be set on a project database at a time, each project in the database can have a different workfile location defined in that one workspace.

Types of Workspaces

Workspaces are public or private. The settings in a public workspace affect everyone using the workspace. The settings in a private workspace affect only the user who created and set it. A private workspace cannot be shared with other users.

Version Manager creates a default public workspace, called the Root Workspace, when a project database is created. Only one workspace defined for a project database can be named Root Workspace. The settings that are initially defined in the Root Workspace are:

- The workfile location you set for the project database when you create the database.
- 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.
- 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 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.

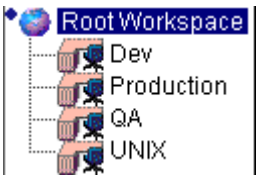
Root Workspaces cannot be deleted or renamed. However, all users can edit the settings defined in the Root Workspace if they have the correct privileges. See the privilege tables on page 239 to determine the correct privileges.

When you add projects to a project database, Version Manager adds workspace settings for each of the projects to the Root Workspace definition.

Anyone using the Root Workspace will be affected by changes made to it. For this reason, using private workspaces is highly recommended when multiple users are accessing the same project database. Typically, the Administrator will create several public workspaces to define the work environments of specific functional groups, such as Dev, Production, and QA. The Administrator can deny privileges to change public workspaces, as well as to delete and rename them. From a public workspace, a user can create a private workspace for working on a local drive.

Workspace Hierarchies

At the top of a workspace hierarchy is the Root Workspace. In the following example, Dev, Production, QA, and UNIX are all child workspaces beneath the Root Workspace.



The initial settings of the Root Workspace are defined as follows:

This setting. . .	Is initially set from. . .
Workfile Location	The workfile locations of the project database and projects when they were created.
Default Version Branch Version Base Version	The master configuration file associated with the project database and the project configuration files associated with the projects (if any).
Default Promotion Group	This value is **not** automatically set.

When you create a workspace, you must choose a workspace from which the new workspace will inherit its settings. This workspace is called the parent workspace. You can change the settings of the new workspace to any value you want, thus, overriding the inherited values of the parent workspace. If you do **not** change (override) the values, the values remain those of the parent workspace. And, if at a later time you change the settings of the parent workspace, the settings of the workspace that was created from the parent workspace are also changed.

For example, if you create a new workspace and do not change the inherited workfile location of `Z:\PRJDB\WORK` and then later change the workfile location of the parent workspace to *`Y:\PRJDB\WORK`*, the workfile location of the child's workspace also changes to `Y:\PRJDB\WORK`.

Once you override the initial values of a newly created workspace, changing values in the parent workspace has no effect on the child workspace. For example, if you create a new workspace and change the inherited workfile location and then later change the workfile location of the parent workspace, the workfile location of the child's workspace does **not** change.

You can return to the original, inherited value by removing the current value in the field and leaving the field with no value.

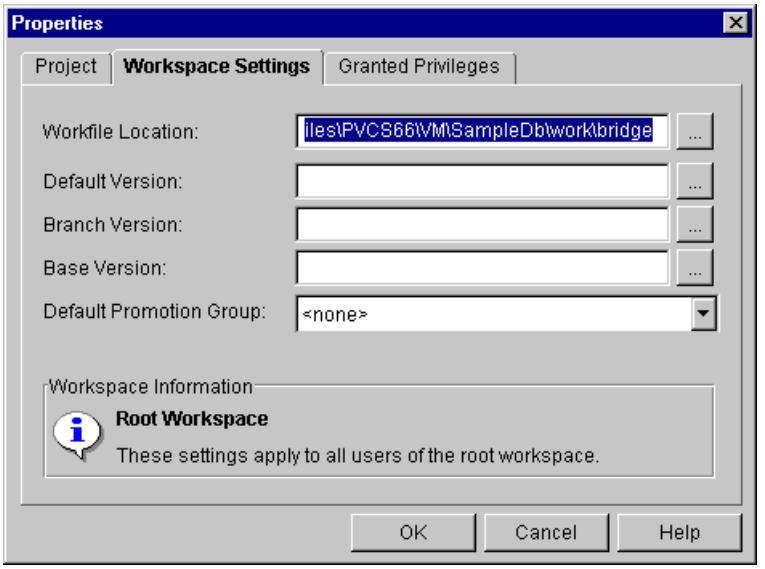
Viewing Workspace Settings

You may find a need to check the settings of a workspace. To do this:

1 Select the project database or project.

2 Select File | Properties. The Properties dialog box appears with the Project Database or Project tab active.

3 Click the Workspace Settings tab. This tab lists the workspace settings.



NOTE The Default Promotion Group field is not displayed on the Workspace Settings tab if a promotion model is not defined for the database.

4 When you are finished checking the workspace settings, click OK. You can use File | Set Workfile Location to quickly view or edit the setting of a workfile location in a workspace.

Using Public Workspaces

As an Administrator, you will be working with public workspaces—workspaces that affect all of your users. Private workspaces are typically created by users, not the Administrator. For information about using private workspaces, see the *Serena PVCS Version Manager User's Guide*.

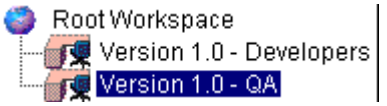
Public workspaces are particularly useful when your organization uses:

- Identical workfile locations on different local or network drives for different groups, such as Quality Assurance (QA) and Developers.
- Different workfile locations on the same drive for different groups, such as QA, Dev, and Production.

Different Network Drives

In the first situation, you would define the Root Workspace using relative workfile locations for all of the projects and subprojects in a project database. The workfile location for the project database must be an absolute path, such as `c:\prjdb1\work`. Then, the Root Workspace might define a relative workfile location for a project such as class_lib, making a workfile location for the project of `c:\prjdb1\work\class.lib`.

You would then create two public workspaces, such as "Version 1.0 - Developers" and "Version 1.0 - QA."

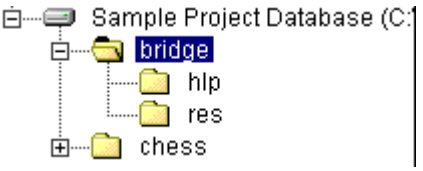


Each public workspace inherits the workfile locations defined in the Root Workspace. For each public workspace, you would define a different absolute workfile location for the project database, such as `Z:\Version_1.0` and `Y:\Version_1.0`. These absolute workfile locations would match the drive where each group actually works. You need not change the relative workfile locations of the projects because the relative locations (such as classlib) will append themselves to the new absolute workfile location of the project database, making, for example, a workfile location of `Z:\Version_1.0\class.lib`.

By creating these workspaces, users can quickly and easily switch between the QA work environment and Developer work environment.

The Same Drive

To define different workfile locations on the same drive for different departments, you would define the Root Workspace using an absolute workfile location for the project database and relative workfile locations for all of the projects and subprojects in the project database. For example, let's say we have the following project database with the following projects and subprojects:



To define different workfile locations, you would create a public workspace for each department. The new public workspaces would inherit the workfile locations from the Root Workspace; therefore, in the new workspaces, you need only append the department directory to each workfile location for the projects and subprojects in the project database.

For example, in Windows:

Workspace	Workfile Location
Root	`C:\sample\work` for the project database that contains the Bridge project
Root	`C:\sample\work\bridge` for the Bridge project
QA	`C:\sample\work\bridge\qa` for the Bridge project
Dev	`C:\sample\work\bridge\dev` for the Bridge project
Production	`C:\sample\work\bridge\prod` for the Bridge project

In the example above, for the QA workspace, you would append \qa to the end of the workfile location inherited from the Root Workspace. The complete workfile location would then be `C:\sample\work\bridge\qa`; `bridge\qa` becomes a relative workfile location. If at any time you change the workfile location of the project directly above the Bridge project in the hierarchy (which in this case is the project database), the workfile location for the Bridge project also changes. For example, if you change the workfile location of the project database to `D:\sample\test\work\source`, then the workfile location for the Bridge project changes to `D:\sample\test\work\source\bridge\qa`.

Restricting the Creating, Renaming, and Deleting of Public Workspaces

You can restrict users from creating, renaming, and deleting public workspaces by **not** giving them either the SuperUser or Unlimited privilege sets. These two privilege sets grant users permission to perform these public workspace actions. Users who can create, rename, and delete public workspaces can disrupt the work of other users.

NOTE The access privileges of the user logged in at the project root is checked to determine if the user is allowed to operate on public workspaces. If your environment includes configuration files and access control databases on sub-projects of the project root, the privileges of the user logged in to a sub-project are not used when deciding whether the user can operate on public workspaces.

For example:

- If a user renames a public workspace that is the default workspace of other users, the Root Workspace becomes their default workspace.
- If a user deletes a public workspace that is used by other users, the workspace is no longer available to those users, disrupting their work.
- If a user deletes a public workspace and then creates another public workspace with the same name, users will be using a public workspace in which the settings may have changed.

For information about assigning privileges to users, see "Using Security" on page 203.

You can also restrict users from modifying the settings defined in public workspaces by denying the users certain privileges. See the privilege tables on page 239.

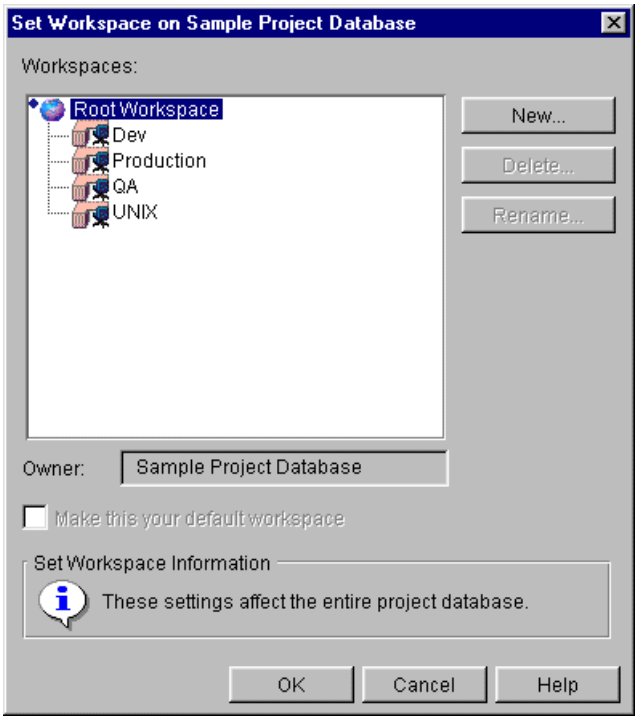
Creating a Public Workspace

As an Administrator, you will be creating public workspaces that affect all of your users. Private workspaces should be created by the user, not the Administrator. For information about creating private workspaces, see the *Serena PVCS Version Manager User's Guide*.

When you create a public workspace, you select a public workspace from which the new workspace will inherit its settings. See "Workspace Hierarchies" on page 175.

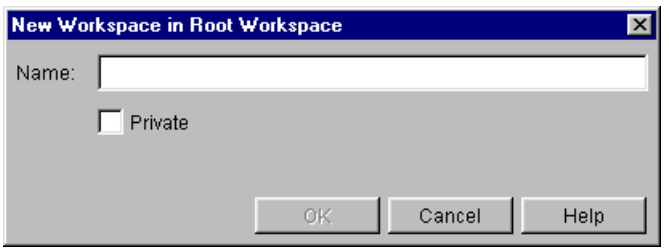
To create a public workspace:

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



3 Select the workspace from which the new workspace will inherit its settings. Public workspaces cannot inherit from private workspaces.

4 Click New. The New Workspace dialog box appears.



5 Enter a unique name for the workspace. No two workspace names can be the same, unless the parent workspaces are different.

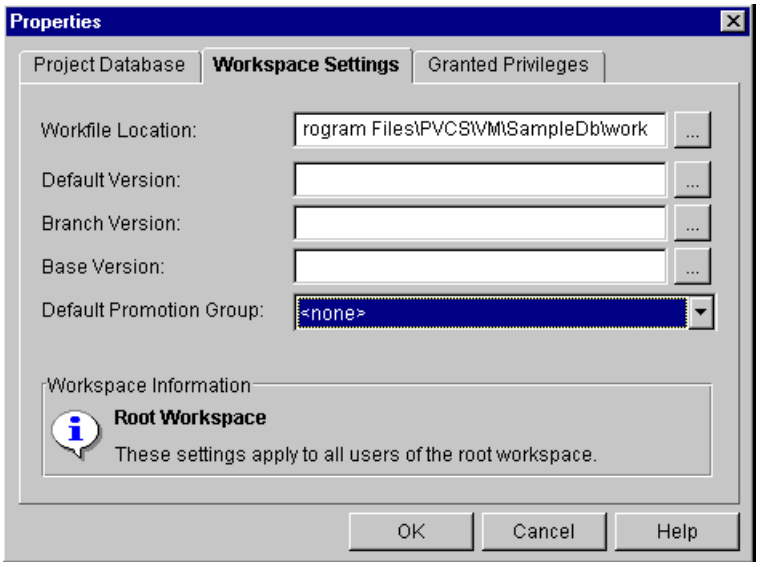
6 Click OK. The Set Workspace dialog box appears with the new workspace selected. You must leave it selected to set the new workspace as the current workspace. Once it is set as the current workspace, you can define its settings.

7 Click OK to close the Set Workspace dialog box. The new workspace is now set as the current workspace.

8 To define the settings for the new workspace, do the following:

a Select File | Properties. The Properties dialog box appears with the Project Database or Project tab active.

b Click the Workspace Settings tab. This tab lists the workspace settings.



c To change the settings, do any of the following:

- Modify the workfile location. You can define a relative workfile location by appending a directory name to the initial workfile location or by entering only a directory name, such as qa. When you enter only a directory name, Version Manager appends this directory to the workfile location of the parent project. You **cannot** specify a relative workfile location for a project database.

  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.

- Specify the revision to operate on for all actions in the **Default Version** field. You can specify a revision number or a version label. By default, Version Manager uses the tip revision.

IMPORTANT! The rich IDE integration (Eclipse; Visual Studio) uses the default version (label) to determine which files are visible in a given Version Manager workspace. To avoid confusion, it is important that you understand how this works.

If you apply a default version or change the existing one for a project database or for a workspace, only files that have the version label will appear in the IDE client. If the project and solution files do not have these labels, you will see no files.

To avoid the potential for confusion:

- Create a Version Manager workspace for any user or group of users who may need their own default version (label).
- Define default versions on a workspace-by-workspace basis (File | Properties | Workspace Settings tab), rather than for the entire project database.

- Define automatic branching by entering the appropriate version label for Base, Branch, and Default. *Base Version* specifies the fixed version label that you assigned to mark the revision from which you want to start a branch. *Branch*

Version specifies the floating version label that you assigned to the tip of the branch. Initially, this will be the revision from which you want to branch. As you check in subsequent branch revisions, this label will automatically reassign itself, or float, to the tip revision on the branch. *Default Version* is the floating version label you specified for the Branch Version option. This version label tells Version Manager which revision to operate on for all actions. See "Branching and Merging Files" on page 263.

Note that these values will override the values in any configuration file associated with the project database or projects if these options are not disallowed in the master configuration file associated with the project database.

- Define a lowest-level promotion group by selecting one from the Default Promotion Group field. This field is available only if a promotion model is in effect for the selected project database or project. This setting is useful when a promotion model has more than one lowest-level promotion group defined. 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 revision, 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 the check out and lock actions. If a promotion model is in effect, then a lowest-level promotion group is required for checking out and locking revisions, and the user is prompted to enter a value.

  When adding workfiles, if a promotion model is in effect and no default value has been set for the lowest-level promotion group, the files are added with no promotion group associated with the new revision.

9 Click OK. You have created a new public workspace and defined its settings. If you want to define workspace settings for any of the projects in the project database, go to Step 10.

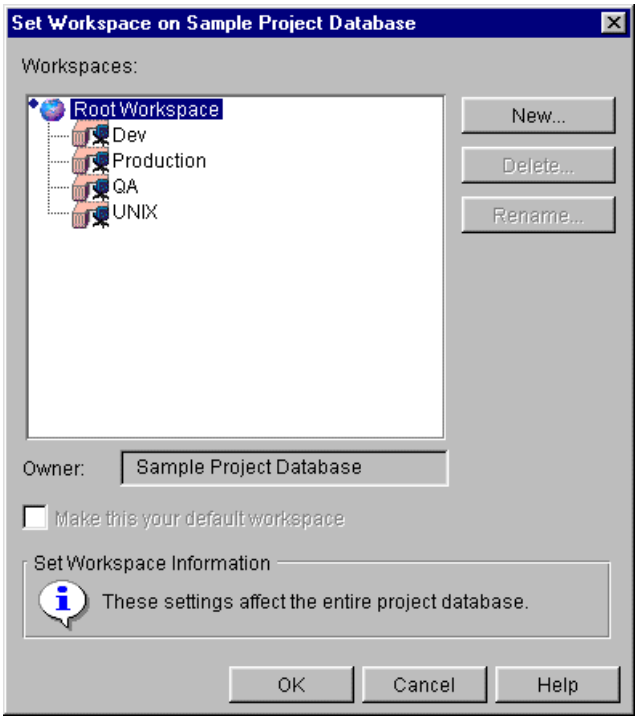
10 To define workspace settings for a project in the project database, select the project in the project pane and repeat Steps 8 and 9.

Setting a Workspace

To set a workspace:

1 Select the project database or any of the projects in the project database for which you want to set a workspace.

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



NOTE The small blue diamond icon to the left of the Root Workspace indicates that the Root Workspace is set as the default workspace, meaning for each Version Manager session the Root Workspace is initially set for the project database. The **Make this your default workspace** check box is grayed when the current default workspace is selected in this dialog box, as shown above. When you select another workspace to set for the project database, the check box becomes active.

3 Select the workspace to be associated with the project database.

4 To associate this workspace with the project database for subsequent Version Manager sessions, select the **Make this the your default workspace** check box. Otherwise, this workspace will only be active during the current Version Manager session.

5 Click OK.

Adding Custom Tools

In the Version Manager desktop client, you can add custom tools that appear as icons on the tool bar and/or as menu items on the Tools menu. This feature is useful when you want to create shortcuts for frequently used applications or tasks.

Each project database can have its own custom tool bar and Tools menu. When you switch from a project database with a custom Tools menu to one without, Version Manager disables the Tools menu. When you switch from project database to project database, Version Manager loads the appropriate tool bar.

The tool configuration features enable you to:

- Add user-defined tool bar icons.
- Remove user-defined tool bar icons—you cannot remove default Version Manager tool bar icons.
- Add menu items to the Tools menu on the menu bar.
- Store and load tool bar and menu bar configurations per project database.
- Pass information such as a user ID, the name of an archive, and the name of a workfile to the executable associated with the tool bar icon or menu item.

Adding a Custom Tool

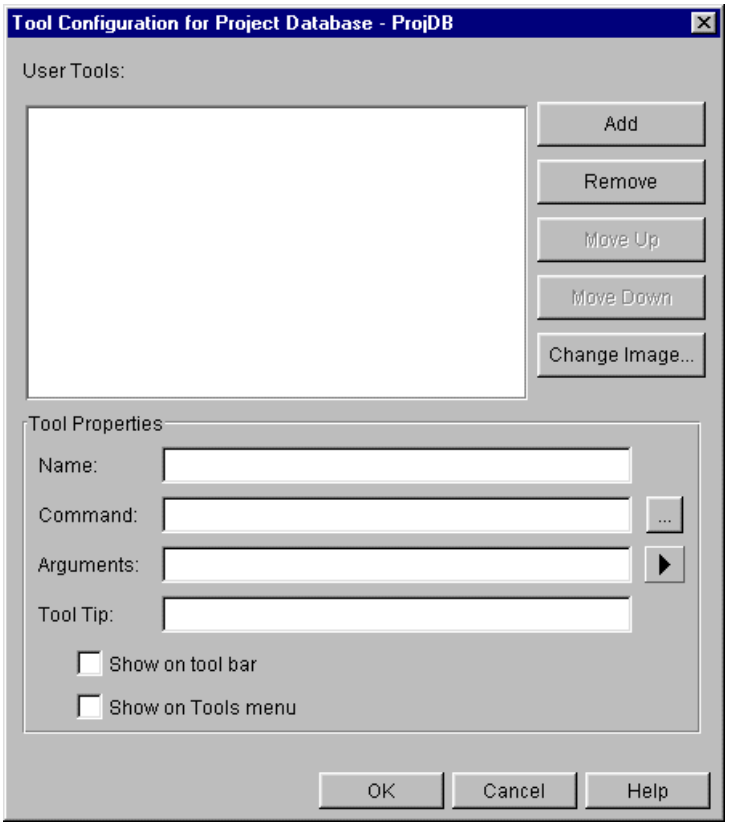
When you add a custom tool, you can specify the following:

- The image to use for the icon. Version Manager provides a default image (.GIF file) for new tool bar icons. You can use your own .GIF or .JPG file. The image is scaled to 21 pixels wide x 19 pixels tall.
- The command (file) to be executed when a user clicks the icon or selects the menu item. Version Manager supports the following:
  - Stand-alone Windows (.EXE) and UNIX executable programs
  - Windows batch scripts (.BAT)
  - Windows program information files (.PIF)
  - UNIX shell scripts
- Information (arguments) to pass to the executable associated with the tool bar icon or menu item. See "" on page 275.
- The tool tip text to display when the mouse is stationary over the tool bar icon.

To add a custom tool:

1 Select a project database.

2 Select Admin | Tool Configuration. The Tool Configuration dialog box appears.



3 Click Add. The default icon with the name [Untitled] appears in the User Tools box on the left. You can change the icon image as discussed in Step 8.

4 Specify a name for the new tool in the **Name** field. If the new tool is a menu item, the text you enter here is the name of the menu item on the Tools menu. If the new tool is a tool bar icon, the text you enter here is the tool bar tip text if you do not specify different text in the Tool Tip field.

5 In the **Command** field, specify the executable file to be invoked when a user clicks the new icon or selects the new menu item.

6 Select the **Show on tool bar** check box to make the new tool a tool bar icon.

7 Select the **Show on Tools menu** to make the new tool a new menu item on the Tools menu.

8 Optionally, you can do any of the following:

- To change the icon image, click **Change Image** and select the GIF or JPG file to use instead of the default one.
- To specify tip text other than the text specified in the Name field, enter the appropriate text in the **Tool Tip** field.

- To pass information (arguments) to the executable specified in the Command field, click the button next to the **Arguments** field, and select a command-line macro or the parameter file (__EventParmFile__) that will pass event information to the executable.

For more information about how to pass information to an icon or menu item event trigger, see "Passing Information to Event Triggers" on page 280.

9 Click OK. By default, the tool configuration information is saved in a file named tools.cfg beneath the project database location. Do not move this file unless you specify its new location, or Version Manager will not be able to customize the tool bar and menu bar.

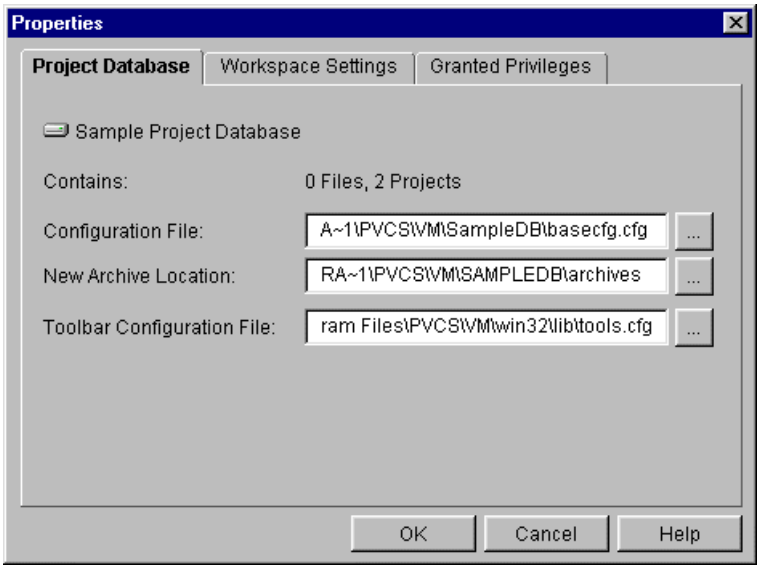
Moving the Tools Configuration File

You can move the tools configuration file (tools.cfg) to a new location. When you do, you must specify its new location so that Version Manager can use the file to customize the tool bar and menu bar. You cannot move a tools configuration file for a 5.3/6.0 project database.

To specify a new location for the tools configuration file:

1 Select the project database that is associated with the tools configuration file.

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



3 Enter the new location in the **Configuration File** field, or click the Browse button to select one.

4 Click OK.

Changing Attributes of Existing Archives

When Version Manager creates archives, it sets the archives' attributes according to the settings in the configuration files, or if Version Manager has not been configured to use configuration files, it sets the attributes according to Version Manager's defaults. See "Default Settings when No Configuration File Is Used" on page 56 for a list of Version Manager defaults.

After the archives are created, you can change some of the attributes of existing archives. When you change attributes of existing archives, Version Manager changes the attributes in all of the revisions in the archives.

When to Change Attributes

Only change the attributes of existing archives when you change the configuration settings that determine the attributes for newly created archives. For example, if an archive has an attribute that prevents multiple locks, and you change the configuration settings to permit multiple locks, only newly created archives will allow multiple locks. The existing archives will not allow multiple locks. The following scenario addresses this situation.

The project leader of a small development group has released the first version of a new Windows product. Until the product was released, the developers only needed to work on trunk tip revisions. Before the project leader created the archives, he configured archives to allow only one lock per archive, thus preventing developers from inadvertently creating branches.

The project leader now wants to develop a version of the product for UNIX while continuing development on the next version for Windows. To do so, he wants to create a branch off of each tip revision for the UNIX version of the product. Because parallel development requires that users be able to lock multiple revisions in an archive, he must change the attributes of existing project files.

Archive Attributes

The following table lists the archive attributes you can change.

Attribute	Definition
Exclusive Lock	Prevents multiple locks on a single archive.
Expand Keywords	Allows expansion of keywords on check in of a workfile.
Comment Prefix	Defines the comment prefix to insert before lines in the Log keyword expansion.
Write Protect	Protects archives from inadvertent deletion or modification.
Store Deltas	Stores only the tip revision as a complete copy of the workfile and all other revisions as a set of deltas.
Newline Character	Defines a newline character for the Log keyword expansion.
Translate EOL	Translates the end-of-line character.
Owner	Specifies the owner of the archive.
Access List	Specifies an access list for the archive.
Record Length	Specifies that either fixed-length or variable-length files are being stored. If fixed-length, you can specify the length of each record.
Column Masking/ Renumbering	Specifies options for files that contain line numbers.

Attributes Set by File Type

Some of the above attributes are set on a file type basis and others on all file types that are stored in archives. For example, the Translate EOL attribute is set by file type because this attribute should only be set for text files; setting this attribute for binary files may corrupt the archives. The Exclusive Lock is an attribute that is typically set the same for all archives regardless of file type. The attributes that are set by file type are:

■ Expand Keywords	■ Newline Character
■ Store Deltas	■ Record Length
■ Translate EOL	■ Column Renumbering
■ Column Masking	

Determining Existing Attributes

To determine the settings for existing archive attributes, generate a history report with archive information only and look at the Attributes section of the report. See "Generating History Reports" on page 301.

Using the Desktop Client

In the desktop client, you can change the attributes of existing archives associated with a project database, project, multiple versioned files, or a single versioned file. When you select a project database or project, you have the option of changing the attributes of existing archives of:

- All of the projects/subprojects within the selected project database or project
- Just the selected project database or project—not including the projects and/or subprojects

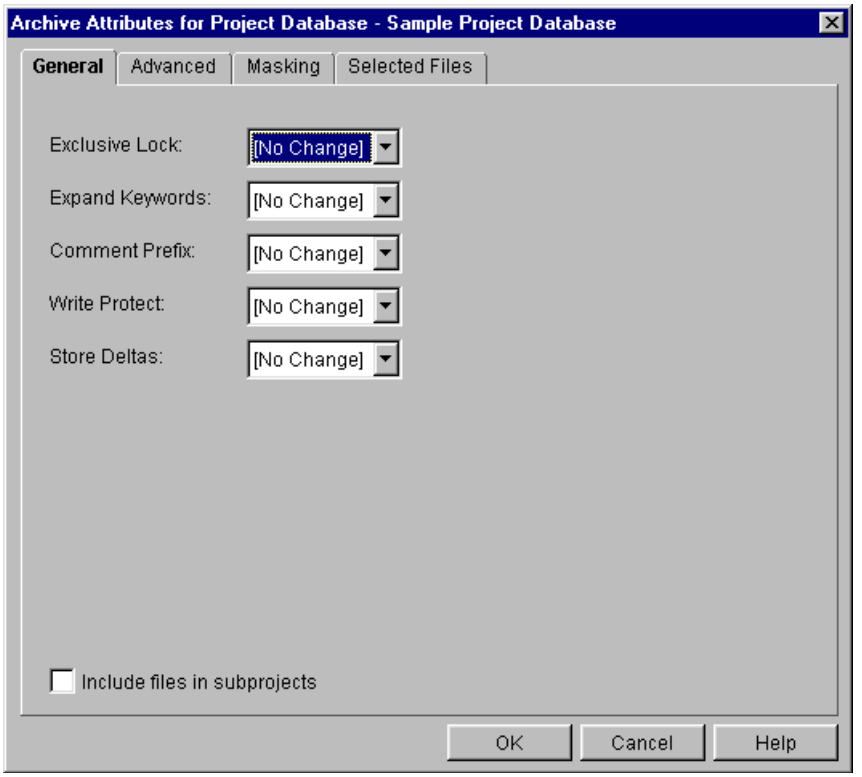
If you are changing attributes that are set on a file type basis (see "Attributes Set by File Type" on page 188), you can use the Version Manager File Filter (View | Filter | Wild Card) to display all versioned files that match a specific criteria. For example, you can display all files with the extension .java, and then select the versioned files.

NOTE A user may select multiple projects and folders when changing archive attributes. Refer to the *Serena PVCS Version Manager User's Guide* for more information on selecting multiple projects and folders.

To change archive attributes:

1 Select a project database, project, or versioned file(s). If you select a project database or project and you are changing attributes that are set on a file type basis, the attributes are changed for all file types in the project database or project.

2 Select Admin | Archive Attributes. The Archive Attributes dialog box appears with the General tab active.



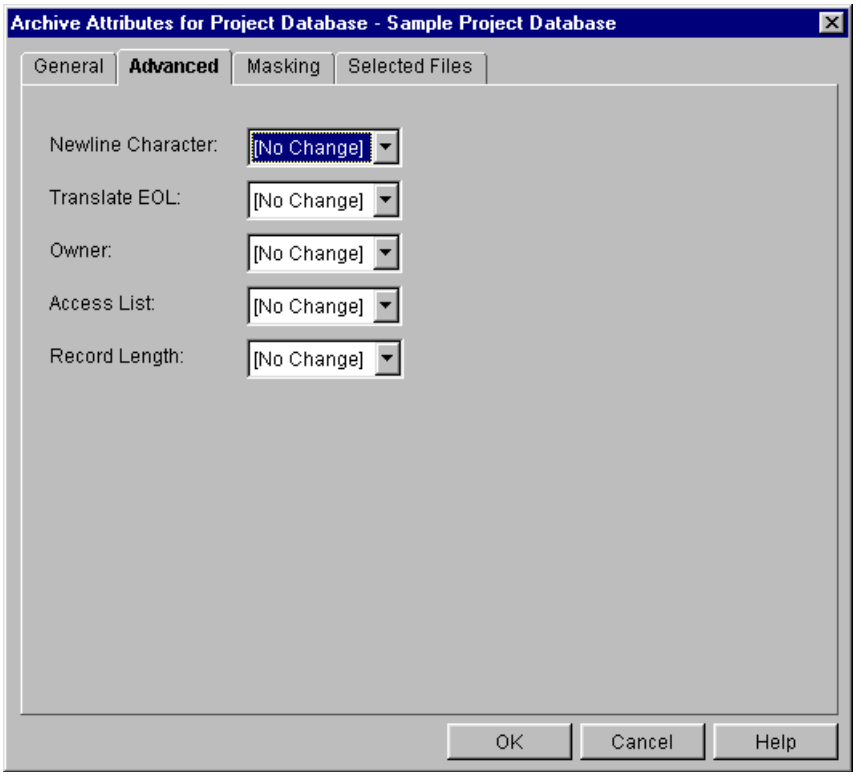
3 The setting for each attribute on the General tab is No Change, which means to leave the attribute setting as it is currently set. To change the settings, do any of the following:

- Select Yes or No for **Exclusive Lock**. Yes prevents multiple locks on a single archive. If you allow multiple locks by selecting No, more than one revision in an archive can be locked at the same time. Also, if Exclusive Lock is specified for an archive, the MultiLock directive is ignored for that archive. See "Archive Creation Options" on page 67.
- Select Yes or No for **Expand Keywords**. Yes expands keywords on check in of a workfile. This is an attribute that is set on a file type basis. See "Options Set on a File Type Basis" on page 96. Do **not** set this option for binary files as they may be corrupted.
- The **Comment Prefix** field applies only if you are expanding keywords. If you are, this field defines the comment prefix to insert before lines in the Log keyword expansion. Select Modify or Delete for Comment Prefix. Modify causes Version Manager to display a Comment Prefix text field in which you can enter a comment prefix. Delete removes the previously specified comment prefix. This is an attribute that is set on a file type basis. See "Options Set on a File Type Basis" on page 96.
- Select Yes or No for **Write Protect**. Yes protects archives from inadvertent deletion or modification. Version Manager commands automatically remove write-protection before modifying archives and replace it afterwards.
- Select Yes or No for **Store Deltas**. Yes stores only the tip revision as a complete copy of the workfile (the work image) and all other revisions as a set of deltas. No stores all revisions as complete copies of the workfile. This is an attribute that is

set on a file type basis. It is recommended that this be set to No for binary files because there may be large differences in these files. See "Storing Deltas" on page 98.

4 If you selected a project database or project in Step 1, select the **Include files in subprojects** check box to change the attributes for existing archives in all of the projects/subprojects.

5 Click the Advanced tab.



6 The setting for each attribute on the Advanced tab is No Change, which means to leave the attribute setting as it is currently set. To change the settings, do any of the following:

- The **Newline Character** field applies only if you are expanding keywords. If you are, this field defines a newline character for the Log keyword expansion. Select Modify or Delete for Newline Character. Modify causes Version Manager to display a Newline Character text field in which you can enter a newline character. Delete removes the previously specified newline character. This is an attribute that is set on a file type basis. See "Options Set on a File Type Basis" on page 96.

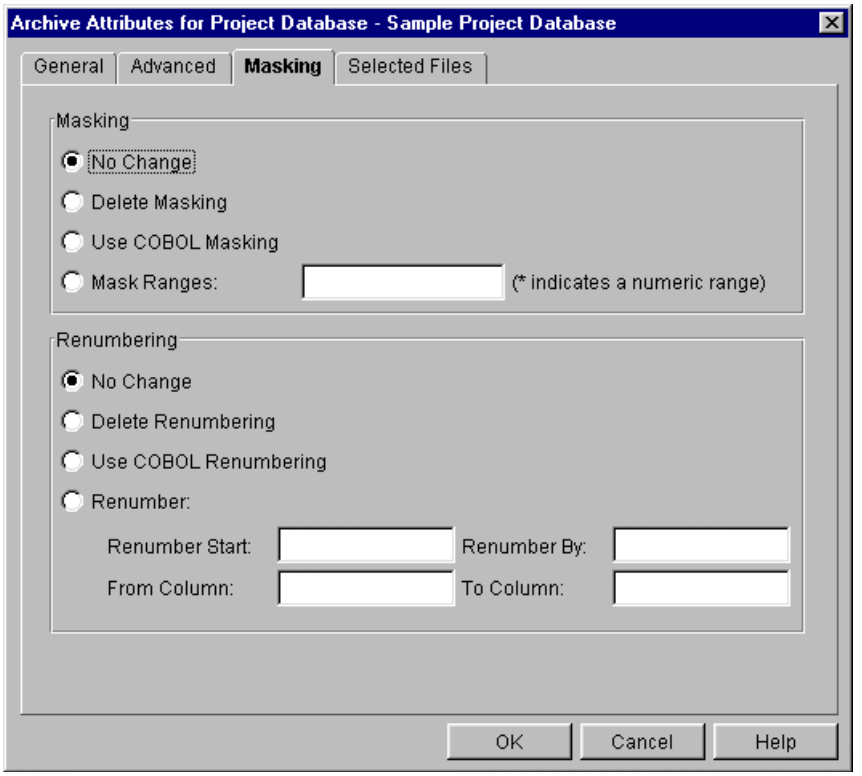
NOTE The conversion between Windows and UNIX newline characters are handled automatically.

- Select Yes or No for **Translate EOL**. Yes translates the end-of-line character. You should always set this option to Yes for text files. This is an attribute that is set on a file type basis. See "Options Set on a File Type Basis" on page 96.

CAUTION! Do not set this option on binary files; it may corrupt the archives.

- Select Modify or Delete for **Owner**. Modify causes Version Manager to display the Owner text field in which you can enter the user ID of the new archive owner. Delete removes the previously specified owner from the archive. See "Archive Creation Options" on page 67.
- Select Modify, Append, or Delete for **Access List**. Both Modify and Append cause Version Manager to display an Access List text field in which you can enter a new access list (in the case of Modify) or enter additional user IDs and/or groups to the existing access list (in the case of Append). The combined value of this field cannot exceed 254 characters in length. Delete removes the existing access list. See "Using Security" on page 203 for an explanation of access lists.
- Select Variable Length or Fixed Length for **Record Length**. Variable Length specifies to Version Manager that the archives are storing variable-length files. Fixed Length specifies that the archives are storing fixed-length files. Fixed Length causes Version Manager to display a Record Length text field in which you enter the length of records in fixed-length files so that Version Manager can generate deltas more efficiently. This is an attribute that is set on a file type basis. See "Options Set on a File Type Basis" on page 96.

7 Click the Masking tab to modify attributes that are applicable only if you are creating and modifying files that contain line numbers. The setting for each attribute on the Masking tab is No Change, which means to leave the attribute setting as it is currently set.



8 On the Masking tab, change any of the following attributes. These attributes are set on a file type basis.

- From the **Masking** group, select one of the following options:
  - **Delete Masking** to not use masking.

- **Use COBOL Masking** to use the default COBOL masking definition for COBOL files, which is: Mask columns 73-80 and mask columns 1-6 only if column 1 is numeric.
- **Mask Ranges** to enter the column to mask from and the column to mask to in the text field. For example, if you enter 1-6, Version Manager would convert columns 1 through 6 to spaces.

  To restrict column masking to numeric fields only, enter an asterisk (*) beside the range. For example, 1-6* masks the columns only if column 1 is numeric.

  You can also enter multiple ranges, separated by a comma. For example, 1-6*,10-12 masks the columns only if column 1 is numeric and would convert columns 10 through 12 to spaces.

- In the **Renumbering** group, select one of the following options:
  - **Delete Renumbering** to not use renumbering.
  - **Use COBOL Renumbering** to use the default COBOL renumbering definition for COBOL files, which is: renumber columns 1-6, start with the number 10 increment by 10.
  - **Renumber** and then enter:

  a The number to start numbering with in the **Renumber Start** field.

  b The number by which to increment each line number in the **Renumber By** field. For example, if you enter 20 as the start number and 5 as the number by which to increment each line, the first line would be numbered 000020, the second 000025, the third 000030, and so forth. This assumes you are renumbering six columns.

  c The column to renumber from in the **From Column** field and the column to renumber to in the **To Column** field.

9 Click OK.

Using the Command-Line Interface

In the command-line interface, you use the VCS command to change archive attributes. The following table lists the archive attribute options for this command.

To change this attribute. . .	Use this command. . .
Exclusive Lock	vcs [-|+]pe
Expand Keywords	vcs [-|+]pk
Comment Prefix	`vcs -ec`*`string`*
Write Protect	vcs [-|+]pw
Store Deltas	vcs [-|+]pg
Newline Character	`vcs -en`*`string`*
Translate EOL	vcs [-|+]pt
Owner	`vcs -o`*`user_id`*
Access List	`vcs -a[`*`user_id`*`|`*`group`*`[,`*`user_id`*`|`*`group`*`...]]`
Record Length	`vcs -xrecordlength=`*`record_length`*

To change this attribute. . .	Use this command. . .
Column Masking	`vcs -xcolumnmask="`*start-end*[(numeric)]..."
Renumbering	vcs -xrenumber=*start-end* [from *start* by *number*]

See the *Serena PVCS Version Manager Command-Line Reference Guide* for complete information about the VCS command.

Moving Archives

By default, when you create a project database or project, Version Manager creates an archive location beneath the project database location. You can specify a different location when you create the database or project. If you decide the archive location that you specified when you created the project database does not suit your needs, you can change the location. For ease of use, it is important that you change the archive location before you populate the location with archive files.

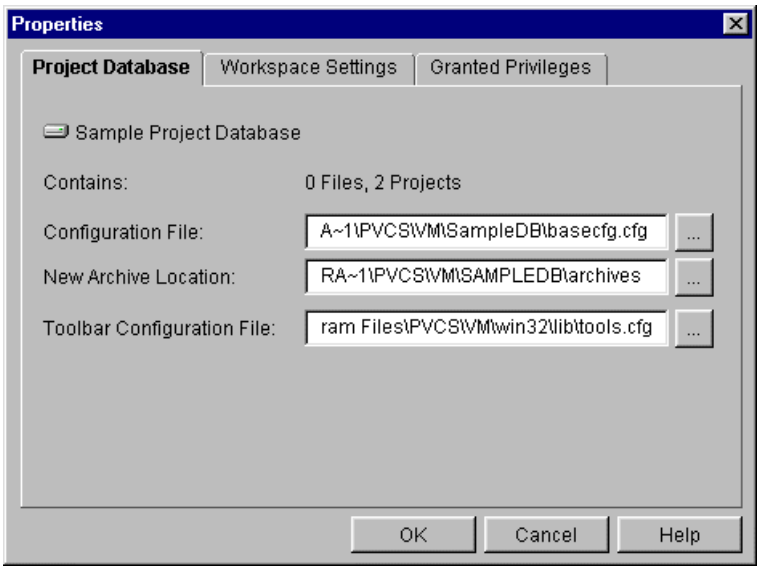
If you have reason to move archives after the archive files are in place, you must import the archives from the new location into the project to restore the reference between the versioned files and the archives.

Changing the Archive Location

This procedure assumes that you have created a project database and specified an archive location, and then, you changed your mind about where you want the archives stored before you populated the location with archive files (no archive files are in the archive location).

To change the archive location:

1. Select the project database or project.
2. Select File | Properties. The Properties dialog box appears.



3 In the **New Archive Location** field, enter the path of the new location.

4 Click OK.

Importing Archives

This procedure assumes that you have moved the archives of a project to a new location. Before you import the archives, you must delete the existing versioned files from the project. To avoid name conflicts, Version Manager will not let you import an archive if a versioned file of the same name already exists in the project. When you import archives from the new location, the archives' versioned files will be placed in the project to restore the reference between the versioned files and the archives.

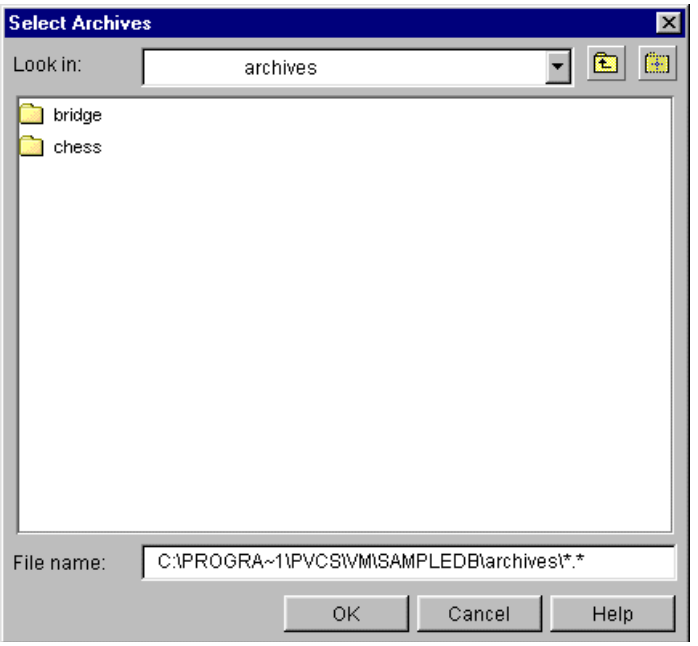
To import archives:

1 Select the versioned files whose archives you have moved. To select all of them, use Edit | Select All.

2 Select File | Delete.

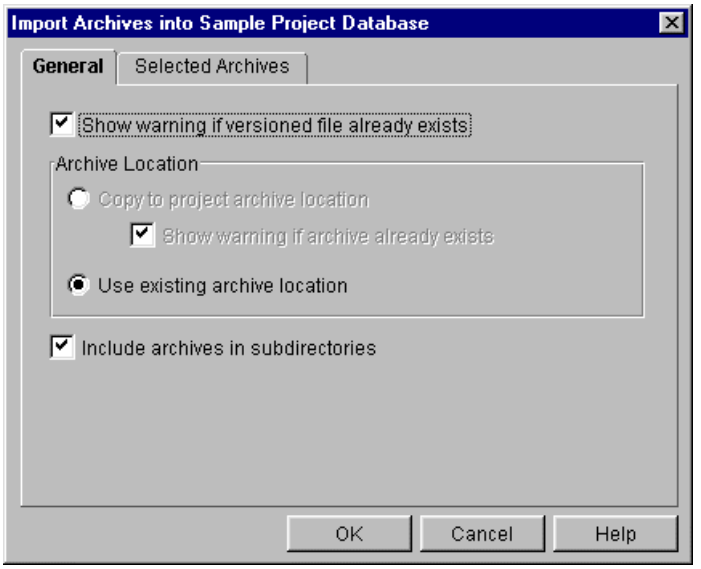
3 Select the project to which you want to import archives.

4 Select Admin | Import Archives. The Select Archives dialog box appears. Note that this dialog box opens to the archive location of the selected project.



5 Navigate to the new location of the archives (the location where you moved the archives). Then, select the entire folder that contains the archives or select only the archives that you want to import, and click OK. The Import Archives dialog box appears with the General tab active.

NOTE In the **File name** field, 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.



6 Select **Use existing archive location** from the **Archive Location** group. This choice changes the archive location for the project to the new archive location.

7 If you selected an entire folder, you can add all archives of all subdirectories included in the archive structure by selecting the **Include archives in subdirectories** check box.

8 To review the selected archives before importing them, click the Selected Archives tab.

9 Click OK.

Copying Projects and Project Databases

You can make copies of projects and project databases. You might want to do this as a shortcut to developing a new project database or as part of a migration to a new system.

NOTE For information on moving existing non file server project databases to a Version Manager file server, see "Adding Project Databases to a Version Manager File Server" on page 148.

Copying Projects

When you copy a project, Version Manager lets you:

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

- Create a new access control database, copy the existing access control database to the new project location, or use the existing access control database

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

Using the System Defaults

By default, when you copy a project, Version Manager:

- References the original project's archives in the existing archive locations. Copied subprojects will also reference their existing archive locations. When copying a project that contains multiple subprojects, remember that the subprojects may have different archive locations. If you copy a project and use the existing archive locations, the new project will reference all of the different archive locations.

NOTE Any *new* archives added to the project after it is copied will use the new project's archive location. The default is a directory named for the project beneath the archives directory of the project database location.

- Uses the project's existing configuration files.
- Uses the project's existing access control databases.
- Includes subprojects in copy project operations.
- Uses a directory named work beneath the project database location as the workfile location. When you check out versioned files from a project, Version Manager creates a directory named for the project beneath the work directory and uses it as the new project's default workfile location. (You can change the default workfile location after you copy a project by selecting File | Properties, then clicking the Workspace Settings tab and modifying the **Workfile Location** field.)

You can override any of these defaults when copying a project. The following procedure explains how.

To copy projects:

1. In the Project pane, select the project you want to copy.
2. Select Edit | Copy. The Copy Project dialog box appears.
3. Select a destination project database or project and click **Next**. You can copy a project to an open project database or project. A second Copy Project dialog box appears.

4 Accept the defaults and click **Next**, or override the defaults by editing any of the fields below:

To...	Do this...
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 will mirror the hierarchy of the target project database within the archives directory.)	Select the **Copy archives to project location** option.
Allow the selection of a specified revision as the base revision (starting point) for the new archives	Select the **Keep specified revision and discard change history** check box. Note that this field is available only if **Copy archives to project location** is selected.
Select the revision that Version Manager will use as the initial revision (or baseline)	Enter a version label or promotion group in the **Default Revision** field to select an initial revision, or click the Browse button to select one.
Include subprojects and their files	Select the **Include subprojects** check box.

5 On the third Copy Project dialog box, accept the defaults and click **Finish**, or override the defaults by editing any of the fields below:

To...	Do this...
Use the existing configuration file without copying the file to the new project location	Select the **Use existing Configuration Files** option. The copied project shares the configuration file with the original project.
Copy the existing configuration files to the new project	Select the **Copy Existing Configuration Files** to copy all existing configuration files to the new project location.
Create a new default configuration file for the new project	Select the **Create a new Configuration File** option. Version Manager creates a default configuration file by copying the default.cfg file and places it in the archives directory beneath the project database location.

To...	Do this...
Copy all existing access control databases to the new project	Select the **Copy Access Control Database to new location** option. The project's access control database is placed in the archives directory beneath the project database location. Project access control databases (if any) are placed in directories named for the projects beneath the archives directory. **NOTE** This option is not available if you opted to use the existing configuration files.
Create a new access control database for the new project	Select the **Create a new Access Control Database** option. Version Manager creates a default access control database by copying the default.db file. This new access control database is disabled in the configuration file. By default, the new access control database file is located in the archives directory. **NOTE** This option is not available if you opted to use the existing configuration files.

6 Click the **Finish** button

Copying Project Databases

Copying a Version Manager project database copies the project database to a new project database. When you copy a project database, Version Manager lets you:

- Copy all of its projects and subprojects or only copy the projects
- Copy the archives to the new archive location or use the existing archive location
- Create new configuration files, copy existing configuration files to the new project database, or use the existing configuration files
- Create a new access control database, copy the access control database to the new project database, or use the existing access control database
- Retain associations with shared archives
- Copy workspaces
- Copy the toolbar configuration file to the new project database location

Using the System Defaults

By default, when you copy a project database, Version Manager:

- References the original project database's archives in the existing archive locations. Copied subprojects will also reference their existing archive locations. When copying a project database that contains multiple subprojects, remember that the subprojects may have different archive locations. If you copy a project database and use the

existing archive locations, the new project database will reference all of the different archive locations.

NOTE Any new archives added to the project database after it is copied will use the project database's archive location. The default is an archives directory of the project database location.

- Uses the project databases's existing configuration files. Copied subprojects will also use existing configuration files (if any).
- Uses the project database's existing access control database. Copied subprojects will also use existing access control databases (if any).
- Copies workspaces.

You can override any of these defaults when copying a project database using the procedure below.

To copy project databases:

1 In the Project pane, select the project database you want to copy.

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

3 Specify a name for the project database in the **Name** field. The name cannot begin or end with a tab or blank space. Any character can be used in the name.

4 In the **Location** field, enter a location for the project database or click the Browse button to select a location. This location must be accessible to all users who will be working with the project database.

5 If you want to change the default location of the archives, enter a location in the **Archive Location** field or click the Browse button to select a location. The default archive location is a directory named archives beneath the project database location. *New* archives for the project database will be stored in this directory.

NOTE Existing archives for the project database will not be copied into this directory unless you select **Copy archives to the project location** on the next screen.

6 In the **Workfile Location** field, enter the workfile location for the project database or click the Browse button to select a location. We recommend that you specify a network drive that is available to most users. Remember that users can create a private workspace to change the workfile location to a local drive if they want.

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

8 Accept the defaults and click **Next**, or override the defaults by editing any of the fields below:

To...	Do this...
Copy the archives to the archives directory beneath the project database location (or the directory you specified)	Select the **Copy archives to project location** option.
Allow the selection of a specified revision as the base revision (starting point) for the new archives	Select the **Keep specified revision and discard change history** check box. **NOTE** This field is available only if **Copy archives to project location** is selected.
Select a revision that Version Manager will use as the initial revision (or baseline)	Enter a version label or promotion group in the **Default Revision** field to select an initial revision, or click the Browse button to select one.
Retain associations with shared archives (rather than making new independent copies of the archives)	Select the **Retain shared archives** option. **NOTE** Available only if the **Copy archives to project location** option is selected. **NOTE** Archives are shared, not projects. If you eventually add a new file to the project, it will not be shared with any other project until you copy the versioned file to the desired project or projects.
Include subprojects and their files	Select the **Include subprojects** check box.

9 On the third Copy Project Database dialog box, accept the defaults and click **Finish**, or override the defaults by editing any of the fields below:

To...	Do this...
Use the existing configuration file without copying the file to the new project database	Select the **Use existing Configuration Files** option. The copied project database shares the configuration file with the original project database.
Copy the existing configuration files to the new project database	Select the **Copy Existing Configuration Files** option.
Create a new default configuration file for the new project database and for any copied projects that had a configuration file in the original project database	Select the **Create a new configuration file** option. Version Manager creates a default configuration file by copying the default.cfg file and places it in the archives directory beneath the project database you are copying.

To...	Do this...
Copy all existing access control databases to the new project database	Select the **Copy Access Control Database to new location** option. The project database's access control database is placed in the archives directory of the project database you are copying. Project access control databases (if any) are placed in directories named for the projects beneath the archives directory. **NOTE** This option is not available if you opted to use the existing configuration files.
Create a new access control database for the new project database	Select the **Create a new Access Control Database** option. Version Manager creates a default access control database by copying the default.db file. This new access control database is disabled in the configuration file. By default, the new access control database file is located in the archives directory. **NOTE** This option is not available if you opted to use the existing configuration files.
Copy existing workspace definitions to the new project database	Select the **Copy Workspaces** option.

10 Click the **Finish** button.

Chapter 6

Using Security

Introduction

In Serena PVCS Version Manager, you can control who can access projects and archives and what actions the authorized users can perform on the projects and archives. Security is defined in Version Manager by using the following features:

- Access control databases
- Access lists
- Privileges

Access control databases define who can access projects—and thereby, the archives of the project. An access list further controls access by defining who can access a single archive of a project. Privileges define the actions users can perform on the projects and archives.

All of these features are discussed in depth in this chapter. Also, this chapter provides information about planning, setting up, and maintaining security.

For information about the privileges you should set to protect Version Manager program files and project data on UNIX and Windows, see the *Serena PVCS Version Manager Installation Guide*.

For information on configuring Version Manager File Server security, see the following table:

For information on . . .	See . . .
Configuring file server security options	"Configuring the File Server" on page 141
File server security considerations	"Security Considerations" on page 165
Path map security options	"Configuring Path Map Security Options" on page 140

About Access Control Databases

An access control database defines the users who are authorized to perform actions on projects and defines the actions that users can perform. In Version Manager, the actions are associated with Version Manager privileges.

An access control database is an encrypted file.

Users

When you define an access control database, part of the definition is a user ID for each authorized user. When you are using an access control database for security, if the user ID that Version Manager obtains is not defined in the access control database for a project database or project, then that user cannot access the project database or project—meaning that the user cannot access the archives. Version Manager uses login sources to obtain user IDs. Refer to "Login Sources" on page 73 for an explanation of login sources.

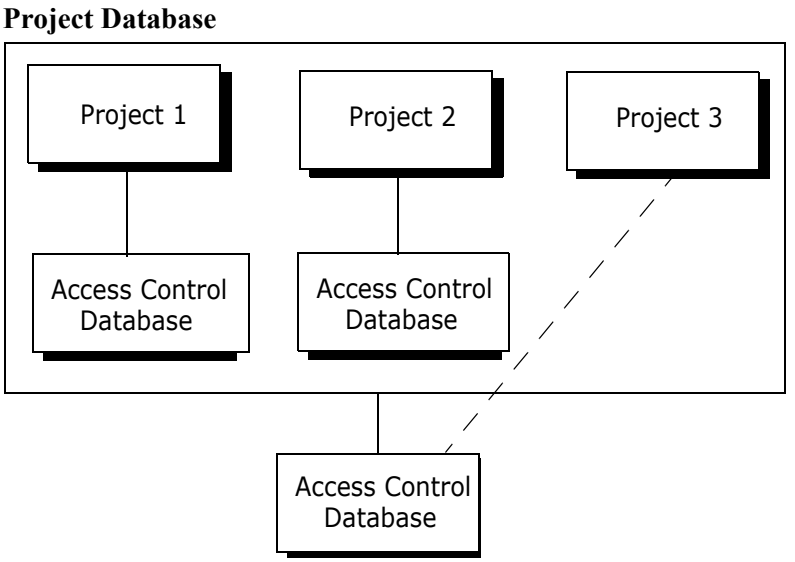
For desktop client users: When Version Manager obtains a user ID, the program can check the access control database to see if the user ID exists there. If the user ID does not exist, you can configure Version Manager to automatically create the user ID in the

access control database and assign default privileges to the user. Refer to "Login Sources" on page 73 for a discussion of how to set up Version Manager to automatically create users in the access control database.

In the Desktop Client

In the desktop client, access control databases are associated with project databases and projects. The desktop client uses an access control database if it is specified and enabled in the configuration file associated with the project database or project or if it is embedded into Version Manager. By default, when you create a project database, an access control database is created and specified but not enabled in the master configuration file that is also created.

A project database can have one access control database associated with it, and each project within the project database can have an access control database associated with it. If you have an access control database defined for your project database and not for the projects within the project database, the projects use the access control database associated with the project database. Also, you can define an access control database for some of the projects in a project database and not others. For example, if you have three projects in a project database and you have defined an access control database for two of the three projects, those two projects will use their access control database and the third will use the project database's access control database (see the following figure).



If you define an access control database for a project database, you have the same access control for all the projects in the project database. By defining an access control database for each project, you can further restrict the control for each project. And, by using access lists, you can control access for each individual archive of a project. Access lists are discussed in the next section.

The Default Access Control Database for Project Databases

By default, Version Manager creates a default access control database for each project database that is created. This access control database contains the user ID of the person who created the project database with the SuperUser privilege set assigned to that user, and a few default privilege sets (see Table 6-3, "Default Privilege Sets," on page 244). This access control database is specified but not enabled in the project database's

configuration file, meaning that the project database does not use the access control database for security.

Version Manager creates this default access control database by copying the default.db file. By default, this file is placed in the following location during Version Manager installation:

- In Windows: *drive*:\Program Files\Serena\vm\common\pvcsprop\pvcs\vm
- On UNIX: /usr/serena/vm/common/pvcsprop/pvcs/vm

You can modify the settings in the default.db file to suit your needs, and then Version Manager will use the modified file to create new access control databases.

In the default access control database for a project database, you define the users who will have the right to access the project database and the actions that they will be allowed to perform. You can define user IDs, passwords, user expiration dates, groups of users, and privileges assigned to the users in access control databases. Only user IDs are required. If you do not assign privileges to the users, Version Manager assigns them the Unlimited privilege set (discussed later).

Embedding an Access Control Database

You can embed one access control database into Version Manager; this access control database will control security for all project databases. Embedding an access control database into Version Manager ensures that all users will be using the same security definition and that users cannot use a different access control database.

Version Manager cannot automatically create users for an access control database that is embedded into Version Manager (see "Users" on page 204).

An access control database that is embedded into Version Manager affects all desktop client and command-line users who are using the copy of Version Manager that has the file embedded in it. For instructions on how to embed an access control database, see "Embedding an Access Control Database into Version Manager" on page 228.

In the Command-Line Interface

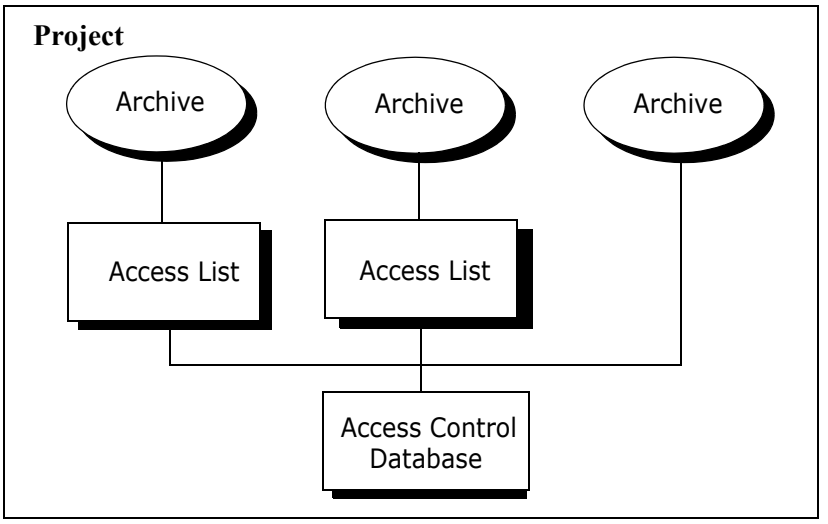
The command-line interface uses an access control database if it is specified in a configuration file used by the command-line interface or if it is embedded into Version Manager. For instructions on how to embed an access control database, see "Embedding an Access Control Database into Version Manager" on page 228.

Unlike the desktop client, the command-line interface cannot automatically create users in the access control database.

About Access Lists

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. You cannot specify a user in an access list that is not defined in the access control database. If an archive does not have an access list associated with it, the access control database controls access to the archive (see the following figure).



Access lists make a very secure system, but they can be time consuming to maintain if you define individual users in the access list instead of groups of users. Therefore, we recommend that you specify groups in access lists rather than individual users.

Access List Groups

Defining access list groups is a method of controlling archive access collectively. Members of a group should have the same project responsibilities and need the same privileges.

Group definitions are not used for security unless you define an access list with the group specified. To use groups in access lists, you must first define the groups and assign the privileges to the groups in the access control database.

The benefit to using groups is that maintenance of access lists is easier. For example, if you create a group that defines the users and privileges for Development and then a new developer joins the company, you simply add the new employee as a user in the group. Then, the new user will have access to all of the archives to which that group has access. Otherwise, for the new developer to have access to all of the correct archives, you would have to add the new user to the access list of each archive.

Three important rules to remember about access list groups are:

- Group definitions are not used unless the groups are specified in an access list for an archive.
- The privileges assigned to a group can only restrict a user's privileges (see "Rules for Privileges" on page 209).
- Access list groups cannot be empty. The definition of an access list group must contain users and/or other access list groups.

About Privileges

A privilege enables a Version Manager action. Privileges are specified in the access control database to grant users permission to use the action. For example, GetTip is the privilege of letting users check out tip revisions from archives.

Version Manager has three types of privileges:

- Base privileges
- Composite privileges
- Privilege sets

You can use the privileges that are built into Version Manager or define custom privilege sets to suit your needs.

Privileges are assigned in an access control database to either individual users or groups of users (access list groups).

In the desktop client, privileges are grouped according to function:

- Action
- Archive
- Project

NOTE The one major action that does not have a privilege associated with it is Create Project Database. To restrict users from creating project databases, you must define a password that users must enter before the action can take place. Then, you withhold the password from users that are not allowed to create project databases. See "Restricting the Creation of Project Databases" on page 230.

Base Privileges

A base privilege is a single Version Manager action. For example, the AddVersion privilege lets users assign version labels to revisions. See Table 6-1, "Base Privileges," on page 240 for a list of base privileges.

Composite Privileges

A composite privilege is composed of two or more base privileges. For example, the Get composite privilege is composed of both the GetNonTip and GetTip base privileges. This privilege lets users check out both tip and nontip revisions from archives. See Table 6-2, "Composite Privileges," on page 243 for a list of composite privileges.

Privilege Sets

A privilege set can be composed of base privileges, composite privileges, and other privilege sets. Version Manager has several default privilege sets that are defined in Table 6-3, "Default Privilege Sets," on page 244.

Version Manager provides the flexibility of allowing you to define a custom privilege set so that you can create privilege sets that are suited to your working environment. For

example, you could create a custom privilege set named BasicUser from the composite privileges Get and Put, and the base privilege InitArchive. Then, users with the BasicUser custom privilege set assigned to them could check in and out revisions as well as create archives.

You can use existing privilege sets (default and custom) as part of the definition of new custom privilege sets.

Rules for Privileges

Version Manager privileges are governed by the following rules:

- By default, users have the Unlimited privilege. And, if a user has been assigned the Unlimited privilege in the access control database, access rights to archives are limited only by access lists.
- If a user has been assigned the SuperUser privilege in the access control database, archive access is not restricted regardless of whether the user exists in the access lists for the archives.
- Privileges assigned to an access list group can only limit the privileges of the users of the group, not increase the privileges of the users of the group. For example, JohnD has the Unlimited privilege set assigned to him and belongs to the Developer group that has the privilege set Dev assigned to it. The Dev privilege set has the rights of lock, unlock, get, and put. An archive that JohnD wants to access has the Developer group on the access list and not JohnD as an individual user. In this case, JohnD is limited to the privileges lock, unlock get, and put; all of the other privileges that were assigned to JohnD are not available to him when accessing the archive because the Developer group is on the archive's access list and he is not.

  If on the other hand, both the Developer group and JohnD were on the archive's access list, JohnD would have his Unlimited privileges available to him.

  Conversely, JohnD is a user with the Dev privilege set assigned to him and belongs to the Developer group that has the Unlimited privilege set assigned to it. An archive that JohnD wants to access has only the Developer group on its access list. In this case, JohnD is limited to the Dev privileges (lock, unlock, get, and put). The Unlimited privilege set that was assigned to the Developer group does not get assigned to JohnD because he did not have them assigned to him as an individual user.
- Access list group definitions can include other groups as well as individual users. If an individual user is a member of more than one group and each of the groups the user belongs to is part of the definition of an access list group, the user has the union of the privileges assigned to each group.
- The SuperUser privilege cannot be assigned to a group—only to individual users.

Planning Security

The following is a list of questions that will help you plan security:

- What level of security do you need?
  - Can you use the same security definition for each project in a project database? If so, then you can define an access control database for the project database—none

for the projects within the project database and no access lists. And, you can assign each user the same privileges. (See Example 1 below).

- Can you use the same security definition for each project in a project database? If so, then you can define an access control database for the project database—none for the projects and no access lists. And, if the users have different roles that they perform on the archives, you can define privilege sets for each user role (or use the default privilege sets) and assign the appropriate privilege set to each user. (See Example 2 below).
- Do you need a different security definition for the archives in each individual project in a project database? If so, then you can define an access control database for each project but not define any access lists.
- Do you need a different security definition for some archives of a project database? If so, then you must define an access list for those archives. This provides the greatest level of security. (See Example 3 below).

- Should you define access list groups to control archive access collectively? You should do this only if you are using access lists. Use access list groups when defining access lists (see "About Access Lists" on page 206).

Examples

This section provides examples for you to model when setting up security.

Example 1

You want to define security for a project database so that users outside the project database are prevented from accessing its archives. All users have the same set of privileges.

Access control database: You define all of the users who require access to the project database and define a custom privilege set or select a default privilege set for the users and assign it to each user.

Access list: None—therefore, no access list groups are needed.

Example 2

You want to define security for a project database so that users outside the project database are prevented from accessing its archives. In this case, you want to assign role-related privileges. For example, developers will have one set of privileges assigned to them, QA engineers another, Project Leaders another, and so forth.

Access control database: You define all of the users who require access to the project database and define a custom privilege set or select a default privilege set for each role-related group of users and assign the appropriate privilege set to each user. Version Manager's default privilege sets are tailored to role-related groups within a company (see Table 6-3, "Default Privilege Sets," on page 244).

Access list: None—therefore, access list groups are not needed.

Example 3

This example builds on Example 2. You need to secure individual archive files that contain sensitive files. For example, most organizations only want select users to access payroll

information. In this example, you want to assign role-related privileges as well as assign an archive list to the sensitive archives.

Access control database: You define all of the users who require access to the project database and define a custom privilege set or select a default privilege set for each role-related group of users and assign the appropriate privilege set to each user. Version Manager's default privilege sets are tailored to role-related groups within a company (see Table 6-3, "Default Privilege Sets," on page 244).

Also, you define the access list group that contains the individual users who will be permitted to access the restricted archives and assign the appropriate privileges or privilege set to the group. This access list group definition can only include individual users who are defined in the access control database.

Access list: One that is assigned to the sensitive archives. The access list group you defined in the access control database will be the definition of the access list.

Setting Up Security

This section provides step-by-step procedures for setting up security using the Version Manager desktop client and the command-line interface.

Using the Desktop Client

The following basic steps provide an overview for setting up security. Following this list are sections that provide detailed procedures.

1. If you want to set up security for a project, you must associate a project configuration file with the project, unless this has already been done. Then, you must specify the location of an access control database or create a new one for the project.
2. If you are setting up security for a project database and you want to use an access control database other than the one that was created with the project database, specify the location of an access control database or create a new one.
3. Define custom privilege sets for the access control database.
4. Define users, which includes assigning privileges to the users.
5. If you are planning to use access lists as part of your security definition, define the appropriate access list groups, which includes assigning privileges and users to the groups.
6. If you are using access lists, define access lists for the appropriate archives.
7. Enable security.

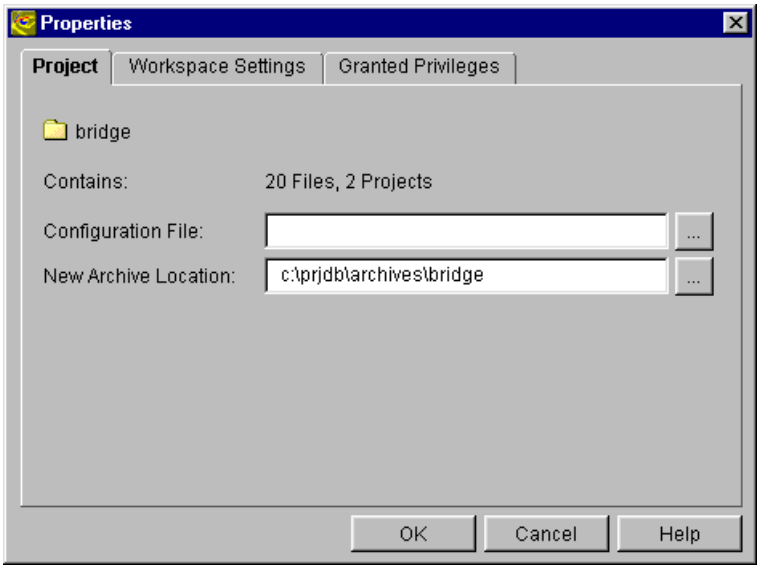
Associating a Configuration File with a Project

Projects, by default, are not associated with a project configuration file.

To associate a project configuration file with a project:

1. Select the project you want to associate with a configuration file.

2 Select File | Properties. The Properties dialog box appears with the Project tab active.



3 In the **Configuration File** field, enter the location and name of the configuration file to be associated with the project. If the configuration file does not already exist, Version Manager creates it automatically, using the name and location you specify here. The default name is project.cfg. The configuration file has no options set.

4 Click OK.

Specifying the Access Control Database for Your Project Database or Project

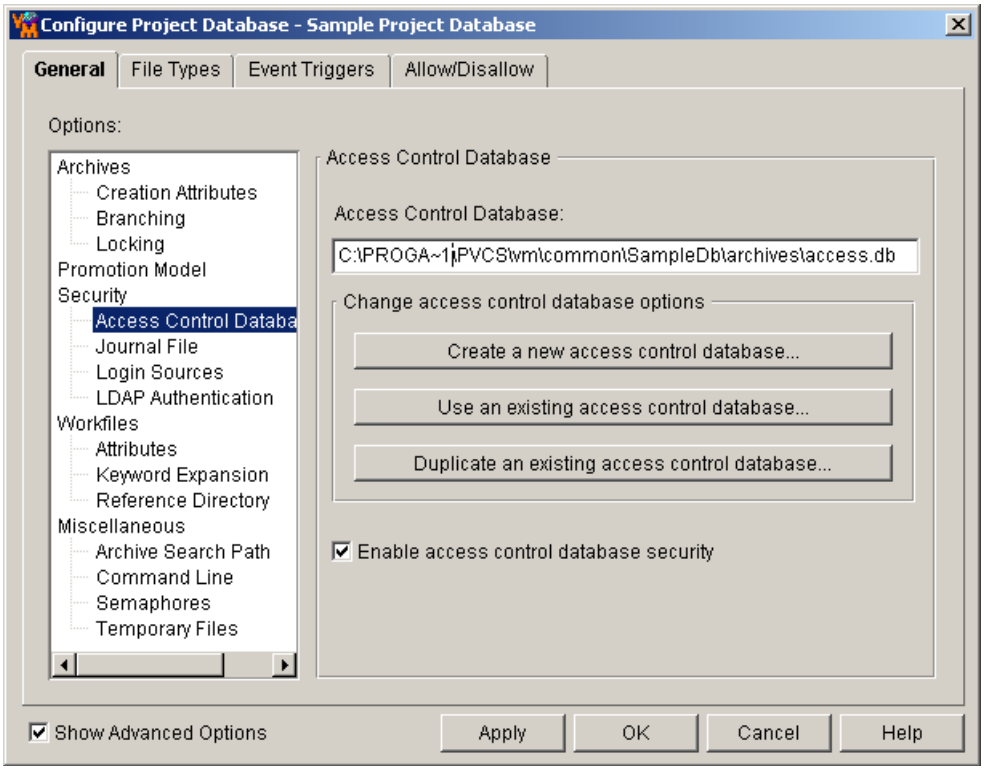
To specify the access control database location:

1 Select the project database or project for which you want to define an access control database. The project database or project must have a configuration file associated with it.

2 Select Admin | Configure Project. The Project Configuration dialog box appears with the General tab active.

3 If not already selected, select the **Show Advanced Options** check box.

4 In the Options list, select **Access Control Database** beneath Security. The access control database information appears in the Access Control Database pane (the right pane).



5 Do one of the following:

- Click the **Create a new access control database** button to create a new access control database in the archive directory of the project database or project. Version Manager copies the default access control database (default.db) to this location.
- Click the **Use an existing access control database** button to use an existing access control database. The Select Access Control Database dialog box appears allowing you to choose an access control database. This option works well when you have multiple project databases that require the same security information. If you use the same access control database for each of the project databases, you only have to update one access control database when your security situation changes, for example, when you need to add a new user to the access control database.
- Click the **Duplicate an existing access control database** button to duplicate an existing access control database. The Select Access Control Database dialog box appears allowing you to choose an access control database. Version Manager makes a copy of the access control database that you choose and places it in the archives directory of the project database.

6 Clear the **Enable access database control security** check box to modify the access control database without having it enforced.

7 Click OK.

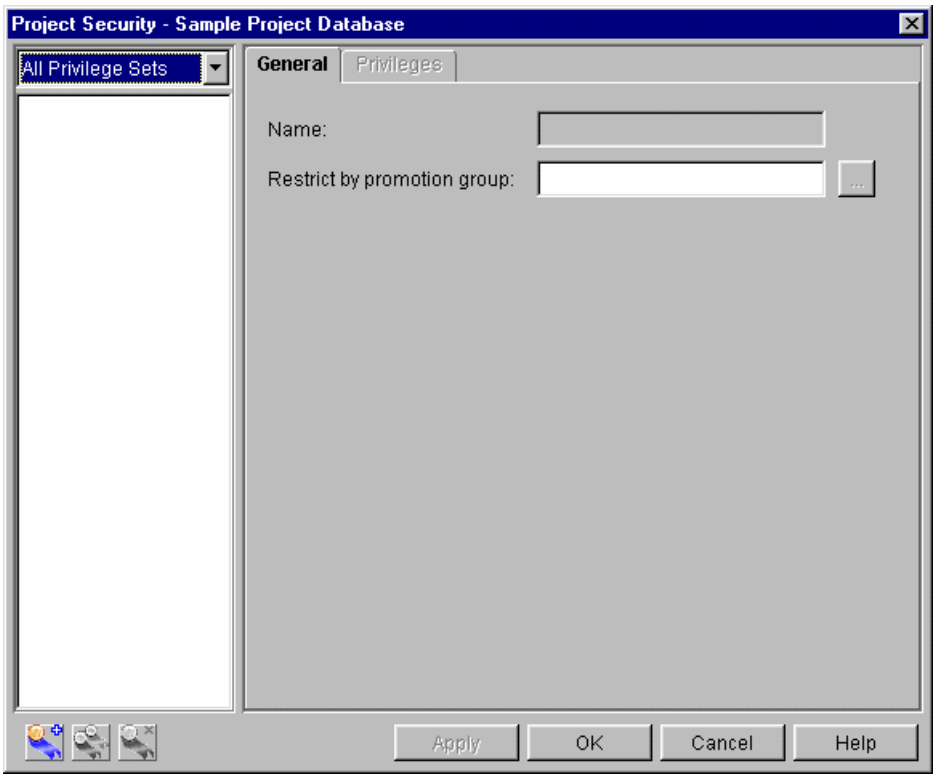
Defining Custom Privilege Sets

Define custom privilege sets for the access control database.

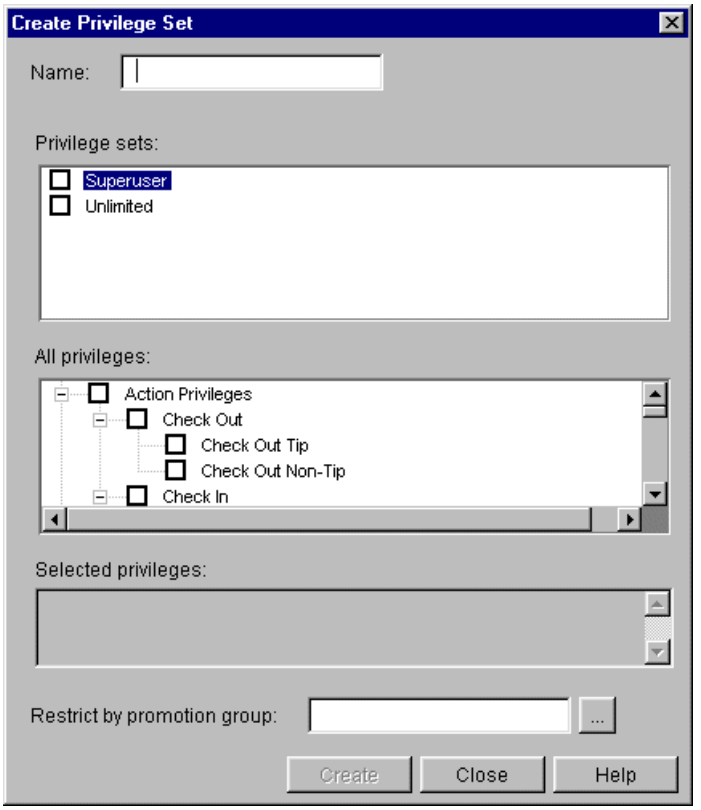
To define custom privilege sets:

1 Select the project database or project associated with the access control database you are defining.

2 Select Admin | Security | Privilege Sets. The Project Security dialog box appears with the General tab active.



3 Click the **New** button. The Create Privilege Set dialog box appears.



4 Enter a name for the custom privilege set. The name cannot begin or end with a tab or blank space. Any other character can be used in the name except left and right parentheses (), a single quote ('), a double quote ("), a colon (:), a backslash (\), and an asterisk (*).

5 Select the base privileges, composite privileges, and existing privilege sets that will define this new custom privilege set. See "Privilege Definitions" on page 239 for a list and definitions of all privileges.

The options with a + in the box beside them are allowed. The Project privileges are the only privileges that can have a – in the box beside them because they are negative privileges. The – means that the privilege is denied. If a Project privilege is denied to a user, it cannot be allowed in a privilege set that is assigned to the user. For example, if the Create Project privilege is denied as a base privilege to a user but is allowed in a privilege set that is assigned to the same user, the user does not have the Create Project privilege.

To change the status of an option, select the check box beside the option. You can allow or disallow an entire set of related privileges by clicking the parent option. For example, you could assign all of the Action privileges by selecting the check box beside Action Privileges to place a + in the check box.

Note that the Selected Privileges box near the bottom of the dialog box displays a cumulation of the privileges you have selected. If any of the Project privileges are allowed, they are not displayed in this box. They are only displayed in this box when they are denied because they are negative privileges.

6 In the **Restrict by promotion group** field, enter the promotion group by which this custom privilege set will be restricted. See "Restricting a User's Ability to Promote" on page 258.

7 Click Create to create this privilege set.

8 To define another privilege set, repeat Steps 4–7. Otherwise, click Close. When you click Close, the Create Privilege Set dialog box closes and the newly created privilege set is added to and selected in the All Privilege Sets list of the Project Security dialog box.

9 Continue to the next procedure, Defining Users.

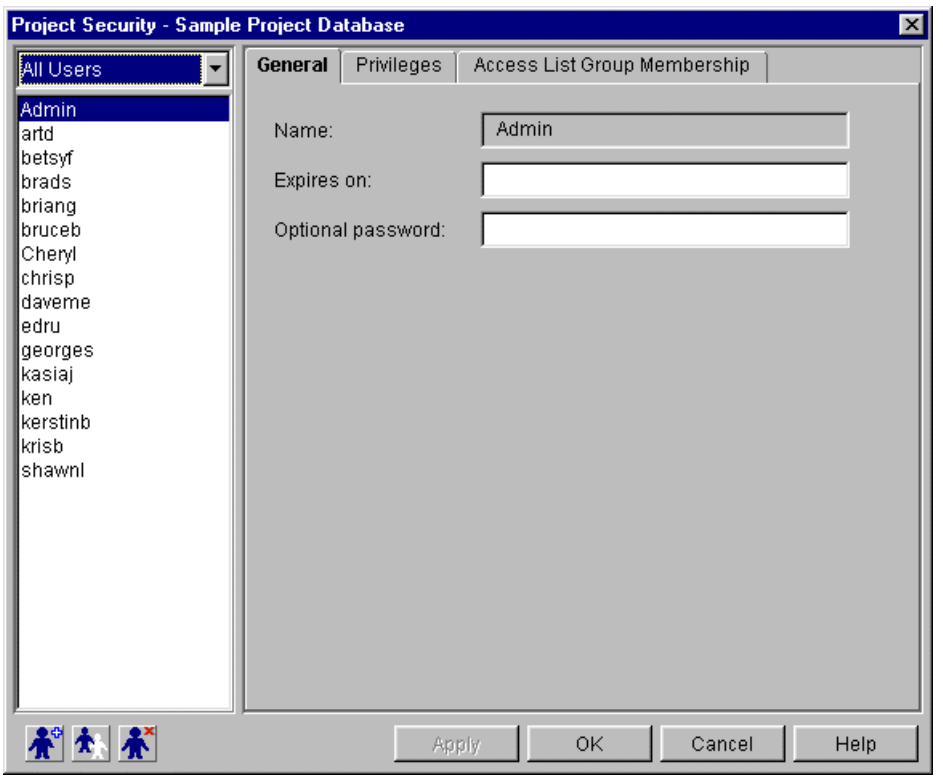
Defining Users

For desktop client users: When Version Manager obtains a user ID, the program can check the access control database to see if the user ID exists there. If the user ID does not exist, you can configure Version Manager to automatically create the user ID in the access control database and assign privileges to the user. Refer to "Login Sources" on page 73 for a discussion of how to set up Version Manager to automatically create users in the access control database.

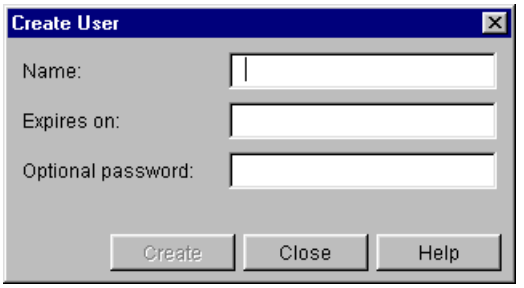
Define the users for the access control database. This includes assigning privileges to the users.

To define users in the access control database:

1 If the Project Security dialog box is already open, select **All Users** from the drop-down list in the upper-left corner. If it is not open, select Admin | Security | Users. The Project Security dialog box looks like this:



2 Click the **New** button. The Create User dialog box appears.



3 Enter the user's ID in the **Name** field. The user ID that you enter here must be the user ID that Version Manager obtains when the user is running Version Manager. See "Login Sources" on page 73 for an explanation of how Version Manager obtains user IDs.

4 Optionally, you can enter an expiration date for the user in the **Expires on** field. For example, if you are defining the user on January 2, 2003 and you enter an expiration date of December 31, 2003, the time period during which this user ID is valid is January 2 to December 31, 2003. You must enter the date and time in the formats that you have set for your operating system. In 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.

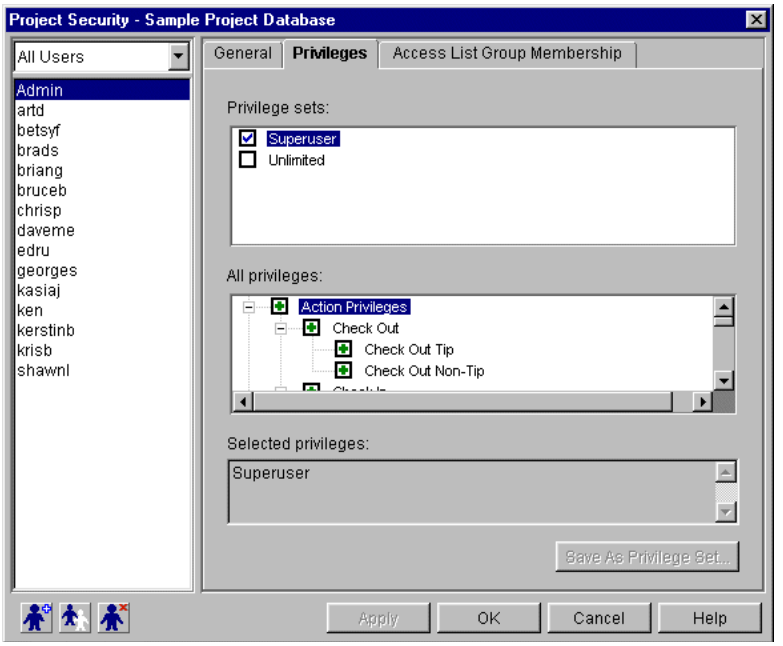
NOTE On UNIX, it is recommend that you set the PVCS_DATE_FORMAT to MM/dd/yyyy, or if you want days to lead months to dd/MM/yyyy. Notice that the month is specified in capital letters, and that the year is specified in four digits. These formats will give you the best results.

5 If you want the user to enter a password before accessing a project database or project that has an access control database enabled, specify a password for the user in the **Optional password** field. Passwords can be up to 30 characters long. If Login Dialog is specified as a login source and Version Manager uses this login source to obtain the user ID, then the user must enter his user ID and password. If any other login source is used to obtain the user IDs and a password is defined for a user, then the user is prompted only for a password, not for both the user ID and password.

Note that users can change their own passwords at any time using File | Change Password.

6 Click Create and then Close to create this user in the access control database. When you click Close, the Create User dialog box closes and the newly created user is added to and selected in the All Users list of the Project Security dialog box.

7 Click the Privileges tab.



8 In the **All Users** list, verify that the user you are defining is selected.

9 On the Privileges tab, select the privileges and/or privilege sets to assign to the user. See "Privilege Definitions" on page 239 for a list and definition of all privileges.

The options with a + in the box beside them are allowed. The Project privileges are the only privileges that can have a – in the box beside them because they are negative privileges. The – means that the privilege is denied. If a Project privilege is denied to a user, it cannot be allowed in a privilege set that is assigned to the user. For example, if the Create Project privilege is denied as a base privilege to a user but is allowed in a privilege set that is assigned to the same user, the user does not have the Create Project privilege.

To change the status of an option, click the check box beside the option. You can allow or disallow an entire set of related privileges by clicking the parent option. For example, you could assign all of the Action privileges by clicking the check box beside Action Privileges to place a + in the check box.

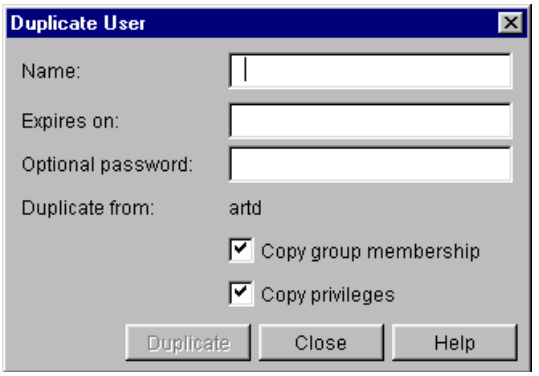
Note that the Selected Privileges box near the bottom of the dialog box displays a cumulation of the privileges you have selected. If any of the Project privileges are allowed, they are not displayed in this box. They are only displayed in this box when they are denied because they are negative privileges.

10 To save this user definition and define other users, click the Apply button and continue to Step 11. Otherwise, click OK to save this definition and return to Version Manager.

11 To create another user, do one of the following:

- Create a new user by clicking the Create User button and repeating Steps 3–10.
- Duplicate an existing user by selecting the user you want to duplicate and then clicking the Duplicate User button. The Duplicate User dialog box appears. The

duplicate function is useful when you want to define several users with the same set of privileges assigned to them.



In the Duplicate User dialog box, do the following:

a Enter the user's user ID in the **Name** field.

b Optionally, you can enter an expiration date for the user in the **Expires on** field. For example, if you are defining the user on January 2, 2003 and you enter an expiration date of December 31, 2003, the time period during which this user ID is valid is January 2 to December 31, 2003. You must enter the date and time in the formats that you have set for your operating system. In Windows, you set the date and time formats from the Control Panel | Regional Settings. On UNIX, you set 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.

c If you want the user to enter a password before accessing a project database or project that has an access control database enabled, specify a password for the user in the **Optional password** field. Passwords can be up to 30 characters long. If Login Dialog is specified as a login source and Version Manager uses this login source to obtain the user ID, then the user must enter his user ID and password. If any other login source is used to obtain the user IDs and a password is defined for a user, then the user is prompted only for a password, not for both the user ID and password.

Note that users can change their own passwords at any time using File | Change Password.

d Select the **Copy privileges** check box to assign all of the privileges defined for the original user to the new user.

e In this procedure, we have not yet defined groups; therefore, deselect the **Copy group membership** check box.

f Click Duplicate.

g To duplicate another user, repeat Steps 11a–f. Otherwise, click Close.

12 If you are planning to use access lists as part of your security definition, continue to the next procedure, "Defining Access List Groups." Otherwise, skip to "Enabling Security" on page 223.

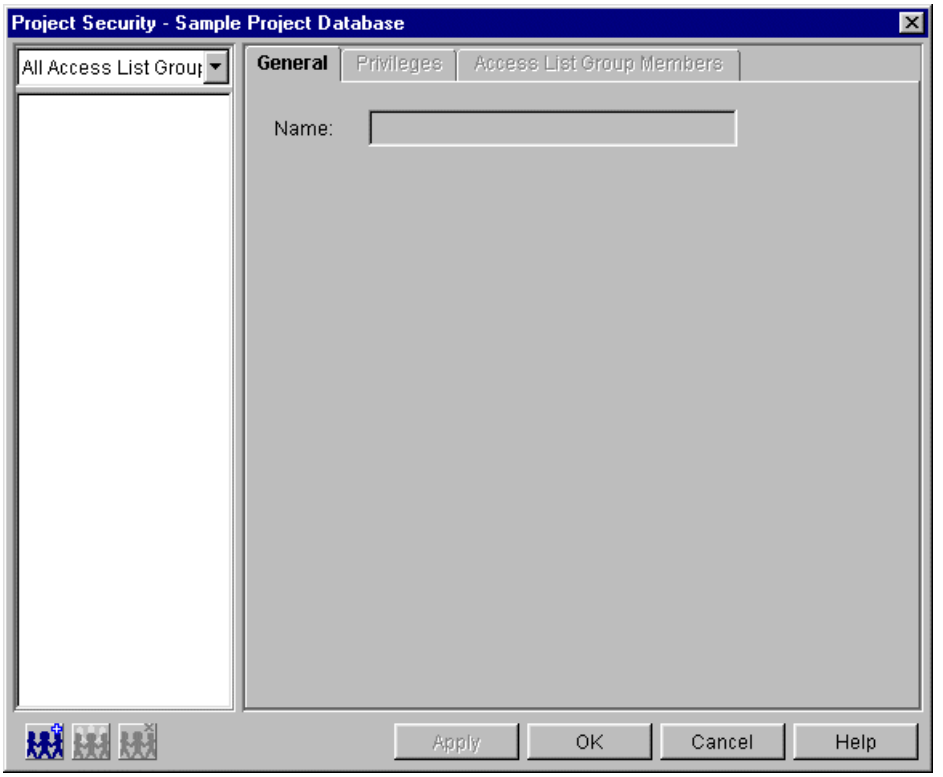
Defining Access List Groups

If you are planning to use access lists as part of your security definition, define the appropriate access list groups, which includes assigning privileges and users to the groups.

NOTE If you create an access list group, you must specify the users and/or other access list groups that define the access list group. Access list groups cannot be empty.

To define access list groups:

1 If the Project Security dialog box is already open, select All Access List Groups from the drop-down list in the upper-left corner. If it is not open, select Admin | Security | Access List Groups. The Project Security dialog box looks like this:

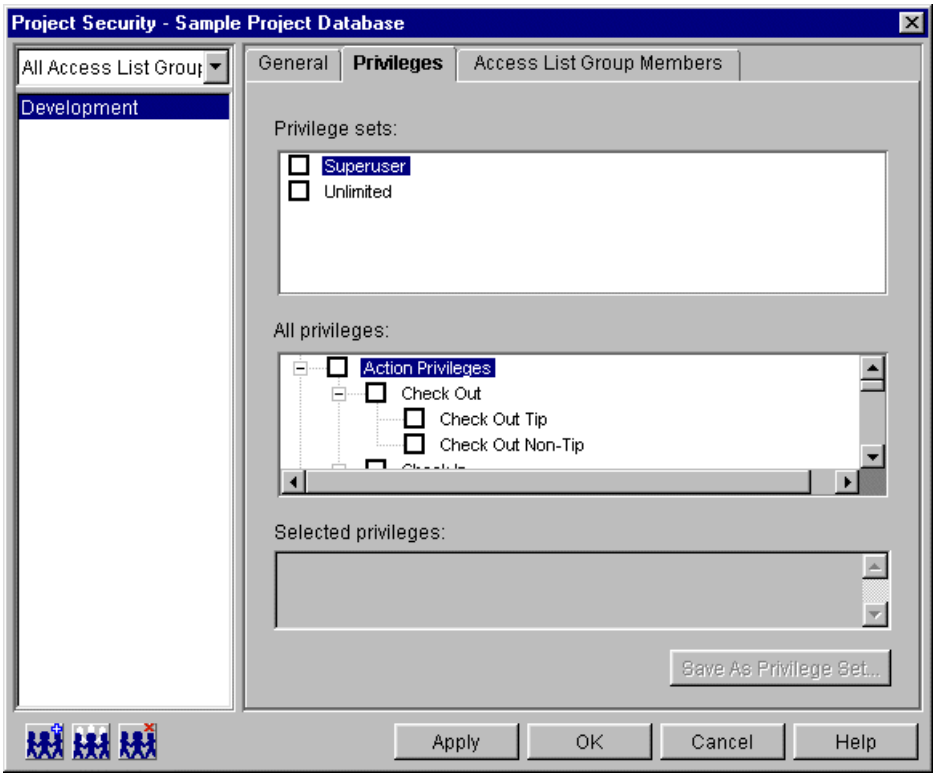


2 Click the **New** button. The Create Access List Group dialog box appears.

3 Enter a name for the group in the **Access list group name** field. The name cannot begin or end with a tab or blank space. Any other character can be used in the name except left and right parentheses (), a single quote ('), a double quote ("), a colon (:), a backslash (\), and an asterisk (*).

4 Click Create. The fields in the dialog box are cleared so that you can create another group. When you are done creating groups, click Close. The Create Access List Group dialog box closes and the newly created group is added to and selected in the All Access List Groups list of the Project Security dialog box.

5 Click the Privileges tab.



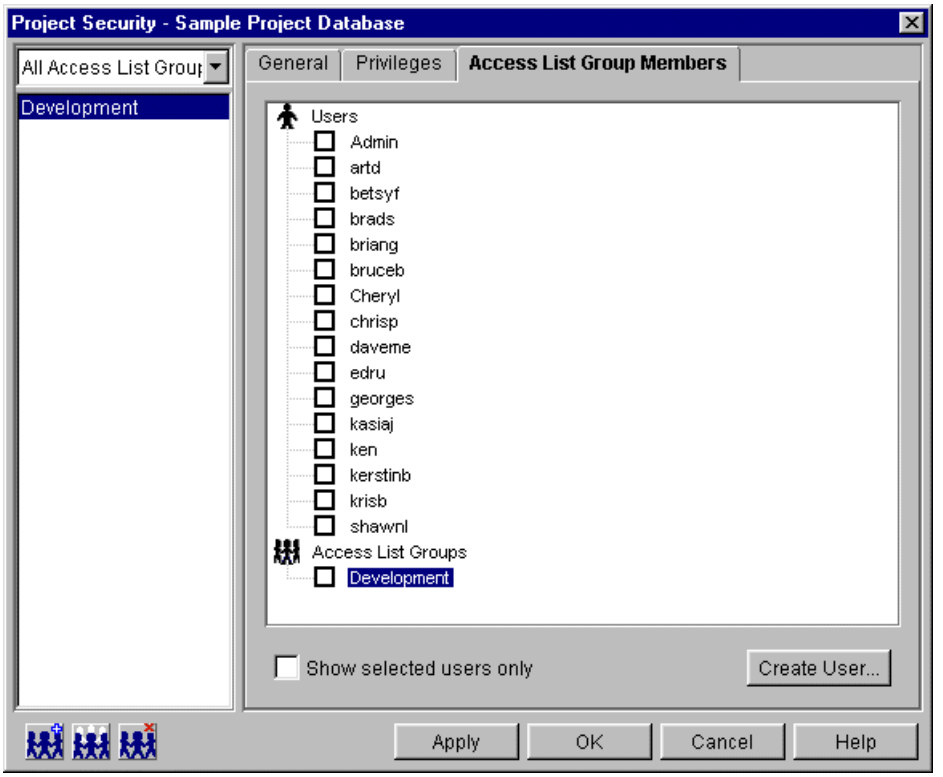
6 In the All Access List Groups list, verify that the access list group you are defining is selected.

7 On the Privileges tab, select the base privileges, composite privileges, and existing privilege sets to assign to the access list group. See "Privilege Definitions" on page 239 for a list and definition of all privileges.

The options with a + beside them are allowed. The options with a – beside them are denied; only Project privileges are negative privileges, meaning if they are not selected, then they are allowed privileges. To change the status of an option, select the check box beside the option. You can allow or disallow an entire set of related privileges by clicking the parent option. For example, you could assign all of the Action privileges by selecting the check box beside Action Privileges to place a + in the check box.

Note that the Selected Privileges box near the bottom of the dialog box displays a cumulation of the privileges you have selected. If any of the Project privileges are allowed, they are not displayed in this box. They are only displayed in this box when they are denied because they are negative privileges.

8 Click the Access List Group Members tab. This tab displays all of the users and existing groups that are defined in the access control database.



9 In the All Access List Groups list, verify that the access list group you are defining is selected.

10 Select the check boxes next to the users and/or access list groups that you want as members of the group. This is a required step. Access list groups cannot be empty.

NOTE The desktop client allows you to apply an empty access list group; however, do not leave them empty.

11 Click **Apply**.

12 To create another access list group, repeat Steps 2–11. When you are finished defining access list groups, click OK.

Defining Access Lists

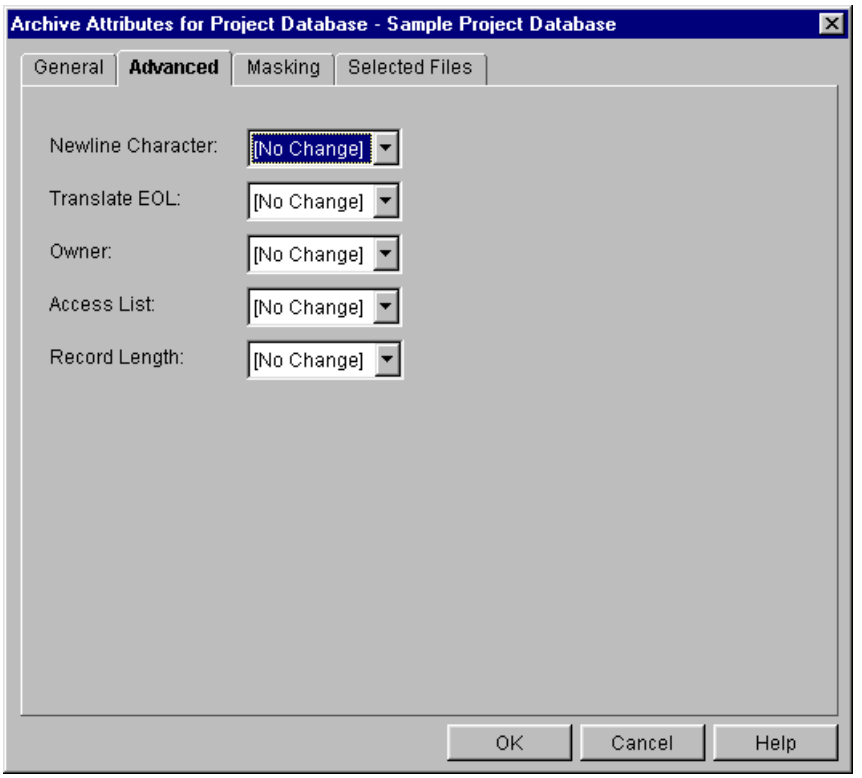
If you are using access lists, define an access list for the appropriate archives. An access list is an attribute of an archive. You can define access lists for newly created archives and for existing archives. For information about how to define access lists for newly created archives, see "Archive Creation Options" on page 67.

To define an access list for an existing archive:

1 Select the project or individual versioned file associated with the archive for which you want to define an access list. When you select a project, you define the same access list for all of the archives of the project.

2 Select Admin | Archive Attributes. The Archive Attributes dialog box appears with the General tab active.

3 Click the Advanced tab.



4 In the **Access List** field, select Modify and then enter the access list groups and individual users who can access the archive in the text field that appears to the right of the field. Each entry in the list must be separated by a comma (,). Or, you can click the Browse button and select the groups or individual users. The combined value of this field cannot exceed 254 characters in length.

Remember that the access list groups and users who you specify must be defined in the access control database.

5 Click OK.

Enabling Security

Now that you have defined an access control database, you must tell Version Manager to use it to control security for your project database or project.

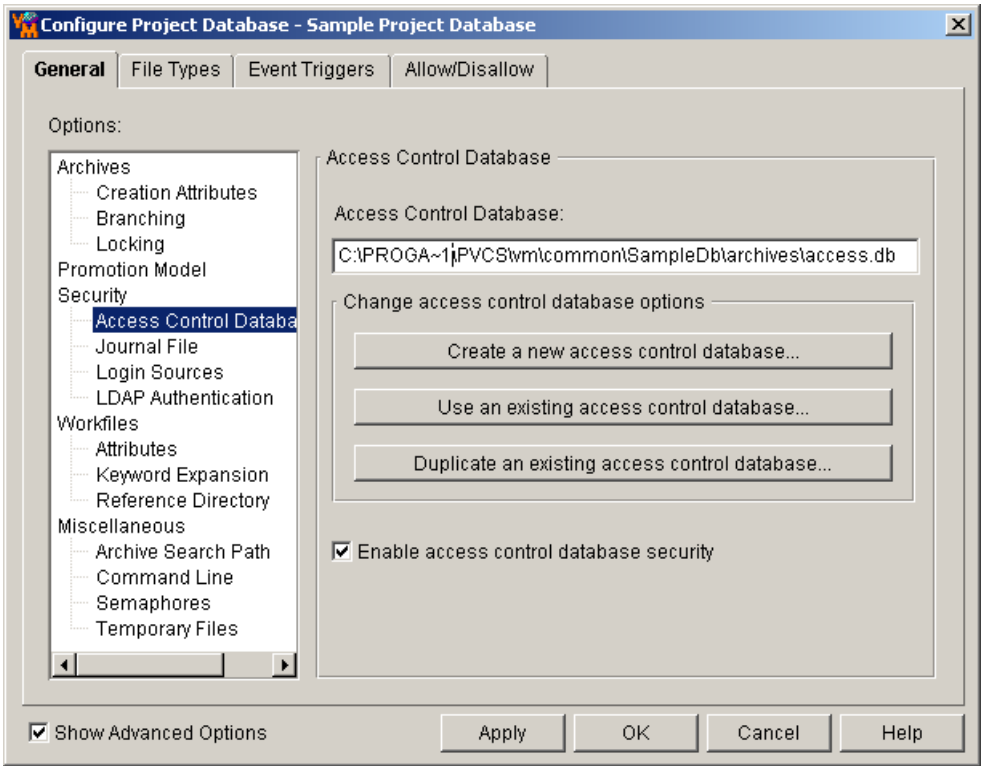
To enable the access control database for security:

1 Select the project database or project for which this access control database will control security.

2 Select Admin | Configure Project. The Configure Project dialog box appears with the General tab active.

3 If not already selected, select the **Show Advanced Options** check box.

4 In the Options list, select **Access Control Database** beneath Security. The Access Control Database pane appears on the right.

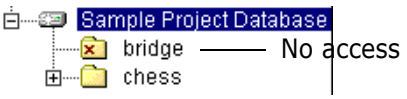


5 Verify that the location and name displayed in the **Access Control Database** field are correct. Modify it if necessary.

6 Select the **Enable access control database security** check box.

7 Click OK.

Projects to which users do not have access (because their user IDs are not defined in the access control database associated with the project) are displayed in the desktop client with a red X on the folder icon.



Using the Command-Line Interface

The following basic steps provide an overview for setting up security using the command-line interface. Following this list are sections that provide detailed procedures.

1 Create an access control text file, which includes:

- Defining any custom privilege sets you want to have.
- Defining users and assigning privileges to the users.
- Defining access list groups and assigning privileges and users to the groups.

2 Create the access control database.

3 Define any access lists you want to have.

4 Enable security.

Creating an Access Control Text File

To create an access control database, you must first create an access control text file. Then, you use this text file to make the access control database.

Use a text editor to create the access control text file. The following is an example of an access control text file that defines two custom privilege sets, five users, and two groups.

```
# Define two custom privilege sets
privilege Update: Unlimited, NoDeleteRev, \
    NoDeleteVersion
privilege View: Get, ViewArchive
# Define five users
user annab/ (superuser)
user jimp/jimp ()
user kayj/kayj ()
user samg/samg ()
user timf/timf ()
# Define two access list groups
    group PROGRAM (Update): jimp kayj
group DOCUMENT (View): samg timf
```

In this example, the three lines below the first comment line define custom privilege sets. The Administrator, annab, is given SuperUser privileges. This means she can perform any Version Manager action on any archive. The other four users have Unlimited privileges, which is indicated by the empty parentheses ( ). However, the group definitions limit their access to archives. Groups can only limit user access; they cannot increase user privileges.

The following rules apply to creating an access control text file:

- You must define each custom privilege set, access list group, and user on a separate line.
- User, access list group, and privilege set names cannot exceed 30 characters in length.
- Comment lines must be preceded with either a pound sign (#) or an exclamation point (!). For example,
  `# Define two custom privilege sets.`
- Use a backslash (\) to continue a line.
- User IDs, groups, and privileges are case sensitive unless you use the NoCase directive to make them case insensitive.
- You can use any name for the access control text file.

Now let's look at the syntax for the custom privilege set, user, and access list group definitions.

Custom privilege set definition

Syntax

```
privilege [=] custom_name: \
    component[,component...] \
    [:promo_group[,promo_group...]]
```

where:

custom_name is the name of the custom privilege set you are defining. This name cannot exceed 30 characters in length.

component specifies the base, composite, or privilege sets that make up the custom privilege set. Separate multiple values with commas.

promo_group specifies the promotion group to which this privilege is assigned. Separate them by commas. Refer to "Restricting a User's Ability to Promote" on page 258.

Example The following example defines a custom privilege set named Update, which consists of the Get, Lock, Unlock, Put, and StartBranch base and composite privileges.

```
privilege Update: Get, Lock, Unlock, Put, \ StartBranch
```

User definition

To identify valid Version Manager users and assign them privileges, you specify their user IDs in the access control database.

Syntax

```
user [=] user_id[/password] \
    [(privilege, privilege...)] \
    [-d date_range]
```

where:

user_id is how Version Manager identifies a user. To provide user access, this value must match the value that Version Manager obtains using a login source. See "Planning Security" on page 209.

password specifies the user's password if the VLOGIN login source is specified in the configuration file being used (desktop client only). If you do not specify VLOGIN as a source for Version Manager user identification, then do not specify a password.

privilege specifies the base, composite, or custom privilege sets assigned to this user. If you specify a custom privilege set, you must already have defined it earlier in the access control database. If you do not specify privileges, the user has the Unlimited privilege. Separate multiple privileges with commas.

date_range is the time period during which this user ID is valid. The syntax for *date_range* is `ddmonyyyy*ddmonyyyy`. For example, `-d 01jan2003*31dec2003`.

Example In the following example, the user ID is annab, there is no password, and the user ID is valid from January 1 through December 31, 2003. Also, annab has been assigned the SuperUser privilege.

```
user annab/ (superuser)\
    -d 01jan2003*31dec2003
```

Access List Group definition

An access list group is a set of users. Members of a group should have the same type of project responsibilities and need the same privileges.

Syntax

```
group [=] group_name [(privilege, privilege...)] \ [:]
    member,[member...]
```

where:

group_name is the name of the access list group you are defining. Group names cannot exceed 30 characters in length.

privilege specifies the base, composite, or custom privilege sets assigned to this group. If you specify a custom privilege set, you must already have defined it earlier in the

access control database. If you do not specify privileges, the group has the Unlimited privilege. Separate multiple privileges with commas.

member is either a user ID or previously defined group name. Separate multiple members with commas.

Example In the following example, user marym belongs to three groups. She is a member of group eng1, and her membership in this group makes her a member of the groups support and engineering. The eng1 group has the Unlimited privilege, indicated by the empty parentheses ( ). The support group has the custom privilege set BasicUser assigned to it. And, the group engineering has the custom privilege set Dev assigned to it. Marym has the union of all of the privileges of the groups she belongs to.

```
group eng1 (): marym, steveb, marvinp
group support (BasicUser): eng1, marthac, adamj
group engineering (Dev): eng1, toms
```

Creating the Access Control Database

To create the access control database from the access control text file, use the makedb command as follows:

makedb -a*database_name text_file*

where:

database_name is the name and location of the resulting access control database.

text_file is the name and location of your access control text file.

NOTE If you are using the makedb command on the output of the readdb command, you must edit the text file before using makedb. readdb places the users at the beginning of the file, and the file must be edited to move the users to the bottom of the file. makedb is located in the admin subdirectory of the bin directory.

The following example creates the access control database access.db in the location c:\pvcs (for UNIX, /usr/pvcs) from the text file access.txt in the location c:\pvcs (for UNIX, /usr/pvcs).

Windows
```
makedb -ac:\pvcs\access.db c:\pvcs\access.txt
```

UNIX
```
makedb -a/usr/pvcs/access.db /usr/pvcs/access.txt
```

NOTE If readdb is used on the Access Control Database that has an expiration date, makedb cannot read the output of the readdb command. The expiration date in the access.txt file cannot be read by makedb.

Defining Access Lists

To use access lists, you must already have defined an access control database.

Use the AccessList directive to specify the access lists for new archives. When the AccessList directive is in your configuration file, each archive you create will have the access list that is specified by this directive.

Use the VCONFIG command with the -A parameter to specify the access lists for existing archives. For information about this command, refer to the *Serena PVCS Version Manager Command-Line Reference Guide*.

Syntax `AccessList = user_id[,user_id...]`

where *user_id* is a user or a group of users who are allowed access to the archive. Separate multiple values with commas. The case of the user IDs or group names in the access list must match the case in the access control database, unless you use the `NoCase` directive to make user IDs case-insensitive.

Example This example places two access list groups in the access list for all newly created archives.

```
AccessList = engineering, qa
```

Enabling Security

Now that you have defined an access control database, you must tell Version Manager to use it to control security. You can do this by either embedding the access control database into Version Manager (see "Embedding an Access Control Database into Version Manager" on page 228) or specifying the access control database in a configuration file (as described here). An access control database specified in a configuration file will be used if the configuration file is:

- Embedded into Version Manager.
- Specified on the command line when you issue a command. Most commands provide the `-C` command-line option that lets you specify a configuration file.
- Defined in the VCSCFG environment variable.

You should define access control in the master configuration file.

To specify an access control database in a master configuration file:

1. Open the configuration file in a text editor.
2. Add the following lines to the master configuration file:

   ```
   AccessDB = database_name
   AccessControl
   ```

   where *database_name* is the name and location of the access control database you created.
3. Optionally, you can disallow the `AccessDB` and `AccessControl` directives so that other users cannot override these settings in another configuration file. To do this, add the following lines to the master configuration file:

   ```
   Disallow AccessDB
   Disallow AccessControl NoAccessControl
   ```
4. Save the file and exit the text editor.

Embedding an Access Control Database into Version Manager

Embedding an access control database into Version Manager ensures that all users will be using the same security definition and that users cannot use a different access control database. An access control database that is embedded into Version Manager affects all

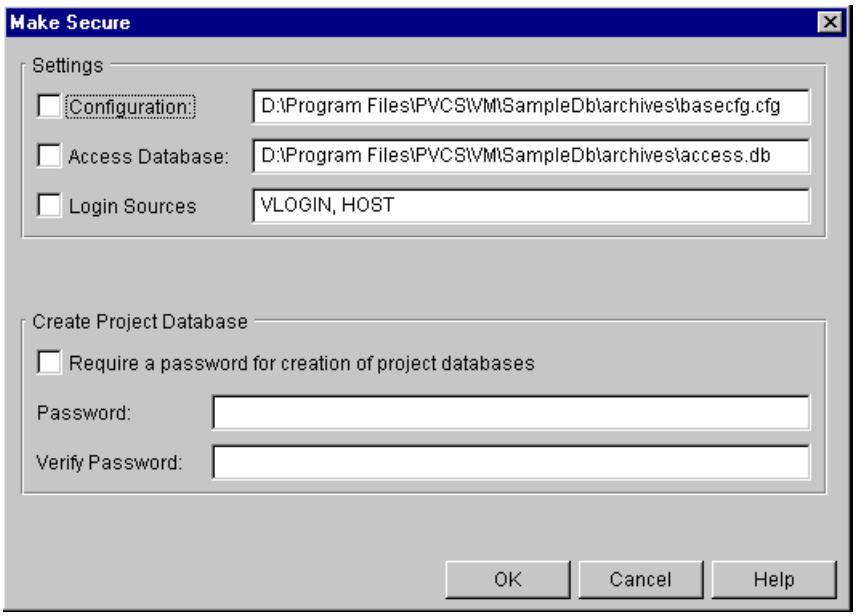
desktop client and command-line users who are using the copy of Version Manager that has the file embedded.

IMPORTANT! An access control database that is embedded into Version Manager does not work unless it is enabled. By default, the access control database is enabled.

Using the Desktop Client

To embed an access control database:

1. Select the project database associated with the access control database you want to embed into Version Manager.
2. Select Admin | Make Secure. The Make Settings Secure dialog box appears.



3. Select the **Access Database** check box. The field next to this check box displays the location and name of the access control database you are embedding. You cannot edit this field.
4. Click OK.

Using the Command-Line Interface

To embed the access control database, use the VCONFIG command:

Syntax `vconfig -a`*`database_name vm_filename`*

where:

`database_name` is the location and name of the access control database.

`vm_filename` is either:

- vmwfvc.dll for Windows

- vmufvc.a for UNIX

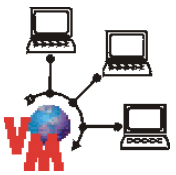
NOTE vconfig is located in the admin subdirectory of the bin directory.

After you embed the name of the access control database, you should move the file (VMWFVC.DLL or vmufvc.a) to the same location as the Version Manager executable files, if the file is not there already. Also, make sure users do not have write permission to this location. You do not want users to be able to embed a different access control database into Version Manager.

For more information about the VCONFIG command, see the *Serena PVCS Version Manager Command-Line Reference Guide*.

Restricting the Creation of Project Databases

To restrict users from creating project databases, you must define a password that users must enter before the action can take place. Then, you withhold the password from users who are not allowed to create project databases.



NOTE The following section applies only to non-file-server project databases. For project databases located on a file server, the password is configured in the Version Manager File Server Administration utility. See "Configuring Path Map Security Options" on page 140.

The password that you define is embedded into a security settings file named vmwfvcj.dll in Windows and vmufvcj.a on UNIX. This file is installed in *install_dir*\vm\common\bin\win32 in Windows and in *install_dir*/vm/common/bin/*OS_Name* on UNIX. By default, this file does not have a password embedded in it, and therefore, any user can create a project database.

When you define a password in this file, all desktop client and project command-line users who are using the copy of Version Manager that contains this file are affected. To define a password in this file, you, as the Administrator, must have write permissions to the bin directory. All other users should **not** have write permissions to the bin directory, or at least to the security settings file, so that they cannot initially define the password or change the password.

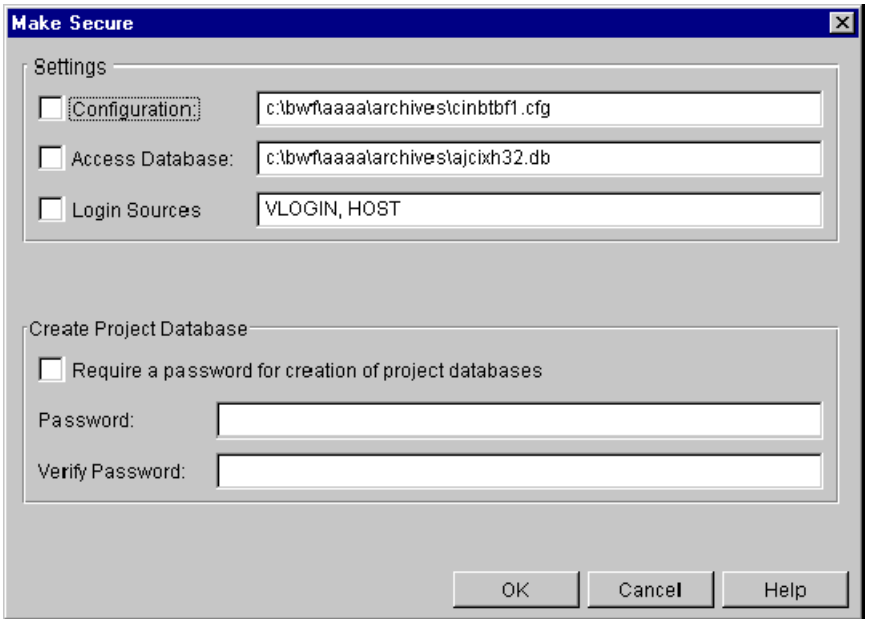
Your users must use the copy of Version Manager that you configure to restrict project database creation. For this to happen, the users must perform a workstation install of Version Manager. Refer to the *Serena PVCS Version Manager Installation Guide* for detailed information about workstation installations.

When the creation of project databases is password restricted, the desktop client prompts users for a password after they complete the Create Project Database dialog box (Admin | Create Project Database) or if they copy a project database (Edit | Copy). The password must be entered exactly as it was defined; the password is case-sensitive.

The **Create a new project database** check box is not visible to users in the Welcome to Serena Version Manager dialog box the first time they start Version Manager when the creation of project databases is password restricted.

To restrict the creation of project databases:

1 Select Admin | Make Secure. The Make Secure dialog box appears.



2 Select the **Require a password for creation of project databases** check box.

3 Specify a password in the **Password** field. The password is case-sensitive. The characters you enter appear as asterisks (*) in the field.

4 Reenter the password in the **Verify Password** field.

Maintaining Security

Once you have set up security, you will need to make changes occasionally to support resignations, additional staffing, changes in projects, etc. This section provides procedures for maintaining security.

You can view the settings of the access control database by using Admin | Security | Show Report. See "Generating Security Reports" on page 307.

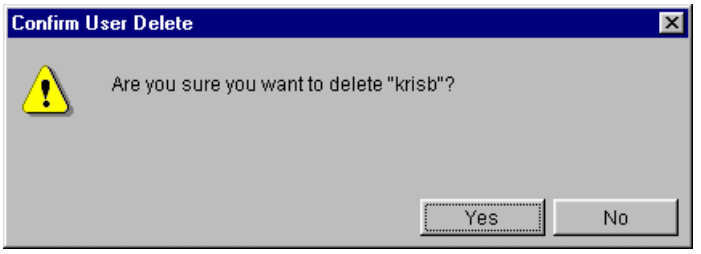
Removing Users from an Access Control Database

To remove users from an access control database:

1 Select the project database or project associated with the access control database you want to modify.

2 Select Admin | Security | Users. The Project Security dialog box appears.

3 In the **All Users** list, select the user you want to remove and click the Delete button. A confirmation dialog box appears.



4 Confirm that you want to remove the selected user by clicking Yes.

Changing a User's ID

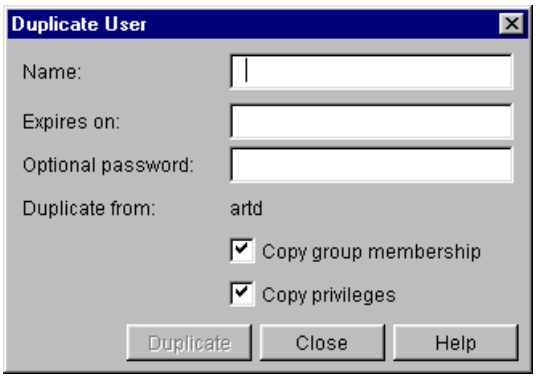
To change a user's ID, you must duplicate the user, change the ID in the duplicated (new) user definition, and then remove the old user definition.

To change a user's ID:

1 Select the project database or project associated with the access control database you want to modify.

2 Select Admin | Security | Users. The Project Security dialog box appears.

3 In the **All Users** list, select the user you want to change and click the Duplicate button. The Duplicate User dialog box appears.



In the Duplicate User dialog box, do the following:

a Change the user's ID in the **Name** field.

b Click Close. The Duplicate User dialog box closes and the Project Security dialog box becomes active.

4 In the Project Security dialog box, select the old user definition from the **All Users** list.

5 Click the Delete button. A confirmation dialog box appears.

6 Confirm that you want to remove the selected user by clicking Yes.

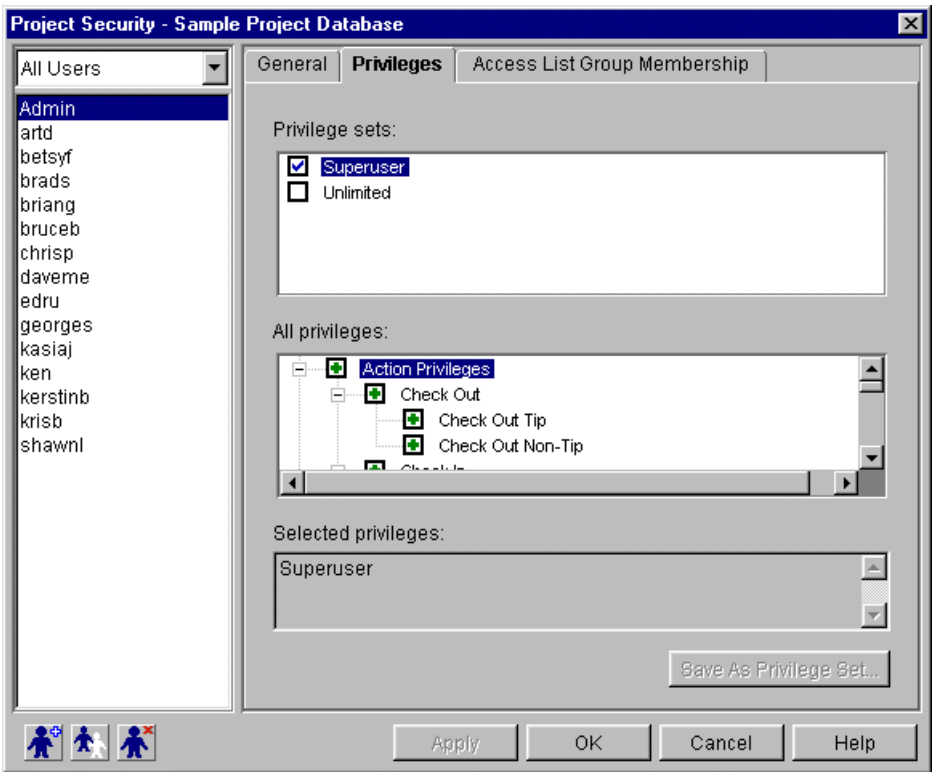
Modifying User Privileges

1 Select the project database or project associated with the access control database you want to modify.

2 Select Admin | Security | Users. The Project Security dialog box appears with the General tab active.

3 In the **All Users** list, select the user for whom you want to modify privileges.

4 Click the Privileges tab. On this tab, select the privileges and privilege sets to assign to the user.



5 Click **Apply**.

6 To change privileges for another user, repeat Steps 3–5. Otherwise, click OK to end the procedure.

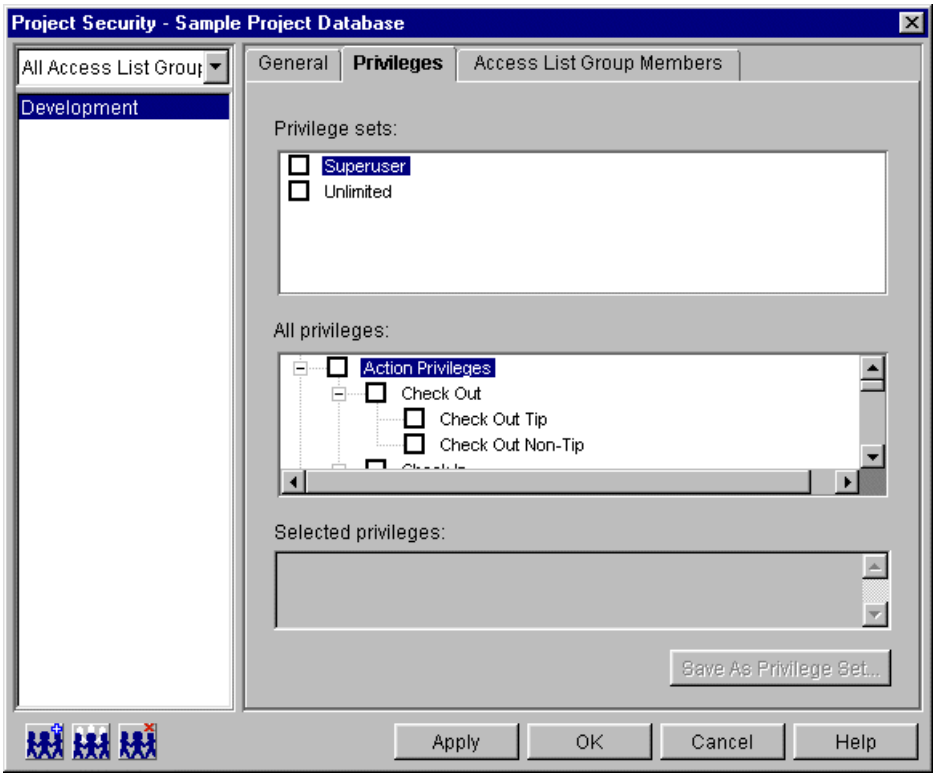
Modifying Access List Group Privileges

1 Select the project database or project associated with the access control database you want to modify.

2 Select Admin | Security | Access List Groups. The Project Security dialog box appears.

3 In the **All Access List Groups** list, select the group for which you want to modify privileges.

4 Click the Privileges tab. On this tab, select the privileges and privilege sets to assign to the group.



5 Click Apply.

6 To change privileges for another group, repeat Steps 3–5. Otherwise, click OK to end the procedure.

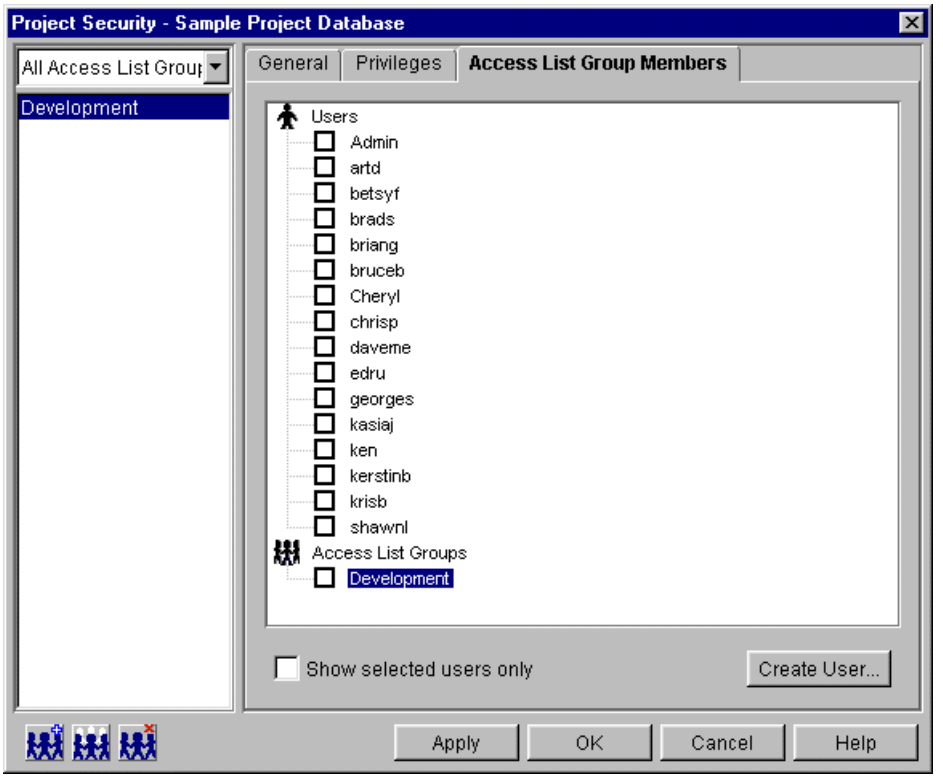
Modifying Access List Group Members

1 Select the project database or project associated with the access control database you want to modify.

2 Select Admin | Security | Access List Groups. The Project Security dialog box appears.

3 In the **All Access List Groups** list, select the group for which you want to modify members.

4 Click the Access List Group Members tab. On this tab, select the users to assign to the group.



5 Click Apply.

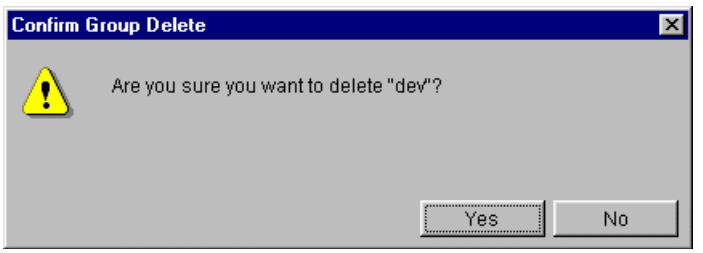
6 To change the members of another group, repeat Steps 3–5. Otherwise, click OK to end the procedure.

Removing Access List Groups from an Access Control Database

1 Select the project database or project associated with the access control database you want to modify.

2 Select Admin | Security | Access List Groups. The Project Security dialog box appears.

3 In the **All Access List Groups** list, select the group you want to remove and click Delete. A confirmation dialog box appears.



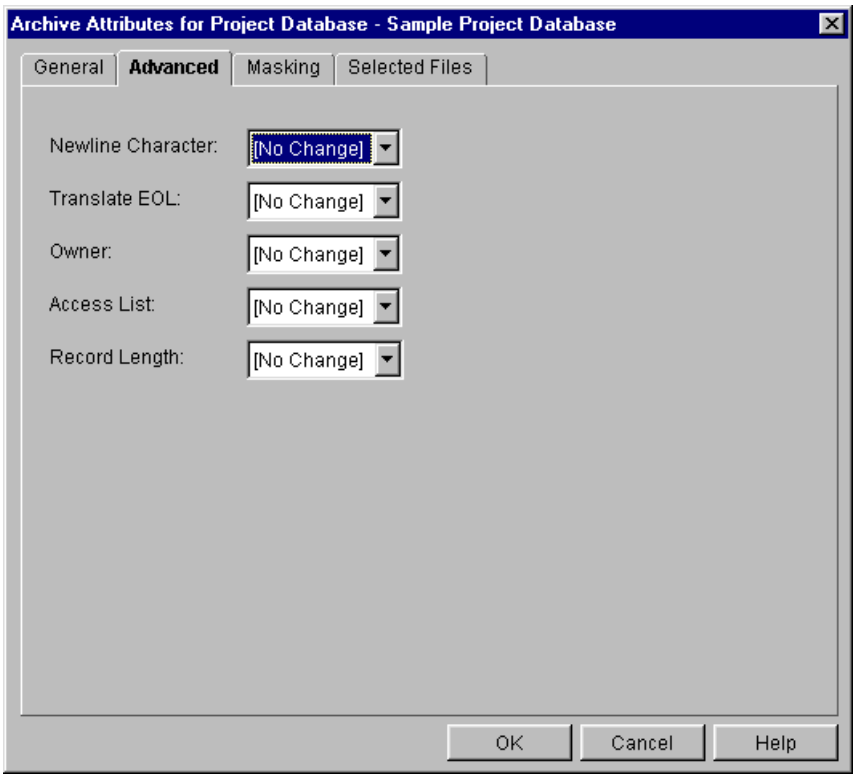
4 Confirm that you want to remove the selected group by clicking Yes.

Modifying Access Lists

1 Select the project or individual versioned file associated with the access list you want to modify. When you select a project, you modify the access list for all of the archives of the project.

2 Select Admin | Archive Attributes. The Archive Attributes dialog box appears with the General tab active.

3 Select the Advanced tab.



4 In the **Access List** field, select Delete, Modify, or Append. If you select Modify, enter the access list groups and individual users who can access the archive in the text field that appears to the right of the field. Each entry in the list must be separated by a comma (,). Or, you can click the Browse button and select the groups or individual users. The combined value of this field cannot exceed 254 characters in length.

If you select Append, enter the additional access list groups and users who can access the archive in the text field that appears to the right of the Access List field.

Remember that the access list groups and users who you specify must be defined in the access control database.

5 Click OK.

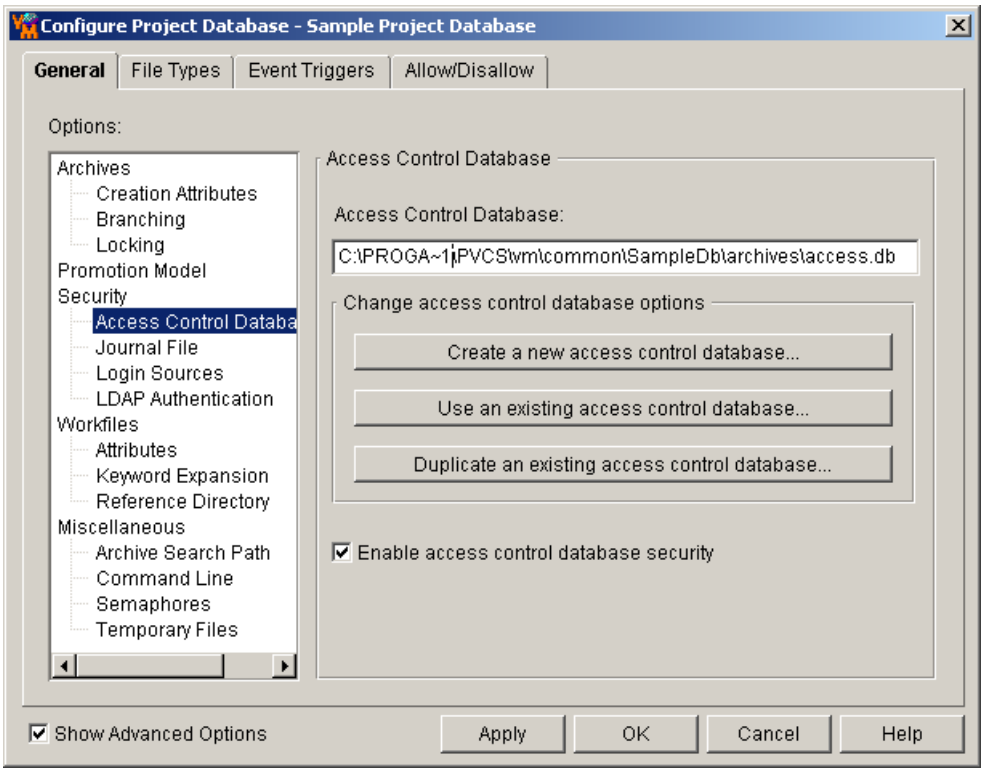
Disabling Security

1 Select the project database or project associated with the access control database that you want to disable.

2 Select Admin | Configure Project. The Configure Project dialog box appears with the General tab active.

3 If not already selected, select the **Show Advanced Options** check box.

4 In the Options list, select **Access Control Database** beneath security. The Access Control Database pane appears on the right.



5 Clear the **Enable access control database security** check box.

6 Click OK. The access control database no longer controls security for the project database or project.

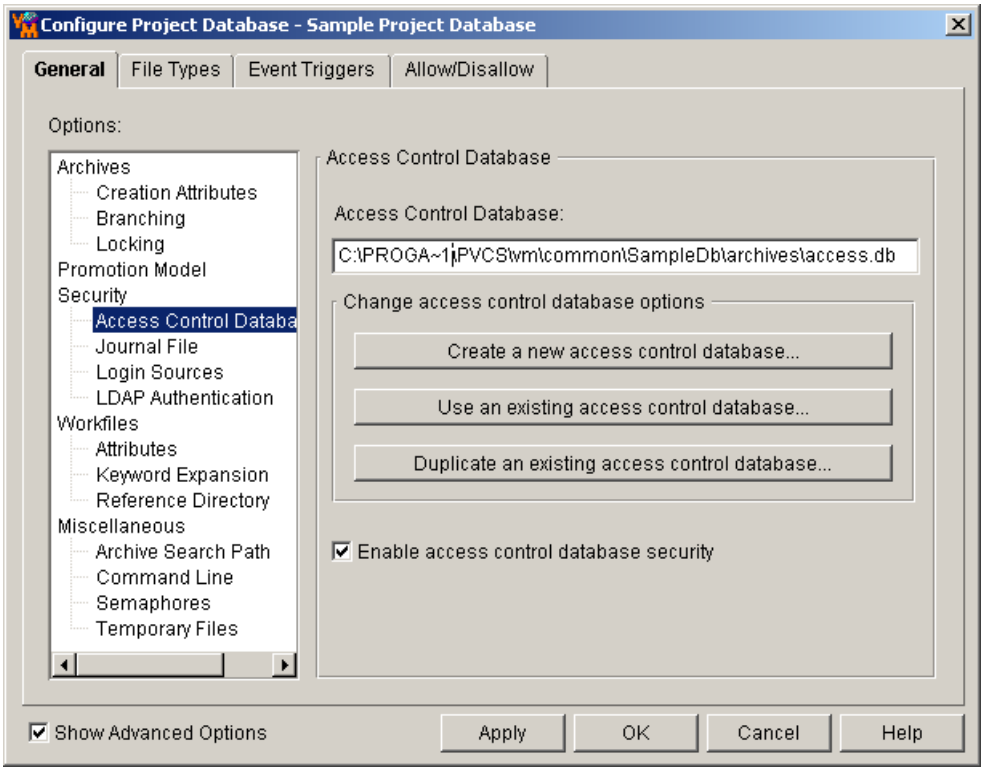
Changing the Access Control Database Associated with a Project Database

1 Select the project database or project for which you want to change the access control database.

2 Select Admin | Configure Project. The Configure Project dialog box appears with the General tab active.

3 If not already selected, select the **Show Advanced Options** check box.

4 In the Options list, select **Access Control Database** beneath Security. The Access Control Database pane appears on the right.



5 The Access Control Database field displays the access control database currently associated with the project database or project. To use a different access control database, do any of the following:

- Click the **Create a new access control database** button to create a new access control database in the archive directory of the project database or project. Version Manager copies the default access control database (default.db) to this location.
- Click the **Use an existing access control database** button to use an existing access control database. The Select Access Control Database dialog box appears so that you can choose an access control database. This option works well when you have multiple project databases that need to use the same security information. If you use the same access control database for each of the project databases, you only have to update one access control database when your security situation changes, for example, when you need to add a new user to the access control database.
- Click the **Duplicate an existing access control database** button to duplicate an existing access control database. The Select Access Control Database dialog box appears so that you can choose an access control database. Version Manager makes a copy of the access control database that you choose and places it in the archives directory of the project database.

6 Click OK.

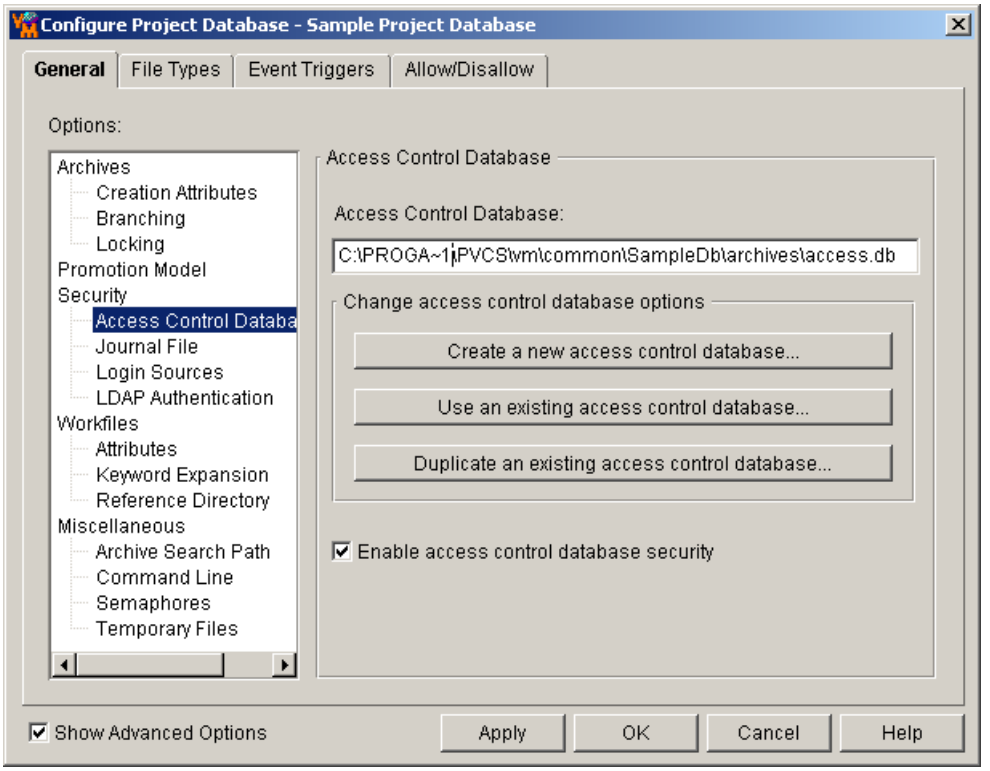
Removing the Association of an Access Control Database with a Project Database

1 Select the project database or project for which you want to remove the access control database.

2 Select Admin | Configure Project. The Configure Project dialog box appears with the General tab active.

3 If not already selected, select the **Show Advanced Options** check box.

4 In the Options list, select **Access Control Database** beneath Security. The Access Control Database pane appears on the right.



5 The Access Control Database field displays the access control database currently associated with the project database or project. To remove the access control database association, remove the value from the **Access Control Database** field; this field must contain no value.

6 Click OK.

Privilege Definitions

This section provides three tables: Base Privileges, Composite Privileges, and Default Privilege Sets.

Table 6-1 lists the base privileges, the actions they enable, and the desktop client command or dialog box fields they enable. Refer to the *Serena PVCS Version Manager*

Command-Line Reference Guide for information about which commands enable privileges in the command-line interface.

Table 6-1. Base Privileges

Command-Line Base Privilege Name Desktop Client Base Privilege Name	Action Desktop Client Command or Dialog Box Fields Enabled
Action Privileges	
GetTip CheckOutTip	Check out tip revisions. Invoke Merge Tool. Generate Difference Reports. Actions | Check Out, Actions | Show Merge, and Actions | Show Differences when tip revision is selected.
GetNonTip Check Out Non-Tip	Check out revisions other than the tip. Invoke Merge Tool. Generate Difference Reports. Actions | Check Out, Actions | Show Merge, and Actions | Show Differences when nontip revision is selected.
PutTrunk Check In Tip	Store revisions. Actions | Check In when tip revision is selected. File | Add Workfiles.
PutBranch Check In Branch	Store branch revisions. Actions | Check In when nontip revision is selected.
StartBranch Create Branch	Create branches. Force Branch check box is enabled in the Check In dialog box.
LockTip LockTip	Lock tip revisions. Actions | Lock when tip revision is selected.
LockNonTip Lock Non-Tip	Lock nontip revisions. Actions | Lock when nontip revision is selected.
Unlock Unlock	Remove locks on revisions owned by user. If both this and the Break Lock privilege are granted, Actions | Unlock.
BreakLock Break Lock	Unlock revisions. Users field enabled in Unlock dialog box.
AddVersion Add Version Label	Assign version labels to revisions. Actions | Version Label | Assign (Combine with DeleteVersion to Rename version labels: Actions | Version Label | Rename.)
ModifyVersion Modify Version Label	Re-assign version labels to different revisions. Actions | Version Label | Assign
DeleteVersion Delete Version Label	Delete version labels. Actions | Version Label | Delete (Combine with AddVersion to Rename version labels: Actions | Version Label | Rename.)
AddGroup Add Promotion Group	Assign promotion groups to revisions. Actions | Promotion Group | Assign
ModifyGroup Modify Promotion Group	Rename/move promotion groups. Actions | Promotion Group | Change

Command-Line Base Privilege Name Desktop Client Base Privilege Name	Action Desktop Client Command or Dialog Box Fields Enabled
DeleteGroup Delete Promotion Group	Disassociate promotion groups from revisions. Actions | Promotion Group | Remove
Promote Promote to the Next Promotion Group	Promote revisions. Actions | Promotion Group | Promote
ViewAccessDB View Access Control Database	Generate a security report. Admin | Security | Show Report
Archive Privileges	
InitArchive Create Archive and Workfile	Create archives. File | Add Workfiles.
ChangeAccessList Modify Archive Access List	Modify access lists. Access List field is enabled in Archive Attributes dialog box.
ChangeCommentDelimiter Modify Comment Delimiter	Change comment prefixes. Comment Delimiter field is enabled in the Archive Attributes dialog box.
ChangeOwner Modify Archive Owner	Change archive owners. Owner field is enabled in the Archive Attributes dialog box.
ChangeProtection Modify Archive Attributes	Change the following archive attributes: whether revision locking is enabled, more than one revision can be locked at a time, keywords are expanded, translation is performed on UNIX, archives are write-protected, information is compressed, and delta records are generated. Protection fields are enabled in the Archive Attributes dialog box.
ChangeWorkfileName Modify Versioned File Name	Change the workfile name attribute for archives. Workfile Name fields is enabled in the Archives Attributes dialog box. The Modify Versioned File Name privilege is currently not implemented. You cannot rename a versioned file.
ModifyChangeDescription Modify Change Description	Edit change descriptions for revisions. Revision Description field is enabled in the Properties dialog box.
ModifyWorkfileDescription Modify Workfile Description	Change workfile descriptions. Workfile Description field is enabled in the Properties dialog box.
DeleteRevTip Delete Tip	Delete tip revisions. Actions | Delete Revision and File | Delete when tip revision is selected.
DeleteRevNonTip Delete Non-Tip	Delete nontip revisions. Actions | Delete Revision and File | Delete when nontip revision is selected.
ViewArchiveHeader View Archive Header	View archive header information. Versioned file information is displayed in the Versioned File Properties dialog box.
ViewArchiveRev View Archive Revisions	View revision histories. Archive header information is displayed in the Versioned File Properties dialog box.
Project Privileges	

Command-Line Base Privilege Name Desktop Client Base Privilege Name	Action Desktop Client Command or Dialog Box Fields Enabled
NoProjectNewProject Create Project	Create projects. File | Create Project
NoFolderChangeFolder Modify Project	Rename projects and change projects' attributes. File | Properties File | Rename Modify workfile locations of Root and public workspaces.
NoProjectDeleteProject Delete Project	Delete projects. File | Delete with project selected.
NoProjectCopyProject Copy Project	Copy projects. File | Copy with project selected.
NoFolderChangeFolderMembers Add or Remove Versioned Files	Add workfiles to a project database or project. File | Add Workfiles Import archives. Admin | Import Archives Delete versioned files from project databases or projects. File | Delete when versioned file is selected.
NoProjectConfigureProject Configure Project	Configure a project or project database. Admin | Configure Project Modify the Base, Branch, Default version, and Default Promotion Group settings for Root and public workspaces.
NoOptionsSecurity Configure Security	Define users, groups, privileges, and login sources for security. Admin | Security
NoActionsJournalReport Show Journal	Generate Journal Reports. Actions | Show Journal.
Folder Privileges (5.3/6.0 projects only, these privileges do not appear in the desktop client)	
NoFolderNewFolder	Create folders. File | Create Folder
NoFolderChangeFolder	Rename folders and change folders' attributes. File | Properties File | Rename
NoFolderDeleteFolder	Delete folders. File | Delete with folder selected.
NoFolderCopyFolderMemebers	Copy folders. File | Copy with folder selected.
NoFolderChangeFolderMemebers	Add workfiles to 5.3/6.0 project. File | Add Workfile Delete versioned files from 5.3/6.0 projects. File | Delete when versioned file is selected.
NoFolderUpdateProjectFolder	Update a 5.3/6.0 project. File | Update Project Folder

Composite Privileges

Table 6-2 lists the composite privileges and the base privileges that comprise them. This table first lists the command-line interface name and then the desktop client name for each privilege. Refer to Table 6-1, "Base Privileges," on page 240 for a definition of each base privilege.

Table 6-2. Composite Privileges

Command-Line Composite Privilege Name **Desktop Client Composite Privilege Name**	**Corresponding Base Privileges**
Action Composite Privileges	
Get Check Out	GetNonTip GetTip Check Out Tip Check Out Non Tip
Put Check In	PutBranch PutTrunk Check In Tip Check In Branch
Lock Lock	LockTip LockNonTip Lock Tip Revision Lock NonTip Revision
Version Label (desktop client only)	Add Version Label Modify Version Label Delete Version Label
Promotion Group (desktop client only)	Add Promotion Group Modify Promotion Group Delete Promotion Group Promote to The Next Promotion Group
Archive Composite Privileges	
Modify Archive Properties (desktop client only)	Modify Archive Access List Modify Comment Delimiter Modify Archive Owner Modify Archive Attributes Modify Versioned File Name
ModifyDescription Modify Description	ModifyChangeDescription ModifyWorkfileDescription Modify Change Description Modify Workfile Description
DeleteRev **Delete Revision**	DeleteRevTip DeleteRevNonTip Delete Tip Revision Delete Non Tip Revision
ViewArchive View Archive Details	ViewArchiveHeader ViewArchiveRev View Archive Header View Archive Revisions

Command-Line Composite Privilege Name Desktop Client Composite Privilege Name	Corresponding Base Privileges
Project Composite Privileges	
Project (desktop client only)	Create Project Modify Project Delete Project Copy Project Add/Remove Workfiles

Default Privilege Sets

Table 6-3 lists the default privilege sets that Version Manager provides and the base privileges, composite privileges, and other privilege sets that provide the definition of the default privilege sets.

NOTE The SuperUser and the Unlimited privilege sets cannot be modified. The remaining privilege sets are included as samples and may be modified.

Table 6-3. Default Privilege Sets

Privilege Set	Base Privileges
SuperUser	All privileges listed in Table 5.1 plus the actions: copy and rename project databases, and not check in workfiles when adding workfiles. This privilege set is not restricted by access lists.
Unlimited	All privileges listed in Table 5.1 plus the action: not check in workfiles when adding workfiles.
Developer	Project Lead privilege set minus Create Project, Configure Project, Copy Project, Delete Project, Add or Remove Versioned Files, Modify Project, and Configure Security.
Project Lead	Lock Tip, Lock Non-tip, Unlock, Check Out Tip, Check Out Non-tip, Check In Tip, Check In Branch, Create Branch, Modify Archive Access List, Modify Archive Owner, Modify Archive Attributes, Modify Comment Delimiter, Modify Versioned File Name, Modify Workfile Description, Modify Change Description, Add Version Label, Delete Version Label, Modify Version Label, Create Archive and Workfile, Delete Tip, Delete Non-tip, View Archive Header, View Archive Revisions, Promote to Next Promotion Group, Add Promotion Group, Modify Promotion Group, Delete Promotion Group, Create Project, Configure Project, Copy Project, Delete Project, Add or Remove Versioned Files, Modify Project, Configure Security, and Show Journal.
Support	View Archive Header, View Archive Revisions, and Show Journal.

Privilege Set	Base Privileges
Quality Assurance	Support privilege set plus Add Version Label, Delete Version Label, Modify Version Label, Promote to Next Promotion Group, Add Promotion Group, Modify Promotion Group, and Delete Promotion Group.
Documentation	Quality Assurance privilege set plus Lock Tip, Lock Non-tip, Unlock, Check Out Tip, Check Out Non-tip, Check in Tip, Check In Branch, and Create Archive and Workfile.

Chapter 7

Using Promotion Models

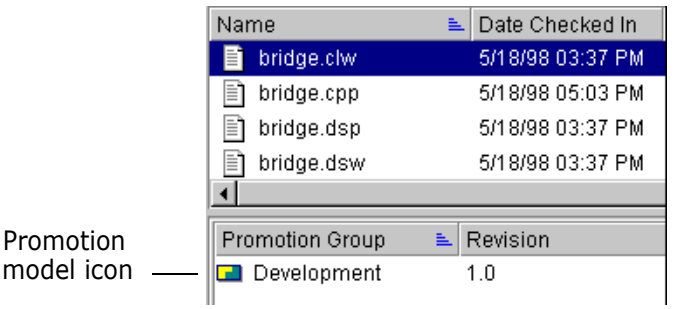
Introduction

A *promotion model* is a hierarchy of milestones in a development cycle. You use promotion to control the development of source code from the design phase to final release. Each milestone is represented by a *promotion group*. Examples of promotion groups include Development, QA, and Production.

The following figure shows an example of a promotion model.

Serena Version Manager's promotion model is based on a logical association between a revision of a file and a milestone in the development cycle. Unlike promotion in the mainframe environment, Version Manager does not require you to maintain files at different stages of development in different physical locations or to move files to new locations to promote them.

Version Manager archives stores information about the logical associations between revisions and promotion groups. The Version Manager desktop client displays these associations, as shown in the following figure. The promotion model icon indicates revisions that are associated with different promotion groups in the promotion model.



Promotion Model Rules

Rule 1: A promotion model can have one or more lowest-level groups.

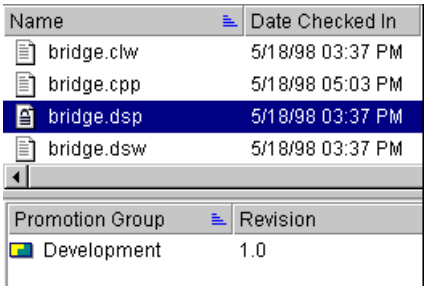
Rule 2: When a promotion model is in effect, every revision that you check out or lock must be associated with a lowest-level promotion group. If your model has only one lowest-level group, Version Manager automatically associates that promotion group with each revision that is checked out or locked.

If your model has more than one lowest-level group and a default lowest-level promotion group was set using File | Properties | Workspace Settings, Version Manager uses the default lowest-level promotion group, unless you explicitly select another one. If your model has more than one lowest-level promotion group and no default has been set, you will be prompted to select one.

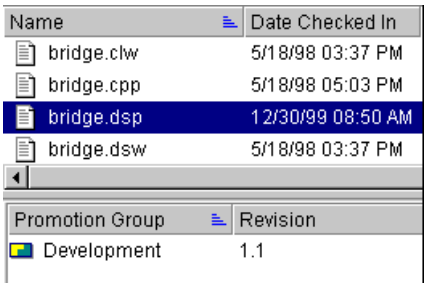
For example, assume that the promotion model in effect is the following:

Development ⇒ QA_Test1 ⇒ Release

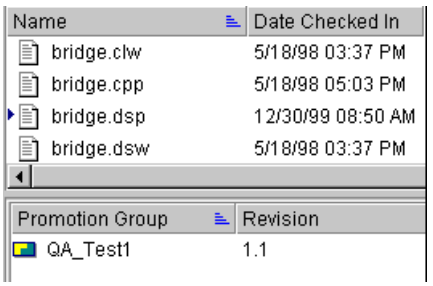
As shown in the following figure, when revision 1.0 of bridge.dsp is checked out, Version Manager automatically associates the revision with the Development group, the lowest-level group.



Rule 3: A lowest-level group "floats" with the most recent revision. As you check in subsequent revisions of a file, Version Manager reassigns the lowest-level promotion group to the newest revision.

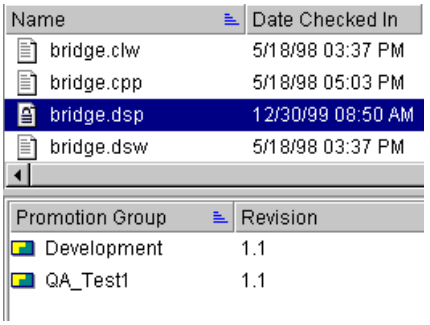


Rule 4: When a revision is promoted to the next higher promotion group, the revision is no longer associated with the lower-level promotion group. As shown in the following figure, when the revision associated with Development is promoted, Version Manager associates the revision with the QA_Test1 group and disassociates it from the Development group.



Rule 5: If a revision is associated with a higher promotion group, upon check out Version Manager also associates the revision with the lowest-level promotion group. As stated earlier, revisions that are checked out or locked must be associated with a lowest-level promotion group. This means that when you check out a revision that is

associated with a higher promotion group, such as QA_Test1, Version Manager creates an additional association with the lowest-level promotion group.



At this point, the revision is associated with both groups, QA_Test1 and Development. However, when you check in the new revision, it is associated with the Development group only and the previous revision remains associated with the QA_Test1 group.

Rule 6: A promotion group can be associated with only one revision in an archive at a time. A revision can be associated with more than one promotion group as we discussed in Rule 5. But, a promotion group can be associated with only one revision in an archive at a time. This means that if the promotion model in effect has only one lowest-level promotion group and you attempt to check out a revision when another revision is already associated with that lowest-level group, Version Manager won't allow you to check out the revision. For example, if there are revisions 1.0 through 1.5 in an archive and the lowest-level promotion group is associated with revision 1.5, Version Manager will not allow you to check out revision 1.4, or any of
the other revisions. If you encounter this situation, you must either:

- Promote the revision associated with the lowest-level group
- Define another lowest-level group

Defining a Promotion Model

A promotion model can be defined in both a master configuration file and project or local configuration files. In the desktop client, you can think of this as a promotion model defined for a project database and other promotion models defined for projects/ subprojects.

When a promotion model is defined in both types of configuration files, the models work together; the promotion model definition in the project or local configuration file does **not** override the promotion model definition in the master configuration file. For example, you could define a promotion model of

Dev ⇒ QA ⇒ Production

for the master configuration file and a promotion model of

Dev1⇒ QA

for a project or local configuration file.

In this case, there are two lowest-level promotion groups, Dev and Dev1, and each promote to QA. And, QA promotes to Production, which is defined in the master configuration file. You **must not** duplicate the QA to Production promotion path in the

project or local configuration file. If you do, the promotion model becomes invalid, and you must correct the configuration (by removing the duplication) before opening the project.

When you define a promotion model in a master configuration file and then define one in a project or local configuration file, you need to be careful not to introduce conflicts between the promotion model definition in the project or local configuration file and the definition in the master configuration file. For example, if the promotion model in the master configuration file is the one stated above, you could not define a promotion model in the project or local configuration file of

Dev ⇒ Production

This would cause a conflict because Version Manager is being told two different things: first to promote Dev to QA and second to promote Dev to Production.

Using the Desktop Client

To define a promotion model, you must have the Configure Project privilege assigned to you, your project database or project must have a configuration file associated with it, and if you are defining a promotion model in a project, the master configuration file must allow the Promote option (directive).

In the desktop client, if you have a hierarchy of projects with each containing a promotion model definition, the desktop client displays the promotion model that is in effect for the project you are working with. Using the previous example, the desktop client would display the following if you were working with the project database (master configuration file):

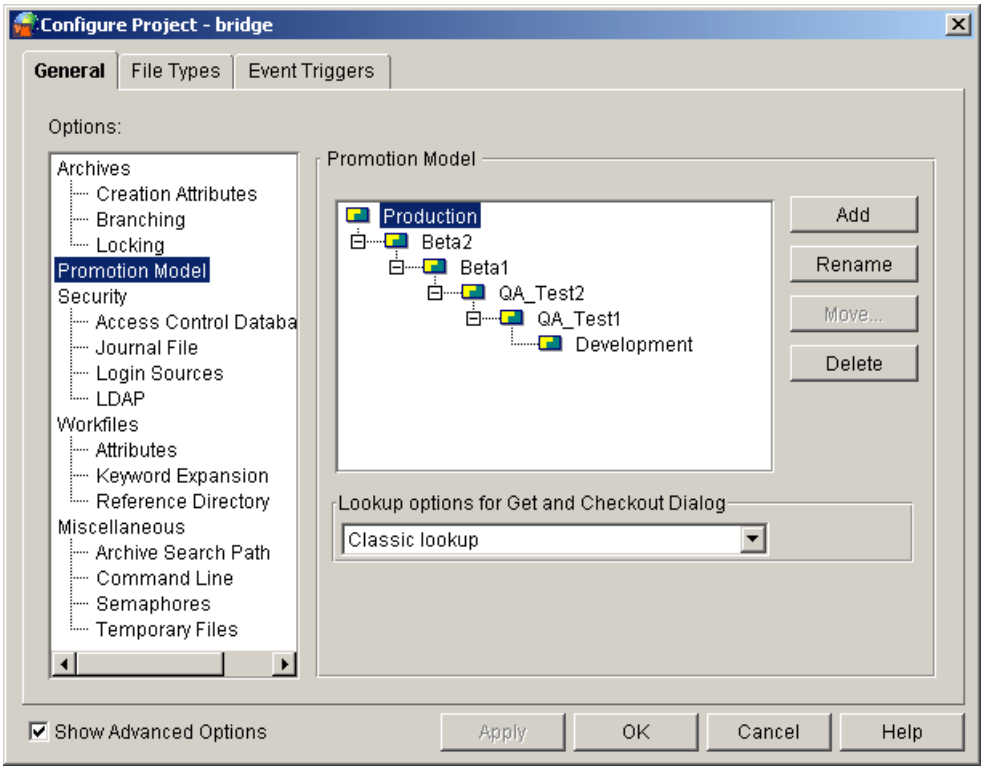
If you were working with the project, the desktop client would display the promotion model defined in the project database and the promotion model defined in the project:

To define a promotion model:

1. Select the project database or project for which you want to define a promotion model.
2. Select Admin | Configure Project. The Configure Project dialog box appears with the General tab active.
3. In the Options list, select **Promotion Model**. The Promotion Model pane on the right appears. If you are defining a promotion model for a project and a parent project or project database has a promotion model defined, the promotion model that is displayed includes the promotion model definition from the parent project. You should

not, however, rename or delete any part of the promotion model that is defined in the parent project. To do this, you must select the parent project in Step 1.

NOTE The desktop client allows you to select a promotion group of a parent project when working with a child project and use the Delete button, but the changes are not saved. Also, the desktop client allows you to use the Rename button on a promotion group defined in the parent project. In this case, the renamed promotion group is added to the promotion model of the current project and the promotion group in the parent project is not renamed. This could possibly cause conflicts. When working with a child project that has a promotion model defined in a parent project, you should only make changes to the promotion model defined in the child project, not in any of the promotion models above it in the hierarchy.



4 To add promotion groups, do any of the following:

- If no promotion groups exist, click **Add** to define the highest-level group.
- The new group is added below the selected group. If there are existing promotion groups, select a group and click **Add**.
- To add an additional lowest-level promotion group, select the existing lowest-level group and click **Add**. You must move the group to the same level as the existing lowest-level group. See Step 6 for how to move a promotion group.

5 To rename any of the promotion groups, select the promotion group you want to rename, click **Rename**, and enter a new name for the promotion group.

6 To move a promotion group up or down in the promotion model hierarchy, select the promotion group and click **Move**. A dialog box appears that lets you select the new parent promotion group of the group you want to move.

7 To delete any of the promotion groups, select the promotion group and click **Delete**. All promotion groups below the deleted group will also be deleted.

8 To configure the default revision lookup behavior for Get and Checkout operations, select one of the following from the **Lookup options for Get and Checkout Dialog** 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.
- **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.

9 Click **OK**.

Using the Command-Line Interface

If you typically use the Version Manager command-line interface, you may want to edit configuration files using a text editor. If you find it easier, you can use the desktop client to configure Version Manager. A configuration file that is maintained in the desktop client is compatible with the command-line interface.

To define a promotion model, your master configuration file must allow the Promote directive if you are configuring a local configuration file.

In the configuration file, add the Promote directive. For example:

```
Promote Development QA
Promote QA Release
```

The above example defines a promotion hierarchy that progresses in the following order: Development, QA, and Release.

For more information about the Promote directive, see the *Serena PVCS Version Manager Command-Line Interface.*

Promotion and Lifecycle Management

The primary purpose of promotion modeling is to formally manage the lifecycle of software development—from the design phase to final release. This section provides an example of using a promotion model for lifecycle management.

Scenario

Let's suppose a simple lifecycle:

Development ⇒ QA ⇒ Release

In this scenario, a promotion model is defined with promotion groups that parallel the lifecycle—Development, QA, and Release. With this promotion model in place, the developers are checking out files associated with the Development promotion group, which is, by default, the tip revision. When they check files back in, the new revision is associated with the Development promotion group because the lowest-level promotion group "floats" with the tip revision.

Next, the developers complete all of the changes needed for the product release and the revisions are promoted from Development to QA. Here is a before-and-after picture of a hypothetical archive:

Before	After
Rev 1.5 Development	Rev 1.5 QA
Rev 1.4	Rev 1.4
Rev 1.3	Rev 1.3
Rev 1.2 Release	Rev 1.2 Release
Rev 1.1	Rev 1.1
Rev 1.0	Rev 1.0

Now the quality assurance department starts testing the software and the developers go to work on the next release, checking out and in more changes. For instance, after a developer adds two revisions in the hypothetical archive, it looks like this:

Rev 1.7 Development
Rev 1.6
Rev 1.5 QA
Rev 1.4
Rev 1.3
Rev 1.2 Release
Rev 1.1
Rev 1.0

The quality assurance department does not have to concern themselves with any of the new development changes because they are referencing the QA promotion group when they check out revisions.

Later, when the quality assurance department is satisfied with the release, the revision associated with QA is promoted to Release. Now the hypothetical archive looks like this:

Rev 1.7 Development
Rev 1.6
Rev 1.5 Release
Rev 1.4
Rev 1.3
Rev 1.2
Rev 1.1
Rev 1.0

Now, the release manager produces a final software product. He references the Release promotion group when checking out revisions; he is not affected by the changes being made by development.

Promotion and Parallel Development

This section discusses how you can use a promotion model to control parallel development. At the end of this section, there are two scenarios in which a promotion model has been defined to control parallel development.

Using a Promotion Model to Control Parallel Development

Parallel development is when a separate line of development branches off of the main line of development. Typically, this occurs when development:

- Makes changes, such as bug fixes, that they do not want to immediately affect the main line of development. For example, to make bug fixes to a release while work continues in development.
- Works on alternate versions of a product simultaneously (for example, multi-platform development).

To develop two versions of the same file in parallel when a promotion model exists, you must create a lowest-level group for each branch.

The benefit of using a promotion model with parallel development is that you can control the number of branches that can be created from any one revision. Because an archive can have only one revision associated with a promotion group and you must always check out a revision at a lowest-level group, the number of lowest-level groups you define in your promotion model determines the number of locks that Version Manager will allow on a revision. Therefore, the number of lowest-level groups you define determines the number of branches that you can create from a revision. For example, if you define two lowest-level groups, you can have two branches—the main branch (the trunk) and one parallel branch.

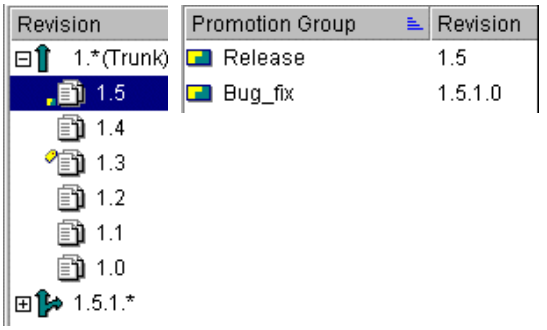
Scenario 1: Defining Promotion Groups for Emergency Bug Fixes

This scenario builds on the scenario that was presented in the section "Promotion and Lifecycle Management" on page 254. It explains one way of setting up a promotion model to allow you to fix bugs on an emergency basis.

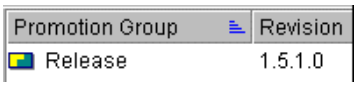
Assume a customer finds a bug in the new release (the one that included revision 1.5 of our hypothetical archive). The release manager wants to get the bug fixed and send a patch as soon as possible.

To handle the situation, the release manager creates another promotion group, Bug_fix. It promotes directly to the highest-level group, Release, as shown next.

To fix the defect, a developer checks out the revision associated with Release. When the developer checks out the revision, he must select the Bug_fix promotion group, which is one of the two lowest-level groups. The developer fixes the defect and checks the revision in as branch revision 1.5.1.0.



The developer immediately promotes the revision to Release, and the release manager rebuilds the application based on the Release promotion group.



At a later time, the developer will merge the branch back to the trunk at the Development group. This will ensure that the bug fix is part of subsequent releases.

Scenario 2: Defining Promotion Groups for Multi-Platform Development

This section explains the implications of using a promotion model to accommodate branches for alternate versions of a product.

NOTE If you never intend to merge a branch back to the trunk, or you plan to make major changes to either the branch or the trunk, you should create separate archives for each line of development.

The promotion model below accommodates multi-platform development. In each archive, the trunk is used to develop the product for Windows, and a branch is used to develop for UNIX.

WinDev ⇒ WinTest ⇒ Production

UnixDev ⇒ UnixTest

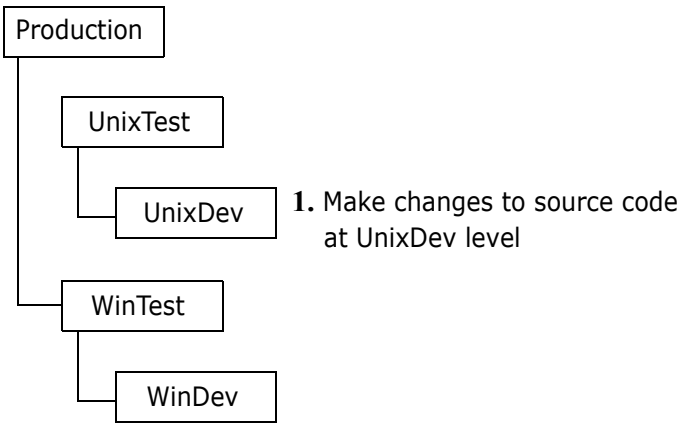
Note that UnixTest never promotes to Production.

Designing a promotion model for this purpose is useful only if the changes made for the UNIX version are fairly minor and if merges are performed on a regular basis.

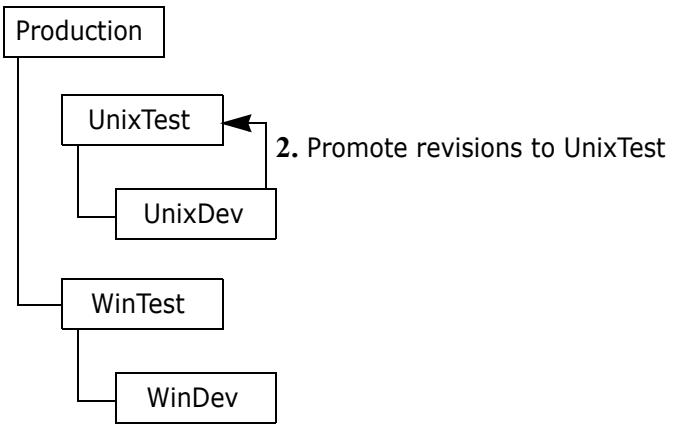
The following steps explain how you would:

- Make changes on the branch
- Promote to testing
- Merge the changes to the main line of development
- Promote to production

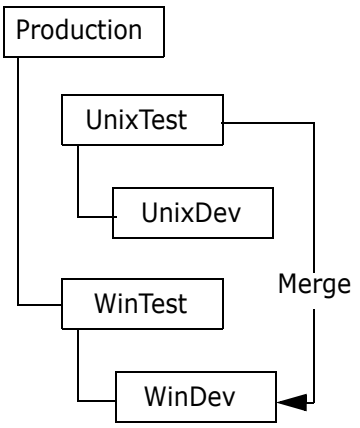
Step 1: Developers make changes to source code at UnixDev level, creating branches for UNIX development. For information about branching, see Chapter 8, "Branching and Merging Files" on page 263.



Step 2: Developers promote revisions to the UnixTest group for unit testing. Revisions associated with UnixTest are never promoted to the top-level group. This ensures that all code is merged and then promoted through the proper testing phases before it reaches Production.

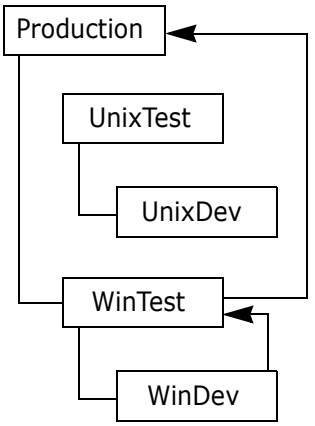


Step 3: Developers merge revisions associated with UnixTest with revisions at the WinDev promotion level. For information about merging, see Chapter 8, "Branching and Merging Files" on page 263.



3. Merge revisions at UnixTest with revisions at WinDev

Step 4: Promote revisions from WinDev to WinTest and finally to Production.



4. Promote through promotion model to Production

Restricting a User's Ability to Promote

NOTE If you are not familiar with how Version Manager security works, read "Using Security" on page 203 before continuing with this section. A promotion model must be in effect for the project database.

Restrict By Promotion Group

This section discusses how to integrate promotion models with security to control how revisions are promoted from one promotion level to the next. To use the Restrict By Promotion Group feature, you must do the following:

1. Define a custom privilege set.
2. Restrict the custom privilege set to a promotion level.
3. Assign this privilege set to a user or access group.

The user can then promote only from the assigned promotion level to the next promotion level. Restricting promotion from one level to the next by user or group ensures that the revision goes through the promotion hierarchy without skipping any levels.

NOTE An Administrator with Create Promotion Group, Modify Promotion Group, and Delete Promotion Group privileges can override the promotion model and process.

Example For example, suppose you have created a promotion model that progresses from Dev to QA to Prod. If you create a custom privilege set and enable the privilege Promote to Next Promotion Group, and then restrict this custom privilege set to the promotion level Dev, then any user assigned this custom privilege set can only promote from Dev to QA. This prevents any user in Dev from promoting directly to Prod.

The custom privilege set used to restrict a user's ability to promote should only have this one restriction defined. Other privileges should be assigned using one of the default custom privilege sets or by creating a new custom privilege set.

NOTE All other custom privilege sets for the user must not have Promote to Next enabled or any of the Administrative functions, such as Create Promotion Group, Modify Promotion Group, and Delete Promotion Group. This would invalidate the custom privilege set defined to restrict promotion by promotion group.

Using Multiple Lowest-Level Promotion Groups

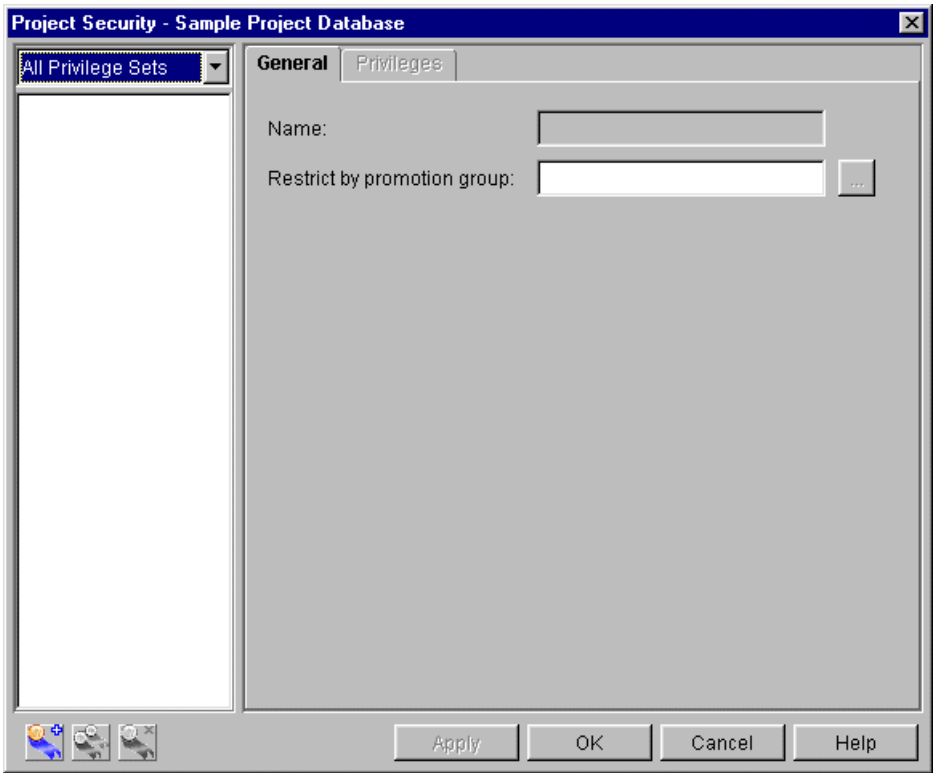
Once a promotion model is applied to a versioned file, then each revision checked out will default to the lowest-level promotion group within the promotion model. If you are using multiple lowest-level promotion groups, then you must restrict Lock to one of the low level promotion groups.

Using the Desktop Client

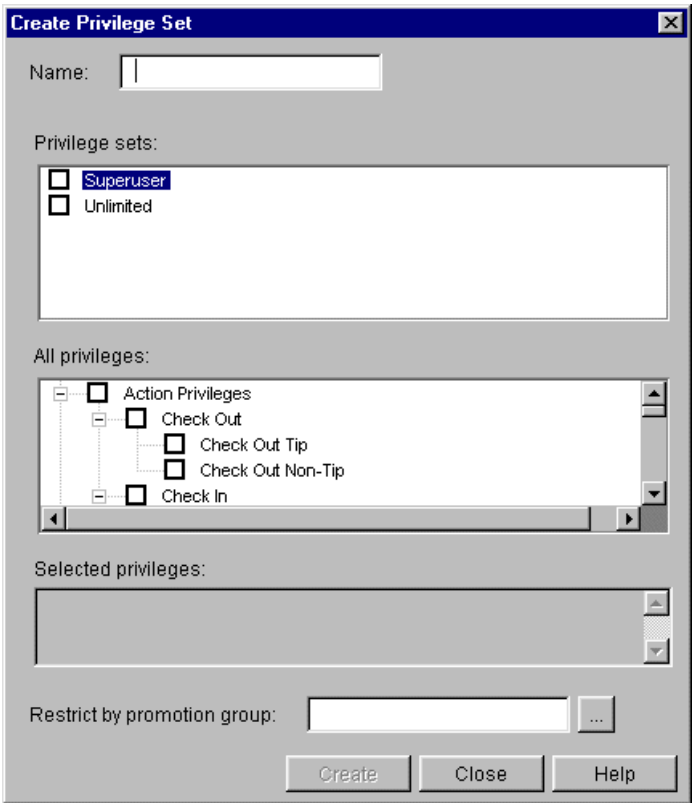
To restrict a user's ability by promotion group:

1 Select the project database or project associated with the access control database in which you are defining privilege sets.

2 Select Admin | Security | Privilege Sets. The Project Security dialog box appears with the General tab active.



3 Click the **New** button. The Create Privilege Set dialog box appears.



4 Enter a name for the custom privilege set.

The name cannot begin or end with a tab or blank space. Any other character can be used in the name except * : \ ' " ( ).

5 Select the Promote to the Next Promotion Group privilege. All other privilege selections should be canceled.

The options with a + in the box beside them are allowed. The Project privileges are the only privileges that can have a – in the box beside them because they are negative privileges. The – means that the privilege is denied.

To change the status of an option, click the check box beside the option. You can allow or disallow an entire set of related privileges by clicking the parent option. For example, you could assign all of the Action privileges by clicking the check box beside Action Privileges to place a + in the check box.

6 In the **Restrict by promotion group** field, enter the promotion group by which this privilege set will be restricted.

7 Click Create and then Close to create this privilege set and return to the Project Security dialog box.

8 Click OK.

9 Assign the user or group of users to this custom privilege set.

For information about how to:

- Define a promotion model, see "Defining a Promotion Model" on page 250.
- Define access list groups, see "Defining Access List Groups" on page 220.
- Define Access Lists, see "Defining Access Lists" on page 227.

Using the Command-Line Interface

To restrict access by promotion group, you must define the custom privilege set in the access control text file to be restricted to one or more promotion groups. See "Custom privilege set definition" on page 225.

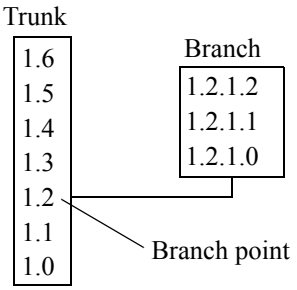
Chapter 8

Branching and Merging Files

Branching

A *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. Figure 8-1 shows that branching lets you develop alternate versions of a file in parallel with other developers who are working on the trunk or on another branch.

Figure 8-1. Branching



Some common reasons for creating branches are:

- To develop 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. See "Scenario 2: Defining Promotion Groups for Multi-Platform Development" on page 256 for an example of branching and promotion groups.
- To fix a bug without interrupting development on the trunk. If you discover a bug, but want development to continue in other areas of the source code file, you can create a branch from the revision containing the bug, fix it, and test it without impeding progress on trunk development. You can later *merge* the fix with the newest revision on the trunk.
- To create a baseline product and then customize 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.

Branches can diverge from the trunk or from other branches. The first branch revision from the trunk carries the two-digit revision number of the trunk revision, followed by its own two-digit branch revision number. The revision from which a branch begins is called the *branch point*.

For example, in Figure 8-1, "Branching," on page 264, 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 .1, so that subsequent revisions on this branch would be 1.2.1.1, 1.2.1.2, etc.

Version Manager identifies an entire branch by its first three successive revision numbers. For example, branch 1.2.1 contains revisions numbered 1.2.1.0, 1.2.1.1, 1.2.1.2 etc. Branch 1.3.1 would contain 1.3.1.0, 1.3.1.1, etc.

When Branches Are Created

Version Manager creates a branch when you do the following:

- Check in a locked, non-tip revision. Refer to the *Serena PVCS Version Manager User's Guide* for information about checking in revisions.
- Force a branch when checking in a locked tip revision. In the command-line interface, you force a branch by using the PUT -FB command. In the desktop client, you force a branch by selecting the Force Branch check box in the Check In dialog box. Refer to the *Serena PVCS Version Manager User's Guide* for more information about how to do this.
- Configure Version Manager for automatic branching. See the next section.
- Check in a revision with a *secondary lock*. This means that multiple users have checked out the same revision with a lock. The first user who locks the revision reserves the primary lock on the revision. The other users who have the revision locked must check in their revisions as branches. You can lock a revision more than once only if your project or project database is configured for multiple locks. See "Using Multiple Locks for Branching" on page 270.

Automatic Branching

Automatic branching lets you create a branch automatically from a trunk revision. Then, by default, you operate on the tip revision of that branch whenever you perform an action.

Without automatic branching, you must create a branch manually, either by locking a non-tip revision or forcing a branch when checking in a tip revision. You must also specify the branch tip revision each time you want to check it out and reassign the version label each time you check in a new revision.

Setting Up Automatic Branching

Before you set options for automatic branching, you must assign two version labels to the revision from which you want to branch. Assign a fixed version label to the appropriate revision to mark the branch point. Then assign another fixed version label to this revision (this becomes a floating label when the branch is created).

To set up automatic branching, you must define all three of the Branching configuration options. These options define the starting and ending points for the branch and specify the default revision on which to operate:

- **Base Version** specifies the version label that you assigned to mark the revision from which you want to start a branch (in the example above, Rel1.0 Base).
- **Branch Version** specifies the version label that you assigned to the tip of the branch (in the example above, Rel1.5 Branch). Initially, this will be the revision from which

you want to branch. As you check in subsequent branch revisions, this label will automatically reassign itself to the tip revision on the branch.

- **Default Version** identifies the version label you specified for the Branch Version option (in the example above, Rel1.5 Branch). This version label tells Version Manager which revision to operate on for all actions.

IMPORTANT! The rich IDE integration (Eclipse; Visual Studio) uses the default version (label) to determine which files are visible in a given Version Manager workspace. To avoid confusion, it is important that you understand how this works.

If you apply a default version or change the existing one for a project database or for a workspace, only files that have the version label will appear in the IDE client. If the project and solution files do not have these labels, you will see no files.

To avoid the potential for confusion:

- Create a Version Manager workspace for any user or group of users who may need their own default version (label).
- Define default versions on a workspace-by-workspace basis (File | Properties | Workspace Settings tab), rather than for the entire project database.

Whenever the version labels specified for the Base Version and Branch Version options refer to the same revision, Version Manager automatically begins a branch from that revision when you check in a file. Because you set the Default Version with the same version label as the Branch Version, you automatically check out the tip revision on the branch.

Using the Desktop Client

To set up automatic branching, you must have the Configure Project privilege assigned to you, and your project database or project must have a configuration file associated with it. If you are setting up automatic branching in a project, the master configuration file must allow a Base Version, Branch Version, and Default Version.

When you set the branching options in the configuration file associated with the project database or project, the same automatic branching definition is set up for all of the projects/subprojects in the database or project. Alternatively, you can set different automatic branching definitions in workspaces. The settings in a selected workspace override the settings in the configuration file, unless you disallow the branching options in the master configuration file. See "Using Workspaces" on page 174.

For example, let's say there are two development groups working on the same code: one group for Rel1.0 and one group for Rel1.5. In the beginning of the project the changes that were being made to Rel1.0 and Rel1.5 were co-existing in the same archives. Once developers needed to start adding new functionality to Rel1.5 that was not relevant to Rel1.0, it was time to start branching. The project database name for this product is SoftwareGenius. The Rel1.0 project leader must go through all of the archives and assigns a version label of Rel1.0 Base to the branch point (the revision where the branching will start). Next, she assigns a version label of Rel1.0 Branch Tip to the same revision.

When the version labels are in place for Rel1.0, the automatic branching options are added to the project database's configuration file. The project leader would assign Rel1.0 Base as the Base Version, Rel1.0 Branch Tip to the Branch and Default Versions.

Two workspaces should be created for the SoftwareGenius project database: Rel1.0 workspace, which will have the Default Revision as the trunk tip and Rel1.5 workspace,

which will have the Default Revision as the branch tip. With these two workspaces created, when a Rel1.0 developer needs to modify code, he sets his workspace to the Rel1.0 workspace. Then, when he selects a revision to check out, the revision associated with the version label Rel1.0 Branch Tip is checked out by default (he will be working on the branch).

If that same developer needs to make a quick change to the Rel1.5 code, he can set his workspace to Rel1.5. Then when he selects a revision to check out, the revision associated with the version label Rel1.0 Base Tip is checked out by default (he will be working on the trunk).

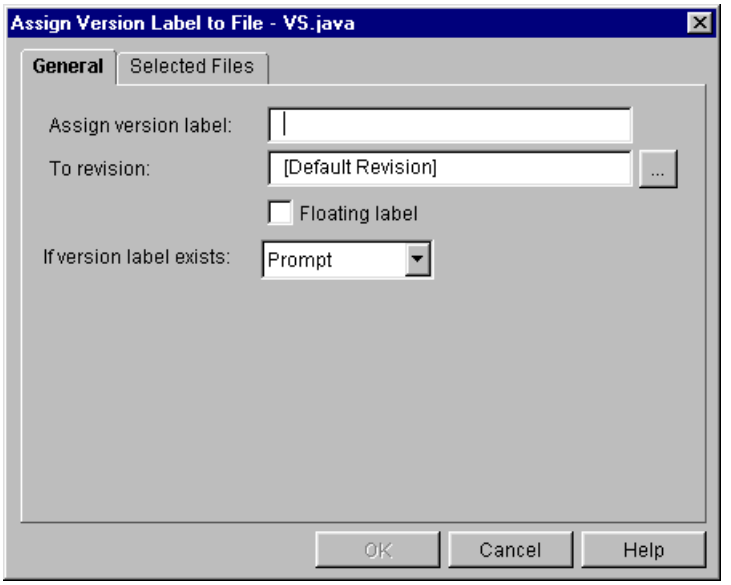
To set up automatic branching:

1 Select the project database or project for which you are setting up automatic branching.

2 Assign a fixed version label to the revision of each versioned file from which you want to start a branch.

a From the selected project, select the versioned file(s) to which you want to assign a version label. Note if you want to assign a version label to a different revision of the files, you must select each versioned file individually.

b Select Actions | Version Label | Assign. The Assign Version Label to File dialog box appears.



c Enter the name of the version label in the **Assign Version Label** field. For example, Rel1.0 Base. Any characters can be used in the name except: asterisks (*), colons (:), double quotes ("), plus signs (+), and minus signs (-).

d Enter the revision number in the **To revision** field or click the Browse button to select one.

e Click OK.

This version label will be defined later as the Base Version.

3 Assign another fixed version label to the revisions of the versioned files you selected in Step 2.

a Select the versioned file(s).

b Select Actions | Version Label | Assign. The Assign Version Label to File dialog box appears.

c Enter the name of the version label in the **Assign Version Label** field. For example, Rel1.5 Branch. Any characters can be used in the name except: asterisks (*), colons (:), double quotes ("), plus signs (+), and minus signs (-).

d Enter the revision number in the **To revision** field or click the Browse button to select one.

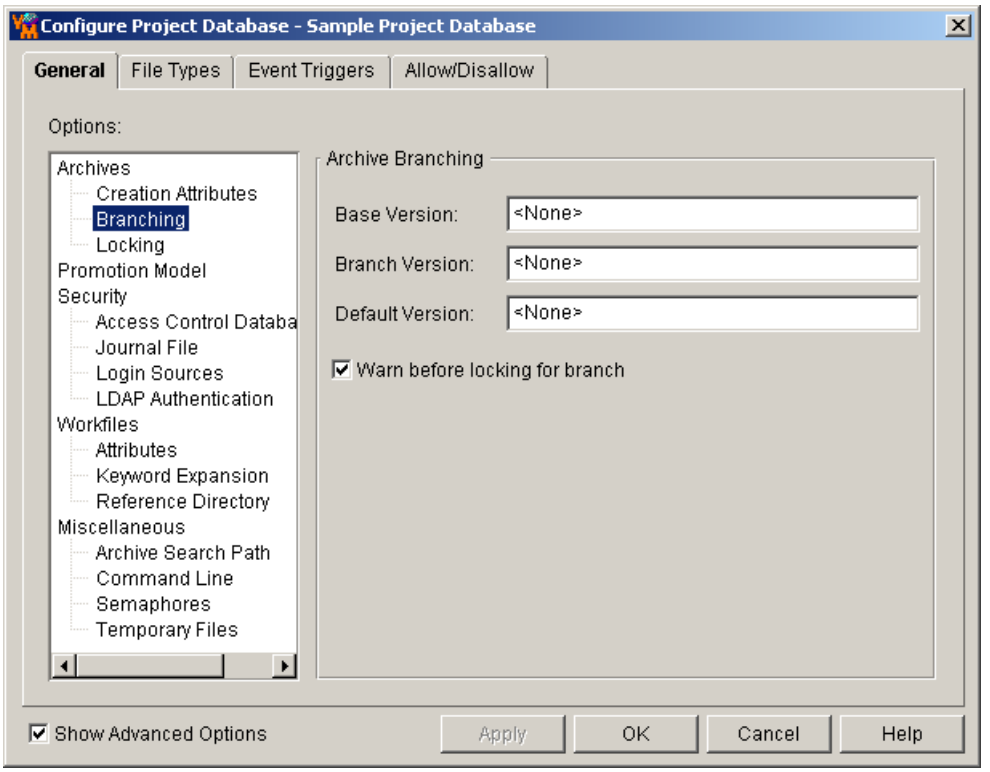
e Click OK.

This version label will be defined later as the Branch Version.

4 Select Admin | Configure Project. The Configure Project dialog box appears with the General tab active.

5 If not already selected, select the **Show Advanced Options** check box.

6 In the Options list, select Branching beneath Archives. The Archive Branching pane appears on the right.



7 Specify the Base version in the Base Version field, which is the version label you assigned in Step 2. For example, Rel1.0 Base.

8 Specify the Branch version in the Branch Version field, which is the version label you assigned in Step 3. For example, Rel1.5 Branch.

9 Specify the Default version in the Default Version field, which is the version label you assigned in Step 3. For example, Rel1.5 Branch.

10 Click OK.

Once you set up Version Manager for automatic branching, you can also automatically merge the branch tip with the trunk tip using the Actions | Show Merge command. From

the version labels you set up as branching options, Version Manager determines the revision from which the branch originated, the trunk tip revision, and the branch tip revision. Therefore, you do not have to specify these values when you perform a merge.

NOTE To later resume work on the trunk, reset the Default Version option to the version label you specified in Step 2. For example, Rel1.0 Base.

Using the Command-Line Interface

If you typically use the Version Manager command-line interface, you may want to edit configuration files using a text editor. If you find it easier, you can use the desktop client to configure Version Manager. Configuration files maintained in the desktop client are compatible with the command-line interface.

You can define the branching directives, `BaseVersion`, `BranchVersion`, and `DefaultVersion`, in either the master configuration file or a local configuration file. However, to define the directives in a local configuration file, the master configuration file must allow the directives.

To set up automatic branching:

1. Use the VCS -V command to assign a fixed version label to the revision in each archive from which you want to start a branch. For example:

   ```
   vcs -vRel1.0_Base:1.1 CVS.JAVA
   ```

   This version label will be defined later as the BaseVersion.

2. Use the VCS -V command to assign a fixed version label to the same revision you selected above. For example:

   ```
   vcs -vRel1.5_Branch:1.1 CVS.JAVA
   ```

   This version label will be defined later as the BranchVersion. It will reassign itself (float) to the tip revision on the branch.

3. Use a text editor to open the master configuration or local configuration file (vcs.cfg).

4. Set the BaseVersion directive to the version label you assigned in Step 1. For example:

   ```
   BaseVersion=Rel1.0_Base
   ```

5. Set the BranchVersion directive to the version label you assigned in Step 2. For example:

   ```
   BranchVersion=Rel1.5_Branch
   ```

6. Set the DefaultVersion directive to the version label you assigned in Step 2.

   ```
   DefaultVersion=Rel1.5_Branch
   ```

7. Save the configuration file and exit the editor.

Once you set up Version Manager for automatic branching, you can also merge the branch tip with the trunk tip automatically using the VMRG -A command. From the version labels

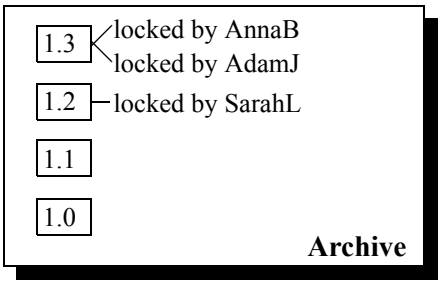
you set up as branching options, Version Manager determines which revisions to merge back on the trunk.

NOTE To later resume work on the trunk, open your configuration file and reset the DefaultVersion directive to the version label you specified in Step 1. For example, Rel1.0_Base.

Using Multiple Locks for Branching

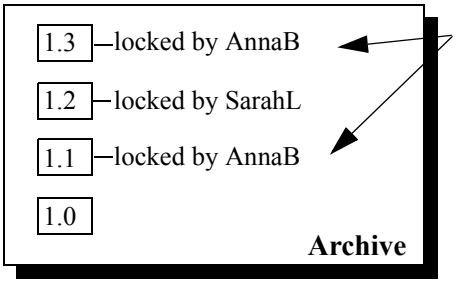
Multiple locking is designed for groups who maintain parallel development paths (branches). You can set options to allow:

- Multiple locks on a single revision (multiple locks per revision)



No user can have more than one lock in an archive, but multiple users can have locks on a revision.

- A single user to lock multiple revisions in an archive (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 simplify the creation of branches. If multiple locks per revision is enabled, different users can lock the same revision. Use this option if a user wants to create a branch from a revision that another user is working on, for example, if one user is working on a trunk revision, and another user is working on a branch from that revision.

If multiple locks per user is enabled, the same user can lock more than one revision in an archive. Use this option if the one user is working on two or more different lines of development of a file, for example, if the same user is working on a trunk and a branch revision of a file.

You can also use both options simultaneously if several users are working on the same two lines of development.

When you are ready to check in a file and a revision has been locked more than once, Version Manager creates a new branch unless you were the first person to create the lock. Before it creates the branch, it displays the revision numbers of the locked revisions and prompts you to specify the lock you have reserved. Version Manager then checks in your revision as a branch.

Setting Up Multiple Locks

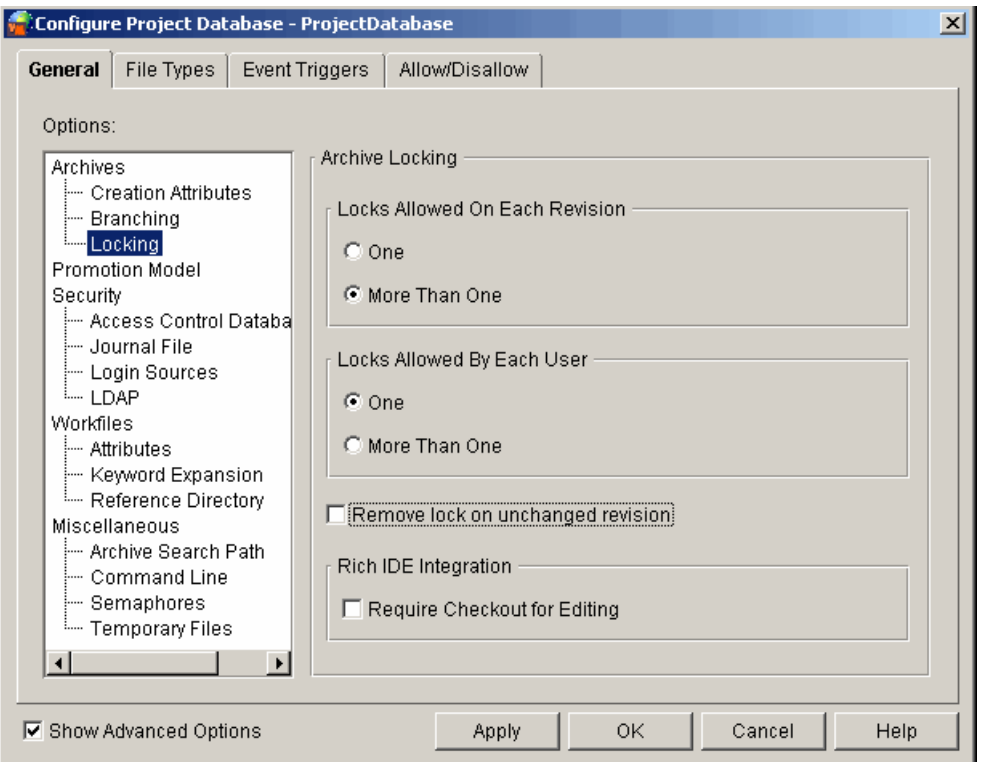
This section provides procedures for setting up multiple locks using the desktop client and the command-line interface.

Using the Desktop Client

To set up multiple locks, you must have the Configure Project privilege assigned to you, and your project database or project must have a configuration file associated with it. If you are setting up multiple locks in a project, the master configuration file must allow the Locks Allowed on Each Revision and Locks Allowed by Each User directives.

To set up multiple locks for newly created archives:

1. Select the project database or project for which you are setting up multiple locking. Once you complete this procedure, all of the archives in the project database or project will allow multiple locking.
2. Select Admin | Configure Project. The Configure Project dialog box appears with the General tab active.
3. If not already selected, select the **Show Advanced Options** check box.
4. In the Options list, select Locking beneath Archives. The Archive Locking pane appears on the right.



5. Do one or both of the following:
   - Select **More Than One** in the **Locks Allowed on Each Revision** group to allow multiple locks for each revision.
   - Select **More Than One** in the **Locks Allowed by Each User** group to allow individual users to lock more than one revision of an archive at a time.

6 Click OK.

NOTE After you have set multiple locking for all archives of a project, you can disallow multiple locking for some archives of a project by using the Archive Attributes Editor (Admin | Archive Attributes) and setting Exclusive Lock to Yes on the General tab.

To set up multiple locks for existing archives, you must use the Archive Attributes Editor (Admin | Archive Attributes). See "Changing Attributes of Existing Archives" on page 186.

If you want to ensure that users cannot change the settings for multiple locks, disallow the Locking options (directives) in the master configuration file. "Allowing and Disallowing Options" on page 101.

Using the Command-Line Interface

If you typically use the Version Manager command-line interface, you may want to edit configuration files using a text editor. If you find it easier, you can use the desktop client to configure Version Manager. Configuration files maintained in the desktop client are compatible with the command-line interface.

To set up multiple locks:

1 Use a text editor to open the master configuration or local configuration file (vcs.cfg).

2 Do any of the following:

- Place the `MultiLock` directive with no parameters in the configuration file to allow multiple locks per revision **and** to allow individual users to lock more than one revision of an archive at a time.
  ```
  MultiLock
  ```
- Place the `MultiLock` directive with the revision parameter in the configuration file to allow multiple locks per revision.
  ```
  MultiLock revision
  ```
- Place the `MultiLock` directive with the user parameter in the configuration file to allow individual users to lock more than one revision of an archive at a time.
  ```
  MultiLock user
  ```

3 Save the configuration file and exit the editor.

NOTE After you have set multiple locking for archives, you can disallow multiple locking for some archives by using the `VCS +PE` command.

Merging

Merging is the process of comparing the differences between two (or more) workfiles or revisions, accepting or rejecting the differences between them, and combining the changes into a new file.

While you can merge any two files or revisions, the most typical merge scenarios are:

- Merging a tip revision with a workfile. If you accidentally modify a workfile without locking a revision and then discover that another user has since locked that revision,

you can use merging to safely combine your edits with the other user's edits when they are checked in. Refer to the *Serena PVCS Version Manager User's Guide* for how to merge a tip revision with a workfile.

- Merging a branch with the trunk. Whether you use branches to maintain separate development paths or simply to accommodate multiple locking options, you can easily merge the branch tip with the trunk tip. If you have set up automatic branching to maintain your branches, you can merge these revisions automatically. See the next section, "Automatic Merging."

For desktop client users running Windows: The Version Manager desktop client on Windows uses a separate merge utility called the Merge Tool for all of its merging tasks. Windows users have the ability to perform three-way merges (a base file and two branch files). This is the same merge tool that is used with Serena Dimensions CM.

For desktop client users running UNIX and command-line interface users: The Version Manager desktop client on UNIX and the command-line interface use the standard Version Manager merge utility. Users have the ability to perform two-way (a base file and a branch file) merges.

Automatic Merging

NOTE To perform an automatic merge, you must have already set up archives for automatic branching.

Automatic merging is the process of merging a branch back to the trunk from archives that you have previously set up for automatic branching. Because you have set up automatic branching, the merge command can determine the revision from which the branch originated, the trunk tip revision, and the branch tip revision. This eliminates having to specify these values when you execute the merge command.

To merge files automatically, you use the `VMRG -A` command in the command-line interface or the Actions | Show Merge command in the desktop client. For further information about how to merge revisions, refer to the *Serena PVCS Version Manager User's Guide* and the *Serena PVCS Version Manager Command-line Reference Guide*.

Chapter 9

Using Event Triggers

Introduction

An *event trigger* is an administrator-specified program that is executed when a specified Serena PVCS Version Manager event occurs. An *event* is:

- A particular action that occurs during Version Manager processing, for example, checking in a file, assigning a version label, adding an entry to the journal file. Table 9-1, "Processing Events," on page 276 lists all Version Manager processing events.
- The click of a user-defined tool bar icon.
- The selection of a user-defined menu item.

Version Manager supports the following as event triggers:

- Stand-alone Windows (.EXE) and UNIX executable programs
- Windows batch scripts (.BAT)
- Windows program information files (.PIF)
- UNIX shell scripts

Version Manager supports both command-line and graphical applications. All nongraphical applications are executed through the shell or command interpreter: COMSPEC under Windows and SHELL under UNIX.

Some event triggers are used to verify new label syntax and author, broadcast a mail message to management when a project is promoted, run a build script after revisions are checked in, and copy journal file entries to a protected file for added security. Examples of event triggers and how to set them up are discussed in "Examples of Event Triggers" on page 290.

Version Manager Processing Events

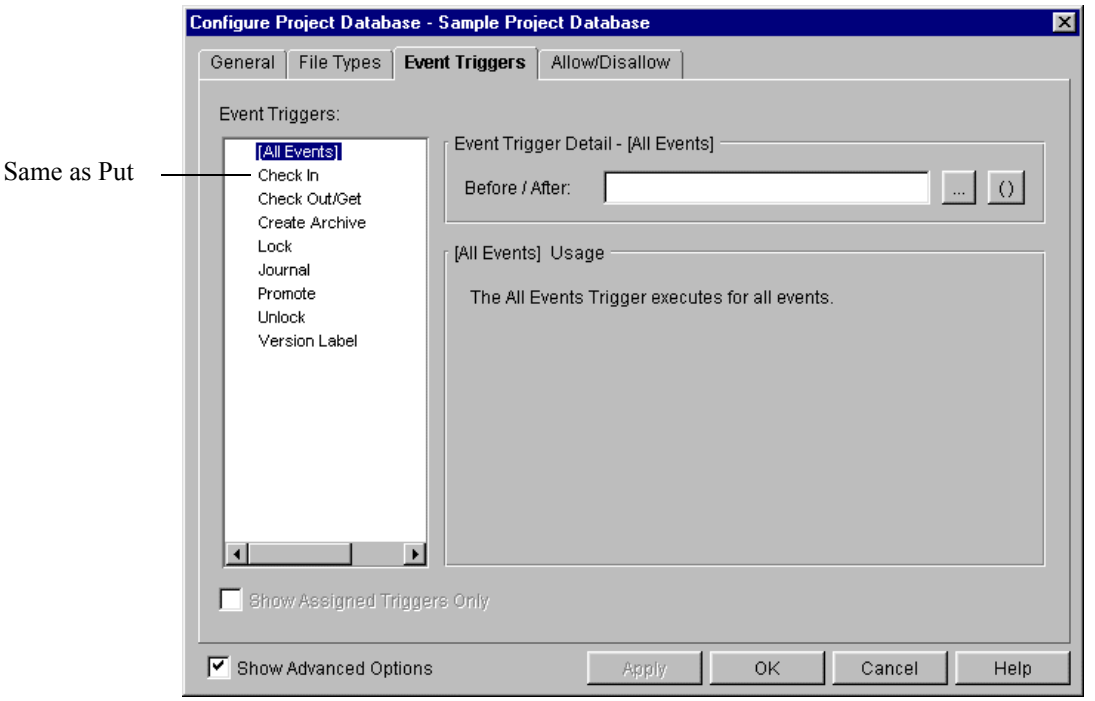
Table 9-1 defines Version Manager processing events. The event names presented in this table are used when defining event triggers in the command-line interface and are the names stored in project configuration files. The dialog box used in the desktop client to define event triggers is shown in Figure 9-1, "Desktop Client Event Names," on page 278.

Table 9-1. Processing Events

Event Name	Definition
AllEvents	All Version Manager events in this table.
PrePut	Just before checking in a workfile. This event occurs if there is nothing to prevent Version Manager from checking in the file.
PostPut	After a successful check in of a workfile.
UnconditionalPrePut	Before checking in a workfile. The workfile has not been read by Version Manager yet.
PreGet	Just before checking out a revision.
PostGet	After a successful check out of a revision.
PrePromote	Just before promoting a revision.

Event Name	Definition
PostPromote	After a successful promotion of a revision.
PreVersionLabel	Just before applying a version label.
PostVersionLabel	After applying a version label. This event does not occur when deleting a label.
PostJournal	When an entry is made to the journal file.
PreLock	Just before locking a revision. This event occurs whether or not a revision is checked out at the same time.
PostLock	After locking a revision. This occurs whether or not a revision is checked out at the time of the lock.
PreUnlock	Just before unlocking a revision. This event also occurs when checking in an unchanged workfile if: ■ The ForceUnlock directive in the command-line interface is in effect. ■ The Remove Lock on Unchanged Revision configuration option is set in the desktop client.
PostUnlock	After unlocking a revision. This event also occurs when checking in an unchanged workfile if: ■ The ForceUnlock directive in the command-line interface is in effect. ■ The Remove Lock on Unchanged Revision configuration option is set in the desktop client.
PreCreateArchive	Just before creating a new archive. This event also occurs before checking in an initial revision if: ■ The AutoCreate directive in the command-line interface is in effect. ■ The OK to Create New Archive on Check In configuration option is set in the desktop client.
PostCreateArchive	After creating a new archive. This event also occurs when checking in an initial revision if: ■ The AutoCreate directive in the command-line interface is in effect. ■ The OK to Create New Archive on Check In configuration option is set in the desktop client.

Figure 9-1. Desktop Client Event Names



Version Manager Icon and Menu Item Events

Icon and menu item events are only available in the Version Manager desktop client. In the Version Manager desktop client, you can add custom tools that appear as icons on the tool bar and/or as menu items on the Tools menu. This feature is useful when you want to create shortcuts for frequently used applications or tasks.

Refer to "Adding Custom Tools" on page 183 for complete information about how to define icon and menu item events.

Setting Up Event Triggers

This section provides instructions for setting up event triggers in the desktop client and the command-line interface.

Using the Desktop Client

To set up event triggers, you must have the Configure Project privilege assigned to you, and your project database must allow the Event Triggers configuration option. Otherwise, you cannot perform this procedure.

This procedure assumes you have an event trigger file defined. Supported event trigger files are identified on page 276.

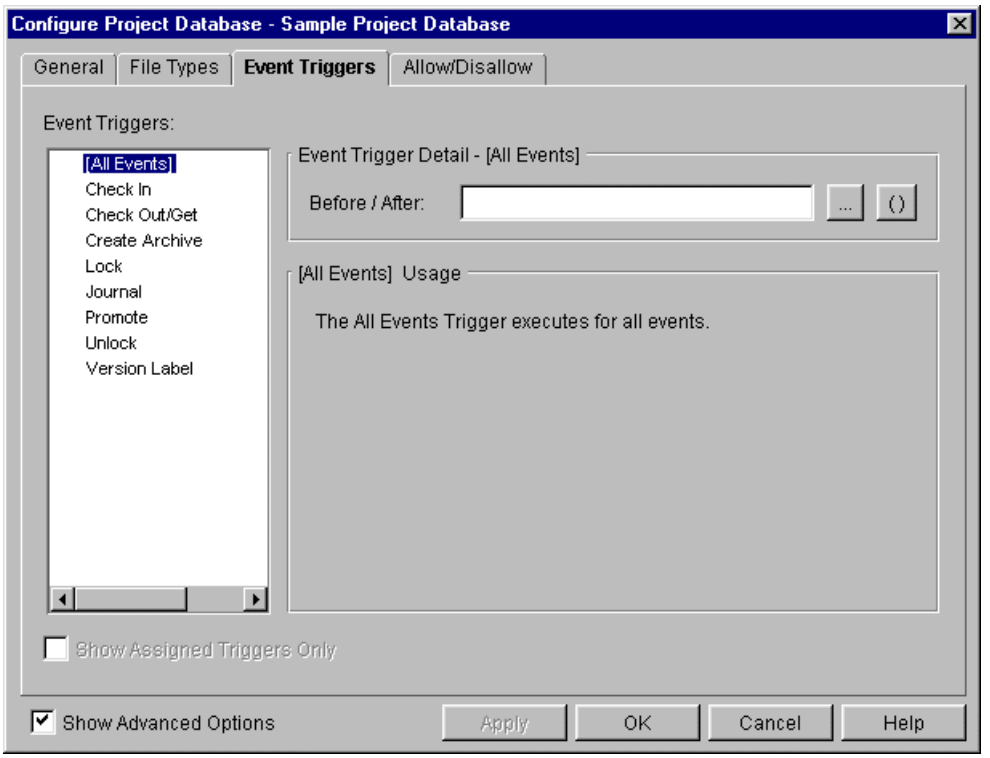
To set up an event trigger:

1 Decide on the event trigger to be executed before or after a Version Manager event.

2 Select the project database or project for which you are defining an event trigger.

3 Select Admin | Configure Project. The Configure Project dialog box appears.

4 Click the Event Triggers tab.



5 Select the event for which you want to define an event trigger in the **Event Triggers** list on the left.

6 Enter the path and the name of the event trigger file in the **Event Trigger Detail** group. For example, for Windows, you could enter `C:\BIN\POSTPUT.BAT` in the After field for a Check In event. You can also pass event information to an event trigger. See "Passing Information to Event Triggers" on page 280 for an explanation of how.

7 To set up another event trigger, first click Apply to save the definition of the event trigger you just defined, and then, repeat Steps 5 and 6. Otherwise, click OK.

Using the Command-Line Interface

If you typically use the Version Manager command-line interface, you may want to edit the configuration files using a text editor. If you find it easier, you can use the desktop client to set up event triggers. Configuration files with event trigger information that is maintained in the desktop client are compatible with the command-line interface.

This procedure assumes you have an event trigger file defined. Supported event trigger files are identified on page 276.

To set up an event trigger:

1. Decide on the event trigger to be executed before or after a Version Manager event.
2. In a text editor, open the configuration file in which you want to add an event trigger.
3. Add event triggers. Each event trigger must be on a separate single line.

   `EventTrigger [=] event_name [event_trigger] [event_information]`

   where:

   event_name is any event listed in Table 9-1, "Processing Events," on page 276.

   event_trigger specifies the path and the name of the event trigger file. For example, for Windows, `C:\BIN\POSTPUT.BAT` or for UNIX, `/usr/bin/postput`.

   event_information is explained in "Passing Information to Event Triggers" on page 280.

   For example:

   Windows `EventTrigger = AllEvents X:\PVCS\EVENT\ALL.BAT`

   UNIX `EventTrigger = AllEvents /usr/pvcs/event/all`

4. Save the configuration file and exit the editor.

Passing Information to Event Triggers

In the desktop client and the command-line interface, you can pass information about the archive being acted upon to an event trigger. This information can be used to collect information about the archive and to validate that the event trigger is acting upon the correct archive.

In the desktop client, you can also pass information about the project being acted upon to an event trigger.

For example, suppose you want e-mail to notify you when someone checks in a revision. Therefore, you create a program that writes an e-mail notify message, and you configure Version Manager to run the program when a new revision is checked in. The program you write can access information that contains the user ID, name of the archive and workfile, revision number, and time. The program will format this information into a text file and instruct the e-mail importer to send the file as a message. The program you write is the event trigger. The event is PostPut.

Table 9-2 lists the information available to event triggers. Some of this information is only available from some of the events, as shown in , "Event Information Availability," on page 283.

Table 9-2. Event Information

Event Information	Definition
EventArchive	A fully qualified path, including the name of the archive.
EventArchiveName*	The name of the archive.

Event Information	Definition
EventChgDesc	The change description. Limited to 1K in size except when used with the parameter file method of passing event information (see "Parameter File Method" on page 288).
EventClientWorkfile*	A fully qualified path, including the name of the workfile on the client or full path if stand-alone.
EventConfigFiles*	Fully qualified paths, including the names of the configuration files listed in the order that Version Manager reads them. This list does not include the master configuration file, if any. Values are separated by semicolons (;).
EventDate	The date when the event trigger was executed. The format of the date is packed decimal numeric. For example, January 20, 2003 appears as 20030120.
EventEntityPath*	A PCLI compatible entity path for the entity the event trigger or toolbar is operating on (\<project>\<subproject>\...\<file>).
EventFolderPath*	Folder, versioned file, revision, version label, or promotion group in a 5.3/6.0 folder. This only applies to 5.3/6.0 folders and not to projects.
EventFQPWorkfile	A fully qualified path, including the workfile name.
EventGroup	The promotion group assigned to the revision. If the event is part of an operation that assigns or changes a promotion group, the new or promoted group name appears; otherwise, if the versioned item was selected using a promotion group, that group name appears.
EventItemSelectionType*	How the item being processed was selected. The possible values are ProjectDB, Project, Folder, File, Revision, VersionLabel, and Group (promotion group).
EventJournalEntry	The journal entry line that was written to the journal file.
EventJournalFile	The name of the journal file.
EventLock	The value is YES if a lock was requested and is not defined otherwise.
EventMasterConfigFile*	A fully qualified path, including the name of the master configuration file.
EventName	The name of the event. The value for EventName will always be ButtonEvent for icon and menu item events.
EventParmFile	A pathname to a temporary file that contains all <variable>=<value> pairs for event trigger information. Intended for programs that want to get their information out of a file instead of reading environment variables or arguments that are passed into the program.
EventPassword*	A limited-lifetime encrypted password (valid for 24 hours) that is based on the password used to log into the project database. This provides authentication for use with a file server.

Event Information	Definition
EventProjectDB*	A fully qualified path, including the name of the project database being used.
EventProjectName*	A fully qualified path, including the name of the project in the project database.
EventRevision	The revision number being operated on.
EventTime	The time when the event trigger was executed. The format of the time is packed decimal numeric. For example, 2:35:26 P.M. appears as 143526.
EventUserID	The user ID of the person performing the operation.
EventVersion	The version label. If the event is part of an operation that assigns or changes a version label, the new label appears; otherwise, if the versioned item was selected using a version label, that label appears.
EventWorkfile	The name of the workfile.
EventWorkspaceBase*	The Base Version defined for the current workspace.
EventWorkspaceBranch*	The Branch Version defined for the current workspace.
EventWorkspaceName*	The name of the current workspace. In the case of the Root Workspace, the name value will be blank.
EventWorkspacePromoGrp*	The Default Promotion Group defined for the current workspace.
EventWorkspaceVersion*	The Default Version defined for the current workspace.

* Not available in triggers executed from CLI commands.

Table 9-3. Event Information Availability

VM Events	AllEvents	PreGet	PostGet	Unconditional PrePut	PrePut	PostPut	PostJournal	PrePromote	PostPromote	PreVersionLabel	PostVersionLabel	PreLock	PostLock	PreUnlock	PostUnlock	PreCreateArchive	PostCreateArchive	Icon/Menu
EventArchive	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	□
EventArchiveName*	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	□
EventChgDesc	□				□	□												
EventClientWorkfile*	□	■	■	■	■	■	□	■	■	■	■	■	■	■	■			□
EventConfigFiles*	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	□
EventDate	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
EventEntityPath*	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■ 1	■ 1	■
EventFolderPath*	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
EventFQPWorkfile	□	■	■	■	■	■												
EventGroup	□	□	□	□	□	□	□	■	■	□	□	□	□	□	□			□
EventItemSelectionType*	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
EventJournalEntry	□	□	□	□			■											
EventJournalFile	□						■											
EventLock	□	□	□	□	□	□												
EventMasterConfigFile*	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
EventName	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
EventParmFile	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
EventPassword*	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
EventProjectDB*	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
EventProjectName*	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	□
EventRevision	□	■	■	□	■	■	□			■	■	■	■	■	■			□
EventTime	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
EventUserID	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
EventVersion	□	□	□	□	□	□	□			■	■	□	□	□	□			□
EventWorkfile	□	■	■	■	■	■				■	■							□
EventWorkspaceBase*	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
EventWorkspaceBranch*	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
EventWorkspaceName*	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
EventWorkspacePromoGrp*	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
EventWorkspaceVersion*	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■

■ This event information is always available.

□ This event information is available when appropriate. For example, group information

(EventGroup) is always available for PrePromote or PostPromote but only available when specified or highlighted for Get or Put.

1 The value is available to the event trigger, but cannot be used to access the entity from PCLI because the archive has not yet been bound to a project.

* Not available in triggers executed from CLI commands.

Event Information for Icon and Menu Item Events

The event information available from icon and menu item events depends on which items are selected in the desktop client when the event takes place. For example, EventArchive event information is available to an event trigger if a versioned file, revision, version label, or promotion group is selected in the desktop client when an icon or menu item event takes place.

The following table lists each type of event information and the item that must be selected in the desktop client for the event information to be made available to an event trigger. If an item is not selected, Version Manager returns no corresponding information to the event trigger.

For example, if you define an event trigger that asks for the name of a workfile (EventWorkfile), but a versioned file, revision, version label, or promotion group is not selected when the event trigger is executed, the name of the workfile will not be passed to the event trigger.

Event Information	Items that must be selected in the desktop client
EventArchive	Versioned file, revision, version label, or promotion group
EventArchiveName	Versioned file, revision, version label, or promotion group
EventWorkfile	Versioned file, revision, version label, or promotion group
EventUserID	Project database or lower
EventRevision	Revision
EventVersion	The version label
EventGroup	The promotion group you are promoting from
EventDate	Project database or lower
EventTime	Project database or lower
EventConfigFiles	Project database or lower
EventMasterConfigFile	Project database or lower
EventClientWorkfile	Versioned file, revision, version label, or promotion group
EventProjectDB	Project database or lower
EventProjectName	Project or lower
EventFolderPath	Folder, versioned file, revision, version label, or promotion group in a 5.3/6.0 folder. This only applies to 5.3/6.0 folders and not to projects.
EventItemSelectionType	Project database or lower
EventName	Project database or lower. The value for EventName will always be ButtonEvent for icon and menu item events.

Event Information	Items that must be selected in the desktop client
EventWorkspaceBase	Project database or lower
EventWorkspaceBranch	Project database or lower
EventWorkspaceName	Project database or lower
EventWorkspacePromoGrp	Project database or lower
EventWorkspaceVersion	Project database or lower

How Version Manager Passes Event Information

Version Manager passes event information through:

- Environment variables
- Command-line macros
- Parameter files

If event information is passed through environment variables, the event trigger queries the environment for values it needs. In the situations where event information cannot be passed using environment variables, use either the command-line macros or parameter file method.

Not all methods of passing event information work for all event triggers, and not all event triggers can be used for abortable events—all Pre events are abortable; all Post events are not. For example, you could write a PrePut event trigger to verify that the program being checked in has been compiled correctly without syntax errors or warnings. If the compile fails, the event trigger could return a nonzero exit code that would instruct Version Manager to terminate the process. In this case, Version Manager would terminate the check in (Put).

Environment Variable Method

If event information is passed through environment variables, the event trigger queries the environment for values it needs. For example, a batch file would query `%EVENTARCHIVE%`, a UNIX shell script would query `$EVENTARCHIVE`, a build script would query `$(EVENTARCHIVE)`, and a C program would perform a `getenv("EVENTARCHIVE")` function call.

Windows An example of a batch file named POSTPUT.BAT:

```
H:\PUBLIC\SEND "Just checked in %EVENTWORKFILE%, revision
    %EVENTREVISION% to group %EVENTGROUP%"
```

The event trigger

```
POSTPUT C:\BIN\POSTPUT.BAT
```

sends a message to a work group after Version Manager successfully checks in a workfile.

UNIX An example of a UNIX postput shell file:

```
/usr/pvcstools/send "Just checked in $EVENTWORKFILE, revision
    $EVENTREVISION to group $EVENTGROUP"
```

The event trigger

```
POSTPUT /usr/bin/postput
```

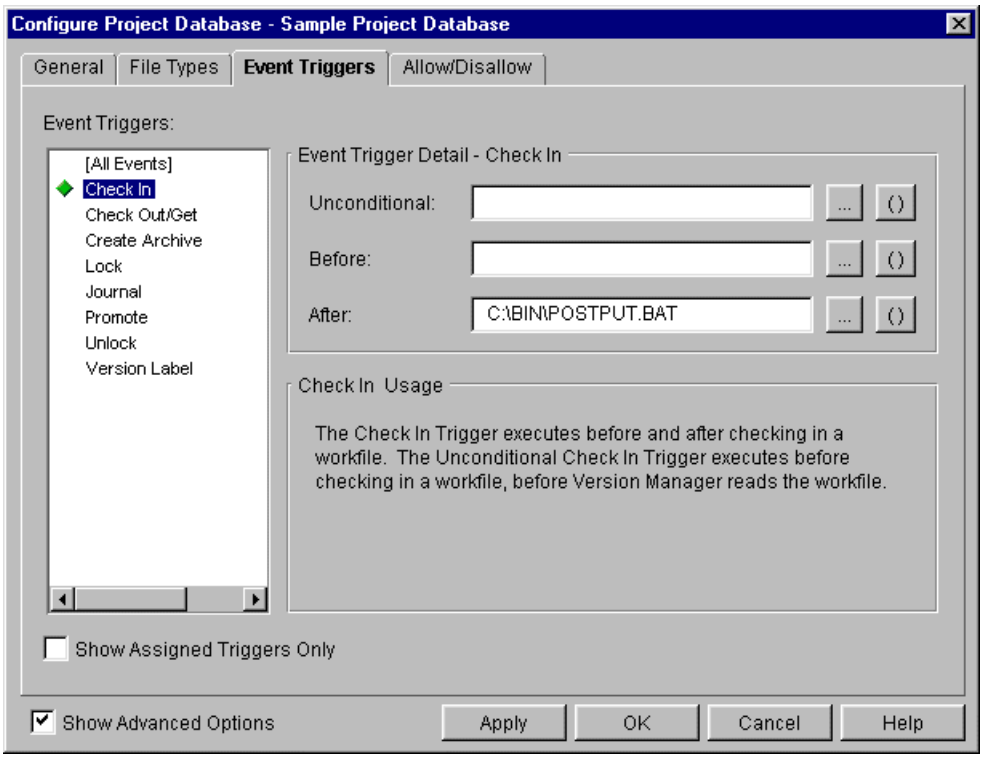
sends a message to a work group after Version Manager successfully checks in a workfile.

Specifying in the Desktop Client

In the **Event Trigger Detail** group of the Configure Project dialog box (Admin | Configure Project | Event Triggers tab), enter the path and the name of the event trigger file. The environment variables are specified in the event trigger file. For example, select the Check In event and enter the following in the After field:

Windows `C:\BIN\POSTPUT.BAT`

UNIX `/usr/bin/postput`



Specifying in the Command-Line Interface

Using a text editor, add an event trigger definition to the appropriate configuration file. The environment variables are specified in the event trigger file. For example:

```
EventTrigger = POSTPUT C:\BIN\POSTPUT.BAT
```

Command-Line Macro Method

An event trigger definition may reference event information through macros that are specified on the command line (not in the event trigger file). These command-line macros

have the same name as the event information names listed in Table 9-2, "Event Information," on page 280. The names are specified by surrounding the names with two underscore characters (__), for example, __EVENTARCHIVE__.

The following example illustrates an event trigger definition that calls Configuration Builder and defines a macro named ARCHIVE that contains the name of the archive.

```
EventTrigger = PREPUT Build.exe -FMYEVENT.BLD ARCHIVE=__EVENTARCHIVE__
```

When Version Manager executes the PrePut event, the command-line executable expands to:

```
Build.exe -FMYEVENT.BLD ARCHIVE=K:\SOURCE\PVCS\LOGFILES\GET.C-ARC
```

For UNIX command-line interface users: All command line macros whose expansion might include spaces or other special characters must have the macro name surrounded by either single or double quotes. For example, when you pass `EventJournalEntry` information using the command-line macro method (`__EventJournalEntry__`) to a `PostJournal` event, you must surround the macro name with either single or double quotes (`'__EventJournalEntry__'`). The information passed by `EventJournalEntry` can contain parentheses, which are special characters on the UNIX command line.

The following macros do **not** need quotes:

- `__EventName__`
- `__EventRevision__`
- `__EventDate__`
- `__EventTime__`
- `__EventParmFile__`

Specifying in the Desktop Client

For processing events

In the **Event Trigger Detail** group of the Configure Project dialog box (Admin | Configure Project | Event Triggers tab), enter the event trigger definition, including the command-line macro that passes the event information. For example, select the Check In event and enter the following in the **Before** field:

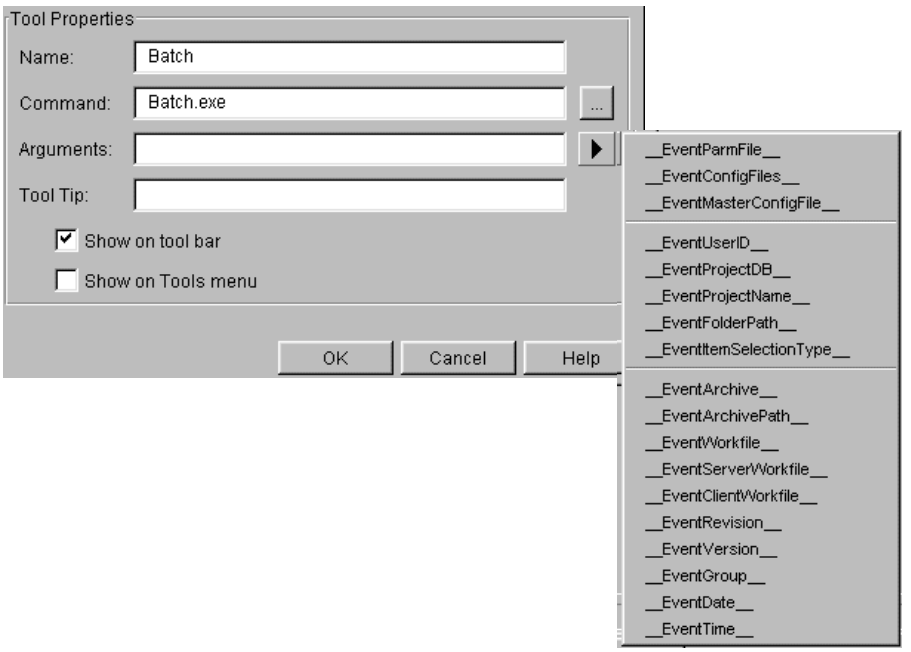
```
Build.exe -FMYEVENT.BLD ARCHIVE=__EVENTARCHIVE__
```

NOTE To pass a command-line macro, you can click the ( ) button next to the Before, After, or Unconditional field and select the macro.

For icon and menu item events

In the **Arguments** field of the Tool Configuration dialog box (Admin | Tool Configuration), click the button next to the **Arguments** field to select a command-line macro to pass event information to the executable.

For example:



Specifying in the Command-Line Interface

Using a text editor, add an event trigger definition to the appropriate configuration file, including the command-line macro that passes the event information. For example:

In Windows:

```
EventTrigger = POSTPUT = E:\WD\POST.BAT __EVENTFQPWORKFILE__
```

On UNIX:

```
EventTrigger = POSTPUT = /usr/wd/post __EVENTFQPWORKFILE__
```

Parameter File Method

The parameter file method of passing event information can be used by specifying the macro __EventParmFile__ on the command line (not in the event trigger file). The parameter file method of passing event information is the slowest. Use it only if the environment variable method does not work for your particular event trigger or if a change description is longer than 1K.

Using this method, Version Manager creates a temporary file that contains values for event information that is available for the event trigger being executed. This information can then be used by the event trigger. Remember that not all event information is available for all Version Manager events.

The following example illustrates the use of __EventParmFile__. The example shows an event trigger definition that calls Configuration Builder and creates a temporary file in the current working directory. Version Manager writes the name and value of each macro definition in the syntax *name=value*.

```
EventTrigger = ALLEVENTS Build.exe -FMYEVENT.BLD EventInfoFile=__EventParmFile__
```

Continuing with the previous example, when Version Manager executes any Version Manager event, the command-line executable expands to the following:

```
Build.exe -FMYEVENT.BLD EventInfoFile=E:\TMP\PVCS0024.TMP
```

Version Manager creates the temporary file, executes the event trigger, and then removes the file. The contents of the temporary file are placed in EventInfoFile for use by Configuration Builder.

Where, for example, the file `E:\TMP\PVCS0024.TMP` contains the following:

```
EVENTNAME=PostPut
EVENTARCHIVE=E:\TMP\ET\FOO.C_V
EVENTWORKFILE=FOO.C
USERID=BLAIR
EVENTLOCK=
EVENTREVISION=1.1
EVENTVERSION=
EVENTGROUP=
EVENTJOURNALFILE=
EVENTJOURNALENTRY=
EVENTDATE=19990123
EVENTTIME=112104
EVENTFQPWORKFILE=D:\TMP\ET\FOO.C
EVENTCONFIGFILES=
EVENTMASTERCONFIGFILE=E:\PRJDB\ARCHIVES\CPE2QP.CFG
EVENTARCHIVEPATH=
EVENTCLIENTWORKFILE=
EVENTPROJECTDB=
EVENTPROJECTNAME=
EVENTITEMSELECTIONTYPE=
EVENTFOLDERPATH=
EVENTCHGDESC=No change.
```

Specifying in the Desktop Client

For processing events

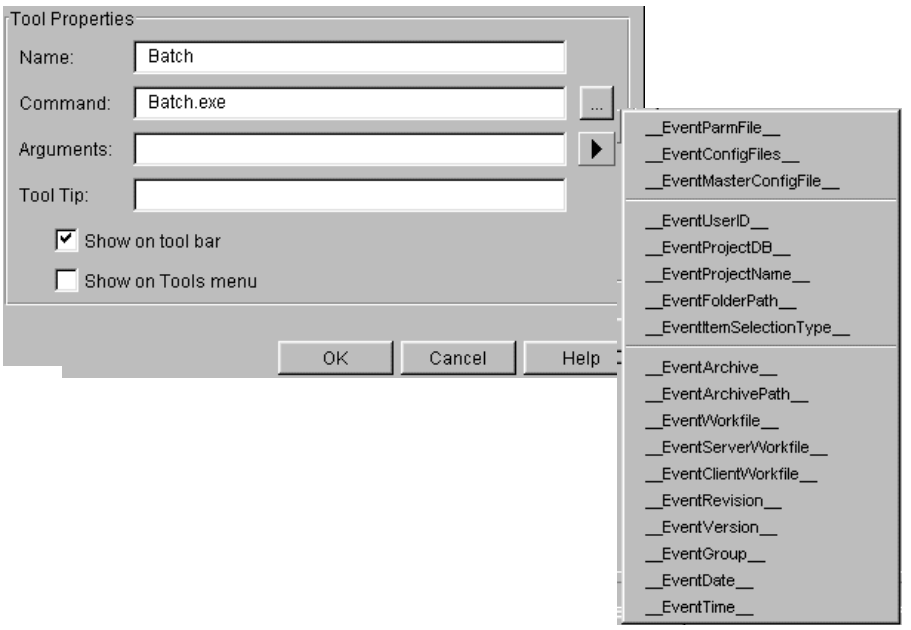
In the **Event Trigger Detail** group of the Configure Project dialog box (Admin | Configure Project | Event Triggers tab), enter the event trigger definition along with the event

information as a parameter file. For example, select the AllEvents event and enter the following in the Before/After field:

```
Build.exe -FMYEVENT.BLD EventInfoFile=__EventParmFile__
```

For icon and menu item events

In the **Arguments** field of the Tool Configuration dialog box (Admin | Tool Configuration), click the button next to the **Arguments** field and select __EventParmFile__ in the drop-down list that appears.



Specifying in the Command-Line Interface

Using a text editor, add an event trigger to the appropriate configuration file, including the event information as a parameter file. For example:

In Windows:

```
EventTrigger = ALLEVENTS E:\PVCS\EVENT\ALL.EXE EventInfoFile=__EventParmFile__
```

On UNIX:

```
EventTrigger = ALLEVENTS /usr/pvcs/event/all EventInfoFile=__EventParmFile__
```

Examples of Event Triggers

Example 1: Creating a PostPut event trigger to place a floating version label on any revision checked in.

The following event trigger places the floating label "LATEST" on **any** revision that is checked in, including branches. If you want to have a floating label on trunk revisions only, then you do not need an event trigger. Instead, place the floating label on the trunk of all archives in the project. The definition of this event trigger in a configuration file is:

```
EventTrigger = POSTPUT vcs -vLATEST:__EventRevision__* -y __EventArchive__
```

Note that this event trigger is being passed event information through command-line macros.

Example 2: Creating a PrePromote event trigger to get (check out without a lock) a revision being promoted and place it in a specific directory.

The following event trigger will get a revision that is being promoted and place it in one directory if it is being promoted to the Stage promotion group and another directory if it is being promoted to the Production promotion group. The definition of this event trigger in a configuration file is:

```
EventTrigger = [path]prepromote.bat __EventGroup__ __EventArchive__
```

The prepromote.bat file looks like this:

```
@echo off
rem %1 = EventGroup and %2 = EventArchive
IF "%1" == "Development" GOTO STAGE
IF "%1" == "Stage" GOTO PRODUCTION
:STAGE
get -cD:\PVCS\PVCSPROJ\prmsmple.prj\vcs.cfg
   -gDevelopment %2(c:\staging\famis
     :PRODUCTION
get -cD:\PVCS\PVCSPROJ\prmsmpl.prj\vcs.cfg
   -gStage %2(c:\production\famis)
```

Example 3: Creating an AllEvents event trigger to echo the event information being passed for all events.

The following event trigger will display the event information that is available for each Version Manager event in a console window. This event trigger is useful for checking which information is passed during which events. The definition of this event trigger in a configuration file is:

```
EventTrigger = [path]echovm.bat
```

The echovm.bat file uses environment variables to get event information and looks like this:

```
echo off
echo Action %EventName% on %EventDate% %EventTime%
echo.
echo EventName:%EventName%
echo EventArchive:%EventArchive%
echo EventWorkfile:%EventWorkfile%
echo EventUserID:%EventUserID%
echo EventLock:%EventLock%
echo EventRevision:%EventRevision%
echo EventVersion:%EventVersion%
echo EventGroup:%EventGroup%
```

Chapter 10

Using Reports

Introduction

Serena PVCS Version Manager provides three types of reports:

- Journal reports, which contain information about actions that modify an archive.
- History reports, which contain information about archives—the archive and workfile names, the archive description, revision numbers, revision history, and more. Note that history reports were called archive reports in previous releases of Version Manager.
- Security reports, which contain the security settings in the access control database.

With the desktop client, 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. You can customize the format of HTML reports.

NOTE If you use an HTML browser to display a history report that is generated for more than 400 files, the browser will take a minute or two to display the generated report. For quicker response, use a text editor to display the history report.

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 and the command-line interface.

Change log

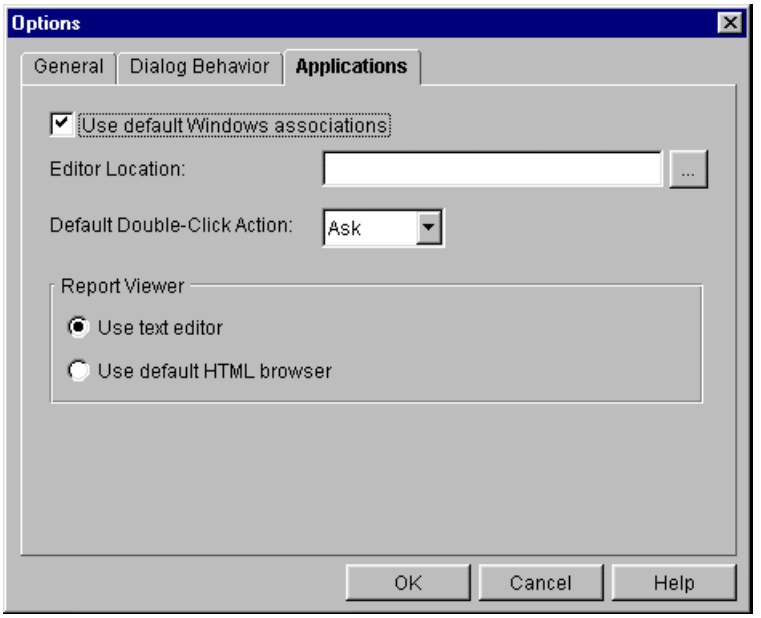
In addition to these reports, Version Manager records changes to the contents of projects, such as adding or removing subprojects or versioned files. These changes are recorded in a `change.log` file located in the root directory of the project database. For more information, see "Viewing Changes to Projects (change.log)" on page 309.

Setting Report Options

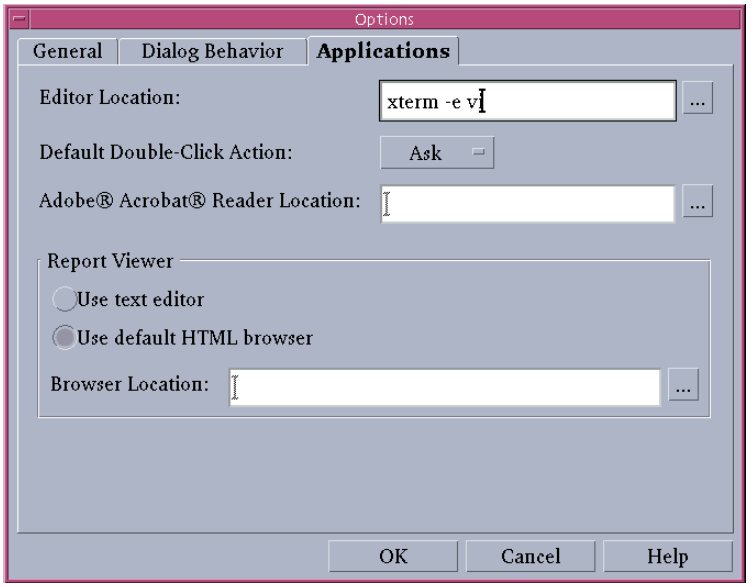
To set report options on Windows and UNIX platforms:

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

- Use default HTML browser



NOTE On UNIX, 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, history, and security reports. There is one template file per type of report—

journal.template, history.template, and accessdb.template. By default, these files are placed in the following location during Version Manager installation:

- In Windows: *drive*:\Program Files\Serena\vm\common\pvcsprop\pvcs\vm
- On UNIX: /usr/serena/vm/common/pvcsprop/pvcs/vm

All three 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">
</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).

Generating Journal Reports

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.

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. If Version Manager is not configured to keep a journal file, you cannot generate a journal report. See "Journal File Options" on page 72 for an explanation of how to configure Version Manager to keep a journal file.

You will find a journal report 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.

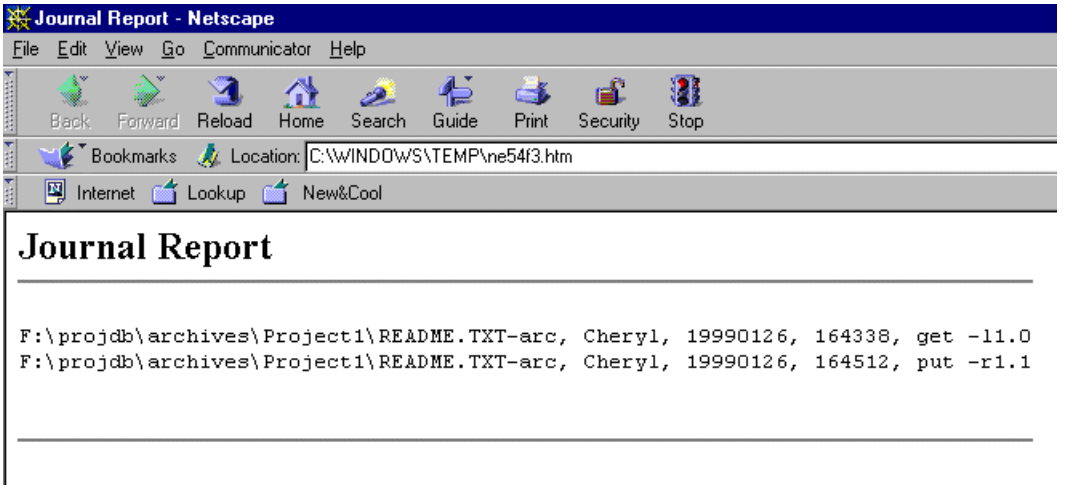
Using a journal report is faster than generating a history report because Version Manager does not have to open the archives before generating a journal report—the data is already in a journal file. If you want general information about an archive, such as the date of creation, revision information, or access restrictions on the archive, generate a history report. See "Generating History Reports" on page 301.

You can set many different options before you generate a journal report that will limit the contents of the report. You can specify to 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
- Revisions currently locked by specified users

How to Read a Journal Report

The following is a sample journal report generated in the desktop client and displayed in an HTML browser:



Each journal report line has the following format:

```
archive, user_id, date, time, action
```

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 modification in packed decimal numeric form. For example, March 31, 2003 appears as 20030331.
- *time* is the time of the modification in packed decimal numeric form (using a 24-hour clock). For example, 4:55:20 P.M. would be 165520.
- *action* is the command-line option of the action that was taken. For example, put -r1.1, which means revision 1.1 was checked in.

Using the Desktop Client

In the desktop client, you can generate a journal report for a project database, project, multiple versioned files, or a single versioned file. When you select a project database or project, you have the option of generating the report for:

- All of the versioned files of all of the projects/subprojects within the project database or project
- Just the versioned files within the selected project database or project—not including the projects and/or subprojects

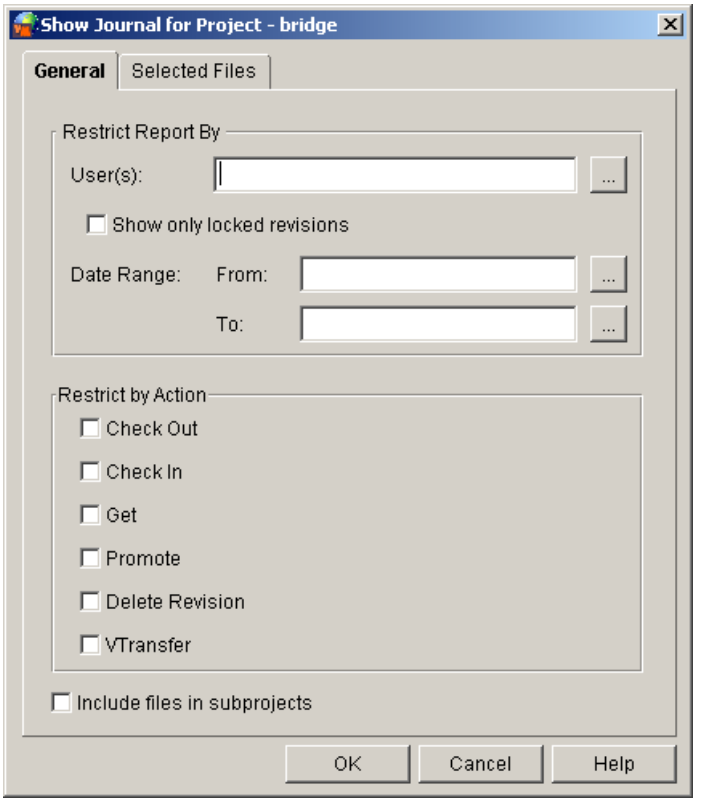
The journal file used to generate the report is the one specified in the configuration file associated with the item you select. If a journal file is not specified in the configuration file for the selected item, a journal report cannot be generated.

You must have the Journal Report privilege to generate a journal report.

To generate a journal report in the desktop client:

1 Select a project database, project, or versioned file(s).

2 Select Actions | Show Journal. The Show Journal dialog box appears with the General tab active.



NOTE If a versioned file is selected in the File pane, the **Includes files in subprojects** check box is not displayed.

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. In 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.
Changes made in a certain range (cont.)	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.
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 workfiles gotten	Select the **Get** 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.
Only archives that have been renamed or deleted on a Version Manager File Server	Select the **VTransfer** 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 To review the selected versioned files before you generate the journal report, click the Selected Files tab.

5 Click OK. The journal report is generated and displayed.

Using the Command-Line Interface

You must have the JournalReport privilege to generate a journal report.

To generate a journal report using the command-line interface, use the VJOURNAL command. The syntax for this command is:

`vjournal [option...] [journal_file...]`

where *journal_file* is the journal file from which to get the information for the report. If you do not specify a journal file, Version Manager uses the journal file specified by the Journal directive in either the configuration file you specify in the command line or the master configuration file that is embedded into Version Manager. If the Journal directive does not specify a file, Version Manager searches the directories named by the VCSDir directive and uses all of the journal files it finds to generate the report. The VCSDir directive specifies the archive directories, and by default, the journal files are placed in the archive directories.

The following table lists frequently used options for this command.

To view. . .	Use this command line. . .
Changes made between certain dates	`vjournal -d`*date_range*
Changes made to certain archives	`vjournal -l`*archive*[,*archive*]
Activity by certain commands	`vjournal -o`*command*[,*command*]
Changes made by certain users	`vjournal -u`*user_id*[,*user_id*]
Currently locked revisions	vjournal -xl

See the *Serena PVCS Version Manager Command-Line Reference Guide* for complete information about the VJOURNAL command.

Generating History Reports

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 that history reports were called archive reports in previous releases of Version Manager.

History reports can include archive information and revision information. Archive information is:

- When and by whom an archive was created
- Archive attributes
- Who has revisions locked
- Archive and workfile names

Revision information is:

- Revision descriptions
- Revision history

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

There are two base privileges that dictate if you can generate a history report and the information that is available for the report:

- ViewArchiveHeader privilege. You must have this privilege to generate a history report that a contains only archive information, but not revision information.
- ViewArchiveRev privilege. You must have this privilege to generate a history report that contains revision information.

How to Read a History Report

The following is a sample history report generated in the desktop client and displayed in text format. This sample 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.

NOTE If NoGenerateDelta is set for the archive, the value for "lines deleted/added/ moved:" will always be 0/0/0.

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

Using the Desktop Client

In the desktop client, you can generate a history report for a project database, multiple projects, a single project, multiple versioned files, a single versioned file, or a specific revision. When you select a project database or project, you have the option of generating the report for:

- All of the versioned files of all of the projects/subprojects within the project database or project
- Just the versioned files within the selected project—not including the projects and/or subprojects

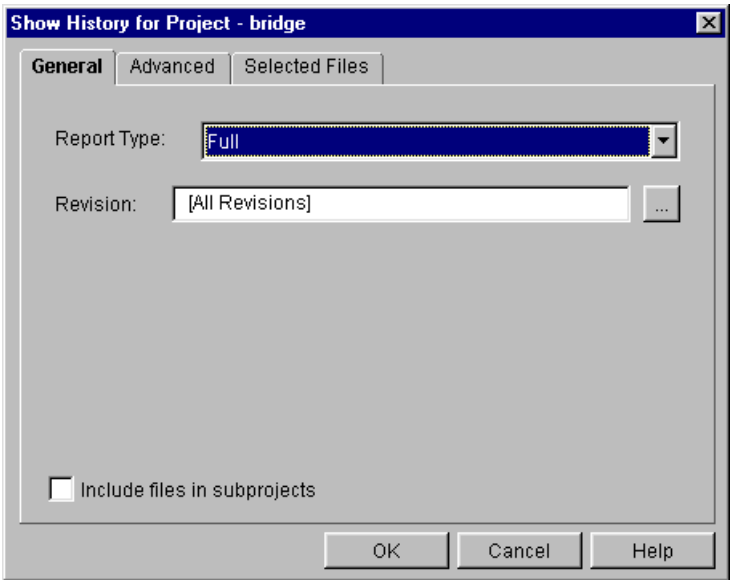
NOTE You cannot generate a history report for a 5.3/6.0 project. You can generate the report on the contents of the project, but not the project itself.

To generate a history report:

1 Select the project database, project(s), versioned file(s), or a specific revision.

To select multiple, adjacent projects, click on the first project, hold down the SHIFT key, and then click on the last project in the series. To select multiple, non-adjacent projects, hold down the CTRL key while clicking on the projects.

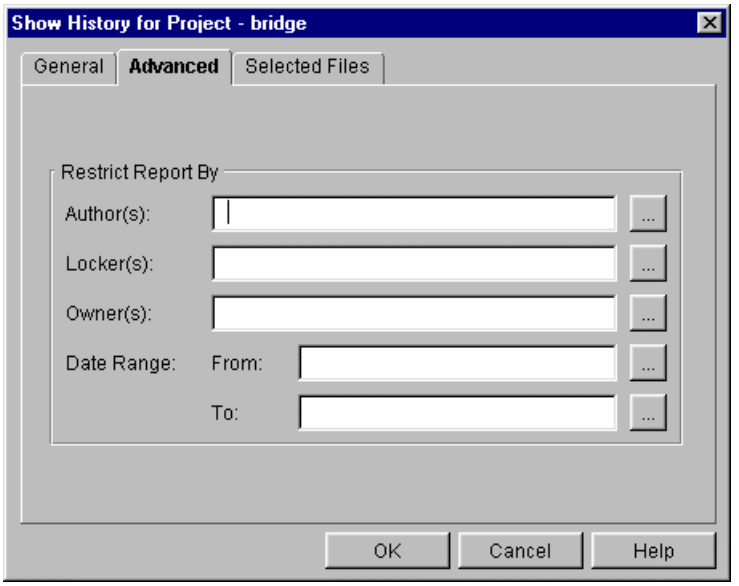
2 Select Actions | Show History. The Show History dialog box appears with the General tab active.



3 To generate a full report (both archive and revision information) for all revisions, click OK. Otherwise, you can limit the contents of the report by doing any of the following on the General tab:

To limit the report to. . .	Do this. . .
A specific revision instead of all revisions	Click the Browse button beside the Revision field; the Select Revision dialog box appears. Select the revision number, version label, or promotion group of the revision you want information about.
Archive information but not revision information	Select **File information only** from the Report Type drop-down list.
Revision information but not archive information	Select **Revision information only** from the Report Type drop-down list.
Currently locked revisions	Select **List locked revisions** from the Report Type drop-down list.
Revision numbers that correspond to a specified version label	Select **List revisions with version label** from the Report Type drop-down list. Then, click the Browse button beside the Label field and select the version label.
Revisions that are associated with a specified promotion group	Select **List revisions in group** from the Report Type drop-down list. Then, select the promotion group by clicking the Browse button beside the Group field and selecting the promotion group.
The newest revisions on the trunk	Select **List newest revisions** from the Report Type drop-down list.
Archives whose tip revisions are not the same as the specified revision number or version label	Select **Check tips against version/revision** from the Report Type drop-down list.

4 To further limit the report by date, author, owner, and/or user who has revisions locked (locker), click the Advanced tab. Otherwise, go to Step 6.



5 On the Advanced tab, do any of the following:

To limit the report to. . .	Do this. . .
Revisions authored by certain users	Enter the user IDs of the authors in the **Author(s)** field or click the Browse button to select the user IDs. Separate multiple IDs by commas.
Revisions locked by specific users	Enter the user IDs of the lockers in the **Locker(s)** field or click the Browse button to select the user IDs. Separate multiple IDs by commas.
Archives owned by certain users	Enter the user IDs of the owners in the **Owner(s)** field or click the Browse button to select the owners. Separate multiple IDs by commas.

To limit the report to. . .	Do this. . .
Revisions checked in within the specified range of dates	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. In 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.
Revisions checked in within the specified range of dates (cont.)	**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.

6 To review the selected versioned files before you generate the history report, click the Selected Files tab.

7 Click OK. The history report is generated and displayed.

Using the Command-Line Interface

To generate a history report using the command-line interface, use the VLOG command. The syntax for this command is:

`vlog [`*`option...`*`] [`*`file_name...`*`]`

where *file_name* specifies one or more archives or workfiles.

The following table lists frequently used options for this command.

To view. . .	Use this command line. . .
The archive name, user ID of the locker, locked revision number, and check-in revision number	`vlog -bl[`*`user_id`*`[,`*`user_id`*`]]`
A one-line report on newest revisions	`vlog -bn[`*`branch`*`]`
Revision information	vlog -br
Archive information	vlog -b
Archives owned by certain users	vlog -o*user_id*
Changes made by certain users	vlog -a*user_id*

See the *Serena PVCS Version Manager Command-Line Reference Guide* for complete information about the VLOG command.

Generating Security Reports

Security is implemented in Version Manager by defining an access control database. An access control database defines the users who are authorized to perform actions on projects and defines the actions that users can perform. For complete information about Version Manager security, see "Using Security" on page 203.

An access control database is an encrypted file; therefore, you must generate a security report to view its contents.

How to Read a Security Report

The following is an example of a security report generated in the desktop client. Reports generated in the command-line interface will be in a slightly different format.

```
PRIVILEGE Developers: \
        LOCKTIP \
        LOCKNONTIP \
        UNLOCK \
        BREAKLOCK \
        GETTIP \
        GETNONTIP \
        PUTTRUNK \
        PUTBRANCH \
        STARTBRANCH \
        CHANGEACCESSLIST \
        CHANGEOWNER \
        CHANGEPROTECTION \
        .
             .
             .
            MODIFYGROUP \
        DELETEGROUP
USER kasiaj/ (SUPERUSER)
USER Admin/ (SUPERUSER)
USER chrisp/ (UNLIMITED)
USER georges/ (UNLIMITED)
USER ken/$$ken (UNLIMITED)
USER edru/ (UNLIMITED)
USER betsyf/ (UNLIMITED)
USER krisb/ (UNLIMITED)
USER briang/ (UNLIMITED)
USER bruceb/ (UNLIMITED)
USER kerstinb/ (UNLIMITED)
USER brads/ (UNLIMITED)
GROUP Dev (Developers): chrisp edru briang \
    betsyf kasiaj shawnl kerstinb
```

In a security report:

- A backslash (\) is used to continue a line.
- Each privilege set, user, and group starts a new line. There is only one per line.

- Each privilege set definition line contains the name of the privilege set followed by the base privileges that define the privilege set. For example:

```
PRIVILEGE Developers: \
LOCKTIP \
LOCKNONTIP \
.
     .
     .
```

  In this example the privilege set name is Developers.

- Each user definition line contains the user ID/password and the given privileges or privilege set in parentheses. A password does not have to be specified. For example:

```
USER kasiaj/ (SUPERUSER)
USER ken/$$ken (UNLIMITED)
```

  The user, kasiaj, has no password specified. The user, ken, has the password $$ken specified.

NOTE To display passwords, you must have the SuperUser privilege.

- Each access list group definition line contains the name of the group, the given privileges or privilege set in parentheses, and the users assigned to the group. For example:

```
GROUP Dev (Developers): chrisp edru briang \
     betsyf kasiaj shawnl kerstinb
```

  In this example, the group name is Dev and has the Developers privilege set assigned to it.

- Comment lines are preceded by either a pound sign (#) or an exclamation point (!).

Using the Desktop Client

You must have the View Access Control Database (ViewAccessDB) privilege to generate a security report.

To generate a security report:

1. Select the project database or project associated with the access control database that you want to view.
2. Select Admin | Security | Show Report. The security report is generated and displayed.

Using the Command-Line Interface

You must have the View Access Control Database (ViewAccessDB) privilege to generate a security report.

To generate a security report using the command-line interface, use the READDB command.

NOTE readdb is located in the admin subdirectory of the bin directory.

The syntax for this command is:

```
readdb [option...]
```

Options

-a The most frequently used option for this command is -a:

```
readdb -adatabase_name
```

The -a option specifies the name of the access control database to read. If you don't use this option, the command uses the value embedded in the Version Manager program files by the VCONFIG -a command or the name specified in the configuration file by the AccessDB directive.

-p To display the password of the current user, use the -p option. If you have the SuperUser privilege, all passwords in the database are displayed.

See the *Serena PVCS Version Manager Command-Line Reference Guide* for complete information about the READDB command.

Viewing Changes to Projects (change.log)

Changes to the contents of a project, such as adding or removing subprojects or versioned files, are recorded in a change.log file located in the root directory of the project database. These text-based files include the following tab-separated data:

- Date of the change
- Time of the change
- Time zone in which the change was made (based on the location of the project)
- ID of the user who made the change
- The nature of the change
- The entity path of the affected object (enclosed in quotes)
- The archive path if the object is a versioned file (enclosed in quotes)

To view a change.log file, open it in a text editor.

Chapter 11

Integrating with Serena Collage

About the Version Manager Integration with Serena Collage

The Serena PVCS Version Manager integration to Collage enables web application development teams to automate all management and deployment of files. Collage provides powerful tools for scheduling the deployment of assets to all of your web servers, enabling you to fully automate your website testing and publishing cycles. By combining the deployment management of Collage with Version Manager, you no longer need to worry about manually extracting your web application files and transferring them to your web servers; Version Manager and Collage will do all of this for you.

Ways to Use the Integration

You can use the Version Manager Integration with Collage to:

- Automate transfer of web application files to Collage, which can then deploy the files to the web server(s)
- Once you decide which files you want to transfer, schedule regular transfers to automatically refresh your web application files

How the Integration Works

Stage	Description
1	The build engineer or project leader sets up the integration.
2	When it's time to deploy the files, the build engineer or project leader initiates the transfer of files from Version Manager to Collage.
3	Files are transferred to Collage if they match the promotion group or label and the file type specified in the setup file.
4	In Collage, the files are copied to the target deploy folder. The asset properties show details about the transfer. In Version Manager, the files are labelled to indicate the transfer. A log file is also created that tracks the transferred files.

Requirements

To enable the integration with Collage, you need the following:

- Version Manager 6.8.1 or higher
- Collage 3.0 or higher
- A Version Manager project database that contains the files to be transferred to Collage
- A Collage project to which to transfer the Version Manager files

Overview of the Integration Setup

After you've installed Version Manager and Collage and have identified the Version Manager files you want to transfer, you are ready to configure the integration. You must configure the integration on the system hosting Version Manager.

To set up an integration between Version Manager files and a Collage project, you must do the following:

- Set up Version Manager and Collage
- Create a setup file
- Set Collage and Version Manager login credentials

Setting up Version Manager and Collage

To set up Version Manager:

1. Identify the group of files you wish to transfer to Collage.
2. Do one of the following:
   - Assign a version label to the files.
   - Assign a promotion group to the files or promote the files to the same promotion group.

See the *Serena PVCS Version Manager User's Guide* for more information on version labels and promotion groups.

To set up Collage:

1. If you are running multiple Collage servers, determine which server you will use to transfer the files. You will need the host name or IP address and, if necessary, port number of this server.
2. If you have not already done so, create the project in Collage to which you will transfer files.
3. Optionally, create the specified deploy folder in Collage to which you will transfer the files. If you do not create the deploy folder ahead of time, it will be created during the file transfer.
4. Make sure that all of the file types which you will transfer from Version Manager have corresponding MIME types and asset types in Collage. If necessary, create MIME types and asset types for any missing file types.

   For information on displaying and creating MIME types and asset types, see the *Serena Collage Project Manager's Guide*.
5. Create the following project metadata fields. Define each as a **String** type field. You can accept the default **Version Handling** option for each field.

NOTE You must create the metadata fields exactly as they appear here, including spaces and capitalization.

Metadata Field	Description
VM Project Database	The name of the source Version Manager project database
VM Project Name	The name of the source Version Manager project
VM Revision	The Version Manager revision number; Collage maintains its own revision numbers, as well
VM Check in Description	The revision description associated with the transferred revision

IMPORTANT! If you run the integration before creating these metadata fields, the integration will create them for you. However, the fields are not automatically associated with the correct asset types, and no values are written to them.

For information on creating metadata fields, see the *Serena Collage Project Manager's Guide*.

6 Associate each of the new metadata fields with each of the asset types that correspond to the types of files you will transfer from Version Manager. For example, if you will transfer **.BMP** files, associate the metadata fields with the **BMP Image** asset type in Collage. At this point, you can also choose where on the Properties dialog box to display the metadata fields, either on the Properties | General view, or the Properties | Metadata view.

NOTE To make sure that users cannot modify the values of these metadata fields, choose **No** for the **Edit by User** option. This ensures that the fields are only modified when you run the integration.

See the *Serena Collage Project Manager's Guide* for more information.

Creating a Setup File

The setup file is a simple XML file in which you specify the information needed to complete the integration. It must be of the following form:

```
<?xml version="1.0" encoding="UTF-8"?>
<VersionManagerXML>
   <ProjectInfo VMPDBRoot="VM Project Database">
      <VMProject>VM Subproject</VMProject>
      <VMRecursive>true or false</VMRecursive>
      <VMRevision>Promotion Group or Label</VMRevision>
      <VMFileType>
         <type>*.file extension</type>
      </VMFileType>
      <VMAppliedLabel>Label after transfer</VMAppliedLabel>
      <CollageProject>Collage Project Name</CollageProject>
      <CollageHost>Collage Host Name or IP:port number
      </CollageHost>
      <CollageLocation>Collage Folder</CollageLocation>
      <Logfile>Log File</Logfile>
   </ProjectInfo>
</VersionManagerXML>
```

Several setup template files are included with your Version Manager installation. Starting with one of the template files, use your text editor or XML editor to define the elements of the setup file. You can create as many setup files as you need.

To create a setup file:

1 Open one of the template files in your text editor or XML editor.

- On UNIX, the files are on *VM_Install_dir*/vm/cm/samples
- In Windows, the files are on *VM_Install_dir*\vm\cm\Samples

2 Define the elements of the setup file as described in the following table:

This element...	Is...
VMPDBRoot	Required The Version Manager project database.
VMProject	Optional The path and name of the Version Manager subproject. If not specified, the transfer starts from the root of the project database. For example: `/bridge/hlp`
VMRecursive	Required Specifies whether to transfer files in subprojects. Values are `true` or `false`.
VMRevision	Required Specifies that files with this version label or promotion group should be transferred. The version label or promotion group name is case sensitive.

This element...	Is...
VMFileType	Required The Version Manager file type to transfer. You can specify multiple types. To specify all file types, enter `*.*`
VMAppliedLabel	Optional The label to be applied to the files in Version Manager after the transfer. If not specified, then the default is `Collage<Collage Folder><Collage Revision>`
CollageProject	Required The target Collage project name.
CollageHost	Required The target Collage machine name. This can be the host name or the IP address and port number, if necessary. For example: `CMServer:81`
CollageLocation	Required The target Collage deploy folder, including path, to which files will be transferred. If this folder doesn't exist, it will be created during the transfer. Use `/` to start this path at the root.
LogFile	Optional The file to view what files are transferred from Version Manager and what project it came from. Go to View | Filter | Version Label and enter the label specified in the setup file to see what files were transferred.

NOTE The setup template file may contain these additional elements: CollageLogin, CollagePassword, VMLogin, and VMPassword. Delete these elements and then follow the procedure in "Setting Login Credentials" on page 316 to correctly define them by running in setup mode.

3 To define multiple transfers in the setup file:

   a Copy the sections from `<ProjectInfo VMPDBRoot>` down to `</ProjectInfo>`.

   b Paste the sections below the first transfer, after `</ProjectInfo>`.

   c Define the elements for the second transfer using the table above.

   d Repeat these steps to create as many transfers as necessary.

4 Save the file.

Setting Login Credentials

After creating the setup file, you must run the transfer command in setup mode to capture your login credentials for the Version Manager project database and the Collage project. Both sets of user IDs and encrypted passwords will be saved in the setup file.

You need to re-run the transfer command in setup mode if you:

- Change which Version Manager project database or Collage project is specified in the setup file
- Define additional transfers in the setup file

To set your login credentials:

1 From the command line, run the following command:

`runvmcm -s setup.xml`

where *setup.xml* is the path and name of your setup file.

2 When prompted, enter your user ID and password for the Collage project.

3 When prompted, enter your user ID and password for the Version Manager project database.

NOTE If your Version Manager login source is set to Host ID, then you will not be prompted for your Version Manager user ID and password.

4 Steps 2 and 3 will repeat for each transfer specified in the setup file.

Transferring Files to Collage

Once you have created the setup file and have run it in setup mode, you can initiate the transfer of files to Collage in three ways:

- **Manually:** From the command line, issue the runvmcm command.
- **Automatically:** You can schedule the transfer as an O/S scheduled task or as a Collage scheduled server command.
- **From the Version Manager desktop client:** From Version Manager, create a toolbar button to initiate the transfer.

Manually Transfer Files

Do one of the following:

- From the command line, run the following command:

  `runvmcm setup.xml`

  where *setup.xml* is the path and name of your setup file.

- To initiate transfers defined in multiple setup files, run:

  `runvmcm setup1.xml setup2.xml setup3.xml`

  where *setup1.xml*, *setup2.xml*, and *setup3.xml* are the paths and names of your setup files.

Automatically Transfer Files

You can set up the transfer to run as an O/S scheduled task or as a Collage scheduled server command. Use the same parameters as shown for the command line. See your O/S documentation or Collage documentation for information on setting up a scheduled task or server command.

Running Collage as a Windows Service

Keep the following important points in mind if Collage is running as a Windows Service:

- Make sure that the user account used to run the Windows Service has sufficient access to the Version Manager project database files. For example, if the project database was originally created with a UNC network path, the user account which runs the Collage service must be a network Windows domain account. A local system account cannot resolve UNC paths.
- You cannot transfer files from project databases that were created using a mapped drive letter. If Collage is running as a service, the Version Manager project database must have originally been created using UNC or another network protocol.

Transfer Files from the Version Manager Desktop Client

You can create a toolbar button in Version Manager that initiates the transfer of files by specifying the transfer command and the setup file.

To set up the transfer from Version Manager:

1. In Version Manager, select Admin | Tool Configuration and then the Add button.
2. Enter a name for the transfer in the **Name** field.
3. In the **Command** field, enter `runvmcm`. For example:

   Command: `runvmcm`

   Arguments: `"D:\Program Files\Serena\vm\CMSamples\vmcmsetup.xml"`
4. In the **Arguments** field, enter the path and name of the setup file. Click OK.
5. To initiate the transfer, click the toolbar button.

Tracking Transferred Files

Once you've transferred a file from Version Manager to Collage, you can view file properties related to the transfer from either program or from the log file.

Program or file	To view transfer information:	These properties appear:
Version Manager	**1** Select View | Filter | Version Label. **2** Enter the applied label name (defined in the setup file as `<VMAppliedLabel>`). The default label is `Collage<Collage Folder><Collage Revision>`.	If default label, then: ■ Collage location to which the file was copied ■ Version of the Collage asset
Collage	**1** Go to the Content | Deploy Folders view. **2** Display the Properties dialog box for the file. **3** Depending on how you set up the metadata fields which display the Version Manager information, the properties appear in one of the following views: • Properties | General • Properties | Metadata	■ Name of the Version Manager project database from which the files were transferred ■ Name of the Version Manager project from which the files were transferred ■ Version Manager revision number that was transferred ■ Revision description associated with the transferred revision
Log file	Open the log file in a text editor. The log file name and location are defined in the setup file as `<LogFile>`. **NOTE** You can reset the log file by running the following command: `runvmcm -r setup.xml`	■ Date and time of the transfer ■ Identity of the originating Version Manager file and revision ■ Collage location to which each file was copied ■ Version of the Collage asset

Chapter 12

Using the Version Manager Conversion Utility for SourceSafe

Introduction

Serena PVCS Version Manager and Microsoft Visual SourceSafe use different storage methods so before you can begin using Version Manager on your SourceSafe files, you must convert them to Version Manager archives. Serena provides a conversion utility to do this for you.

This chapter provides information about the vss2vm converter utility and how you use it to move your files from Visual SourceSafe to Version Manager.

IMPORTANT! Before using the converter, ensure that all the files you want to convert are checked in to Visual SourceSafe.

The converter:

- Retains file histories and comments on each SourceSafe file you choose to convert.
- Creates Version Manager projects that correspond to the SourceSafe projects being converted.
- Retains user-defined labels and comments. See "User-Defined Labels" on page 322.
- Converts branches.
- Configures projects to use the shared files encountered during the conversion.

IMPORTANT! If you are using Version Manager with any network configuration other than the supported network configurations listed in the *Serena PVCS Version Manager Installation Guide*, the converter may not work properly. Before starting a conversion, see "Before You Begin" on page 325.

User-Defined Labels

SourceSafe assigns user-defined labels (known as version labels in Version Manager) differently than Version Manager. When you choose to assign a label to the tip (newest) version of a SourceSafe file, SourceSafe creates a new version (known as a revision in Version Manager) of the file for each assigned label.

For example, if you choose to assign a label to version 1 of a file (which is the tip), SourceSafe creates version 2 of the file and assigns the label to version 2. If you choose to assign two labels to version 1 of a file (which is the tip), SourceSafe creates version 2 and version 3 of the file and assigns one label to version 2 and one label to version 3.

Version Manager, on the other hand, does not create a new revision of a file when a label is assigned to a revision. For example, if you choose to assign a label to revision 1.1, Version Manager assigns the label to revision 1.1 and does not create a new revision. If you choose to assign two labels to revision 1.1, Version Manager does not create a new revision for each label; it assigns both labels to revision 1.1.

The converter uses the Version Manager labeling scheme when converting SourceSafe files. If the converter encounters a label on a SourceSafe version, it does not increment the Version Manager revision number. For example, if there is a label on versions 2, 3, and 4, the converter places all of the labels on one Version Manager revision.

SourceSafe allows users to enter comments when defining a label for a version of a file. The converter retains these comments. In Version Manager, the labels and their comments are part of the Version Manager change description, as follows:

```
This is the change description.
Label: label comment
Label2: label2 comment
```

About Version and Revision Numbers

SourceSafe creates a new version each time a file is checked in **AND** each time a label is applied, whereas Version Manager creates a new revision only when a file is checked in.

To enable easy correlation between the original SourceSafe versions and the resulting Version Manager revisions, the Version Manager revision number is incremented each time the SourceSafe version number is incremented. Since Version Manager does not create revisions for version labels, there will be a gap (skipped revision number) wherever there was a user-defined label in SourceSafe. Also, any versions deleted from SourceSafe will result in a skipped revision number in Version Manager. See the example in the following table.

SourceSafe Versions	VM Revisions	VM Revisions with -vr Option
7 (Label-3)	-	-
6 (Revision)	1.5 (Label-3)	1.6 (Label-3)
5 (Revision)	1.4	1.5
4 (Deleted Revision)	-	-
3 (Label-2)	-	-
2 (Label-1)	-	-
1 (Revision)	1.0 (Label-1; Label-2)	1.1 (Label-1; Label-2)

NOTE The `-vr` option causes the numbering of Version Manager revisions to start at 1.1 rather than at 1.0 so that the portion of the revision number to the right of the decimal point matches the SourceSafe version number. Thus SourceSafe version **1** equals Version Manager revision 1.**1** and SourceSafe version **5** equals Version Manager revision 1.**5**. See "Options" on page 335.

Limitations

The converter has the following limitations:

- Workspace settings are not retained during conversion.
- Labels on branches are not applied to branch revisions during conversion if the same label is already on the converted trunk. Instead, the branch label is placed in the Version Manager change description for the branch revisions. If the label is not found on the trunk revision, it will be applied to the branch.
- Locks on Visual SourceSafe files are not preserved during the conversion.

- Many development environments, such as Visual C++, are limited in that they can use only the last source code control interface installed (such as visual SourceSafe or the Version Manager SCC Interface). Therefore, you must plan so that all projects that a user works on can be converted at the same time. As an example, assume a user works on two projects in Visual C++, both using the Visual SourceSafe Source Code Control interface. After the Version Manager Source Code Control interface is installed, the user will no longer be able to access the SourceSafe projects corresponding to the Visual C++ projects.
- The archives will not be split into revision libraries and metadata. If you wish to split the archives for use with a Version Manager file server, you must do so as a separate step after the conversion. See "Creating Revision Libraries" on page 157.
- If a version comment includes a multi-byte character that spans the 256th character position (or a multiple of 256), the character may be corrupted. This corruption is passed on to Version Manager as output from the SourceSafe command line, and as such is beyond our ability to correct.

Where the Archives Are Located After Conversion

After the conversion of the SourceSafe files, the Version Manager archives are located in a directory beneath the Version Manager project database that you specify when you convert the files. For example, `d:\test\archives`, where `d:\test` is the location of the Version Manager project database.

If you are converting an entire SourceSafe database, the converter creates a subdirectory for each SourceSafe project that exists in the database and places the converted archives into the appropriate directories. For example, if your SourceSafe project structure looks like:

```
$\
├─src
├─ss_new
├─test
```

The converter creates three subdirectories in the directory named archives beneath the Version Manager project database location: `archives\src`, `archives\ss_new`, and `archives\test`.

The converter also creates three Version Manager projects in the project database with the same names.

If you are converting a project, the converter creates a single subdirectory beneath the directory named archives for that project and places the converted archives in that subdirectory. For example, `archives\project`. The converter also creates a Version Manager project in the project database with the same name as the SourceSafe project name.

When converting a single project that contains archives shared with other projects, these archives will be marked as shared when they are encountered by the archive converter, but not actually converted during the project conversion.

When a shared archive is encountered during the process of importing the converted archives, the path where the shared archive originated is checked to see if an archive with that name exists. If it exists, the archive is imported into the project.

NOTE This path is printed in the Shared.log file and is also printed in the .txt file that is generated to indicate a shared file.

If the archive is not found, the archive converter is called to convert the parent archive of the share, unless it has been deleted. In that case, the shared archive is used.

If you are converting a single file, the converter places the converted archive in the directory named archives; no directory beneath the archives directory is created for the single file.

NOTE In the process of converting SourceSafe files the converter creates a directory named convArch beneath the directory from which you run the converter. This directory temporarily contains the Version Manager archives. The converter deletes all but the log files from this directory after the conversion.

Before You Begin

Before you run the vss2vm converter you should assess the effort necessary to convert your Visual SourceSafe projects to Version Manager and verify that your system setups are compatible with the conversion process. The following sections help you with these tasks.

Assessing the Effort of VSS2VM Conversions

The guidelines below help you:

- Predict the effort necessary to convert Visual SourceSafe projects to Version Manager
- Identify risk factors that increase the difficulty of a conversion

NOTE Serena makes no claims about the duration of the conversion task.

The time needed to complete a conversion depends on many factors, including:

- The complexity of the SourceSafe data (note the risk assessment values: low, medium, high, and extreme).
- The size of the VSS Database
- The number of projects
- The number of archives
- The speed of the hardware and the network (if applicable) where the conversion is run
- The user's experience with the tools and data

If you have followed the *Microsoft Visual SourceSafe Best Practices* guidelines, your chances of a relatively quick and successful conversion to Version Manager are greatly increased. See the following URL for more information:
`http://msdn.microsoft.com/ssafe/technical/articles.asp`

User's experience level with the tools and the data

If you have significant experience with Visual SourceSafe, Version Manager, command line, script writing, and the SourceSafe data itself, your risk of problems is reduced. Naturally the converse is true; lack of experience in one or more of these areas will increase your risk.

A lack of experience with Visual SourceSafe or Version Manager, or unfamiliarity with the existing project, will increase the difficulty of deciding how to best deal with any problem files that should to be handled manually. Additionally, command-line experience and scripting are important skill sets for handling problem files. If either of these key skills are missing from the skill set of the person assigned to complete the conversion, consulting services should be considered.

Important experience areas include:

- User experience/skills with Version Manager
- User experience/skills with Visual SourceSafe (VSS)
- User experience/skills with DOS/Command Line
- User skills with creating scripts
- User familiarity with VSS data

Risk Factors

The following sections summarize the risk factors. See Table 12-1, "Risk Assessment Worksheet," on page 328 to learn how these risk factors impact the potential conversion time.

Low Risk

- Minimal branching - This means that you do not have more than 2 levels of branches
- No deleted branches
- No rollbacks
- No pinned files
- No archived Versions
- No shares
- No Special Characters in project or file names (parentheses, $, &, etc.)
- User is experienced with Version Manager and Visual SourceSafe
- The *Microsoft Visual SourceSafe Best Practices* have been followed
- Have not "Stored only latest version"
- Visual SourceSafe repository is Less than 500MB

Medium Risk

- Moderate Branching - This means that you do not have more than 4 levels of branches
- Some deleted branches
- Some deleted shares
- No rollbacks
- No pinned files
- No archived Versions on branches
- No shares across top level projects
- No Special Characters in project or file names (parentheses, $, &, etc.)
- User is experienced with Version Manager and Visual SourceSafe
- The *Microsoft Visual SourceSafe Best Practices* have been followed
- Have not "Stored only latest version"
- Visual SourceSafe repository is 500MB to 1GB

High Risk

- Major Branching - This means that you have more than 4 levels of branches
- Some deleted branches
- No rollbacks
- Pinned files
- No shares across top level projects
- Deleted shares
- Archived Versions
- Users have included snippets of code in change descriptions/comments
- Special Characters in project or file names (parentheses, $, &, etc.)
- User is experienced with Version Manager or Visual SourceSafe but not both.
- The *Microsoft Visual SourceSafe Best Practices* have not been followed
- Have not "Stored only latest version"
- Visual SourceSafe repository is 1GB to 3GB

Extreme Risk

- Major Branching - This means that you have more than 4 levels of branches
- Multiple deleted branches in one or more VSS archives
- Deleted shares
- Rollbacks
- Pinned files
- Archived Versions

- Users have included snippets of code in change descriptions and comments
- Special Characters in project or file names (parentheses, $, &, etc.)
- Shares across top level projects.
- User has little or no experienced with Version Manager and Visual SourceSafe.
- The *Microsoft Visual SourceSafe Best Practices* guidelines have been ignored
- Stored only latest version (see Note below)
- Visual SourceSafe repository is Greater than 3GB

NOTE Use of the "Store only latest version" feature removes previous copies of the workfile, but leaves history. Performing a get of an object utilizing this feature will fail, as it is only possible to retrieve the latest revision of the file. In cases where this feature is enabled, get the latest revision and use the Add Workfile option in Version Manager to create a new archive.

Risk Assessment Worksheet

The following worksheet is designed to help determine the risk level of a project, as a predictor of a successful (and timely) completion of the Visual SourceSafe to Version Manager conversion.

Under the heading of VSS Risk Factors are SourceSafe *capabilities* that, if used, increase the risk of your conversion to Version Manager having problems. The more these features have been used, the more difficult the conversion is likely to become.

Use the risk factor guidelines above to fill in the worksheet below.

NOTE Boxes that are grayed out are not to be used. For example, if you have pinned files, the first option is a High effort; the more pinned files, the greater the effort.

Table 12-1. Risk Assessment Worksheet

VSS Risk Factor	Low	Medium	High	Extreme
Branching				
Deleted shares				
Shared Archives				
Size of Visual SourceSafe Repository				
User's Experience with Version Manager (choose Low for the most experience; Extreme for the least experience)				
Deleted Branches				
Special Characters in Project or File Names				
Archived Versions				
Pinned Files				
Snippets of Code in Change Descriptions/ Comments				

VSS Risk Factor	Low	Medium	High	Extreme
Shares Across Top-Level Projects				
Stored only latest version				
Rollbacks				

Interpreting the Assessment Worksheet

To determine your effort level, look at the table entries you have filled in. If you have any entries in the Extreme column, you may assume that it will be an extreme effort.

The fewer check marks in the right-hand columns, the easier the task should be. More than three check marks in the High and Extreme columns suggests that you may have a complicated VSS project database that could require on-site consulting services to assist you with the conversion.

NOTE The following are rough estimates of the possible duration of the conversion task. Every conversion is different; even though you have rated your conversion as *low,* it may take more time than the below estimates.

Risk Assessment	Estimated Conversion Times
Low	1 Day or less
Medium	1 to 5 Days
High	1 to 3 Weeks
Extreme	3 to 6 Weeks

Verifying System Setups and Settings

Before running the converter utility, you must complete the recommended steps with respect to your system setups, Visual SourceSafe settings, and Version Manager settings.

Prerequisite System Setups

1. Visual SourceSafe 5 or 6 and Version Manager 6.5 or later must be installed on your workstation or accessible network server.
2. The command-line executable ss.exe must be installed on your workstation or accessible network server.
3. The Visual SourceSafe and the Version Manager executables must be in your path statement.

   The Visual SourceSafe path must be to the directory containing the ss.exe file and the Version manager directory is the *<Install Directory>*\vm\Win32\bin directory containing the pvcsvmnt.exe file.

   To test this step for Version Manager, at the command-line prompt, enter vcs -u. You should see the Version Manager version/copyright banner.

   To test this step for Visual SourceSafe, at the command-line prompt, enter ss. You should see a version/copyright banner.

4 You must set the SSDIR environment variable to point to the directory where the SRCSAFE.INI file for the database to be converted is located. To find this directory, in SourceSafe select File | Open Database. The path to the directory is displayed to the right of the database name.

To set the SSDIR environment variable, at the command-line prompt, enter:

```
set SSDIR=\\<server_name>\share\vss
```

To test this step, at the command-line prompt, enter:

```
ss dir -r
```

You should see the contents of your Visual SourceSafe database. If this is a large database, do not use the **-r** option.

Prerequisites in Visual SourceSafe

1 You must have a user ID in the SourceSafe database that matches the network ID of the user who is logged in when the converter is started. This user must have FULL permissions to the SourceSafe database. Users are viewed and added using the Visual SourceSafe Administrator.

To test this step, at the command-line prompt, enter:

```
ss whoami
```

2 In the Visual SourceSafe Administrator, configure Visual SourceSafe to "Use network name for automatic user login" (Tools | Options | General tab).

3 If you have moved your SourceSafe data, set the Data_Path variable in the SRCSAFE.INI file to point to the SourceSafe database that you want to convert so that SourceSafe accesses the correct database.

4 In Visual SourceSafe, run Analyze DB on the source database. If there are errors, manually fix them or run the Analyze and Fix DB functions in Visual SourceSafe.

5 Configure SourceSafe to preserve the modification date on check out so that Version Manager checks in the revisions with the original modification date. If this is not done, the modification date of the Version Manager archives will have the *date of check in* rather than the true modification date. To preserve the modification date, in the Visual SourceSafe Explorer select Tools | Options | Local Files tab. Then, select Modification from the drop-down list labeled **Set date/time on local files**.

6 In Tools | Options, uncheck the options for **Assume working folder based on current project** and **Assume project based on working folder**. If these are not checked, no action is necessary.

7 Exit the SourceSafe GUI before running the converter.

Prerequisites in Version Manager

1 You must configure any desired configuration settings in the Version Manager master configuration file. The master configuration file must not have a promotion model defined. If there is one defined, disable it during the conversion.

2 If the master configuration file is embedded, un-embed the master configuration file before converting. Refer to "Embedding a Master Configuration File" on page 59.

NOTE If a different archive suffix is desired, first convert the archives into Version Manager format and then change the archive suffix of the converted files. To change the archive suffix, run the Convertarch utility against the converted archives. The Convertarch utility is available from Serena support. You should also change the default archive suffix for the Version Manager project database so that archives you create in the future will have the desired archive suffix.

3 Keyword expansion should be turned off during conversion, which prevents unintentional expansion of keywords if they happen to exist in the original SourceSafe versions. You can turn this option back on after conversion is complete. To turn this setting back on, use Admin | Archive Attributes from the Version Manager desktop client or enter:

```
vcs +pk foo.c-arc
```

from the command line interface, where *foo.c-arc* is the name of one or more archive files. Refer to "Keyword Expansion Options" on page 84.

4 It is strongly recommended that you disable the Version Manager access control database during the conversion. If you disable the access control database, you must perform the next step (to add the Source Safe user IDs to the Version Manager access control database). For more information see "Disabling Security" on page 236.

5 If you have disabled the Version Manager access control database, you must add all user IDs that are in the SourceSafe database to the Version Manager access control database. Otherwise this step is recommended, but not necessary. For more information see "About Access Control Databases" on page 204.

6 You must create a Version Manager project database in which to place the Version Manager projects that are created from the SourceSafe projects. Refer to "Creating a Project Database" on page 22.

7 Define the environment variable, ISLVINI, to point to the directory that contains your ISLV.INI file. By default, this is C:\WINDOWS or C:\WINNT.

Converting IDE Projects

If you are converting any files that were placed under source control from within an Integrated Development Environment (IDE) that complies with Microsoft's Source Code Control Interface (SCC) standard, you need to follow the procedures outlined in this section for your particular IDE.

Converting Visual Studio .NET Projects

If you are converting files that were added to Visual SourceSafe through Visual Studio .NET, you must complete the following pre and post conversion procedures.

Pre Conversion Procedure for Visual Studio .NET Projects

Preparing .NET projects for conversion:

1 Get the latest revision of the entire Visual Studio .NET solution from Visual SourceSafe.

NOTE Visual SourceSafe must still be your registered SCC provider. The last provider installed will be in effect.

To change the SCC provider to Visual SourceSafe, edit the registry key:
`Hkey_Local_Machine | Software | SourceCodeControlProvider`
by setting the string value: `ProviderRegKey`
to: `Software\Microsoft\SourceSafe`
Then restart the IDE.

2 Disconnect the IDE project from Visual SourceSafe:

a Select File | Source Control | Change Source Control.

b Select the solutions and projects and click **Unbind**.

NOTE Depending on how the projects were added to source control, you may not be able to select every project for one unbind operation. Repeat the selection and unbind step until all projects are unbound.

c Select File | Save All.

d Close Visual Studio .NET.

3 Use Windows Explorer to browse to the location that contains the local copy of the IDE project (the files you got in Step 1), and delete all files with the following extensions from the directory structure:

- *.scc
- *.vspscc
- *.vssscc

NOTE If the solution contains web projects, you must locate the projects on your Web server and delete all *.scc and *.vspscc files.

4 Set Version Manager as your SCC provider by doing one of the following:

- If you have not yet installed Version Manager, install it. The SCC provider will be set to Version Manager during the installation.
- If Version Manager is already installed but the SCC registry key has been overwritten by another provider or manually edited, edit the registry key:
  `Hkey_Local_Machine | Software | SourceCodeControlProvider`
  by setting the string value: `ProviderRegKey`
  to: `SOFTWARE\Merant\Integrations\SCC`
  Then restart the IDE.

5 Add the solution to Version Manager source control (File | Source Control | Add Solution To Source Control). See the *Version Manager IDE Client Implementation Guide* for the complete add procedure.

IMPORTANT! If the solution contains Web projects, it is best to add each individual project to source control, rather than the entire solution.

Once the above steps are complete, see "Running the Converter" on page 335.

Post Conversion Procedure for Visual Studio .NET Projects

Once you have converted the Visual SourceSafe files to Version Manager archives, complete the steps below to complete the process.

Finalizing converted .NET projects:

1 Use Windows Explorer to browse to the location that contains the converted archives and delete all files with the following extensions from the directory structure:

- *.scc-arc
- *.vspscc-arc
- *.vssscc-arc
- *.sln-arc
- *.*project_type*-arc
  For example: *.csproj-arc or *.vbproj-arc

2 Use Windows Explorer to copy the converted archives over the original archives. You may have to do this in smaller chunks if the directory structure is not the same.

3 Test the converted projects. You should now have access to any previous revisions and history through Visual Studio .NET.

4 To connect additional workstations to the converted solution, go to each workstation and select File | Source Control | Open From Source Control and complete the dialog and prompts as needed. See the *Version Manager IDE Client Implementation Guide* for the complete procedure.

IMPORTANT! If the individual projects were added to source control rather than the entire solution, do the following:

1 Create a new solution.

2 Select File | Source Control | Add Project From Source Control.

3 Select a project to add and complete the dialog and prompts as needed. See the *Version Manager IDE Client Implementation Guide* for the complete add procedure.

4 Repeat steps 2 and 3 for each project.

5 Select File | Save All.

Converting Other SCC-Based IDE Projects

If you are converting files that were added to Visual SourceSafe through an IDE **other than** Visual Studio .NET, you must complete the following pre and post conversion steps.

IMPORTANT! To convert Visual Studio .NET projects, see "Converting Visual Studio .NET Projects" on page 331.

Pre Conversion Procedure for Other SCC-Based IDE Projects

Preparing IDE projects for conversion:

1 Get the latest revision of the entire IDE project from Visual SourceSafe.

NOTE Visual SourceSafe must still be your registered SCC provider. The last provider installed will be in effect.

To change the SCC provider to Visual SourceSafe, edit the registry key:
`Hkey_Local_Machine | Software | SourceCodeControlProvider`
by setting the string value: `ProviderRegKey`
to: `Software\Microsoft\SourceSafe`
Then restart the IDE.

2 Close your IDE.

3 Use Windows Explorer to browse to the location that contains the local copy of the IDE project (the files you got in Step 1), and delete all files with *.scc extensions from the directory structure.

Once the above steps are complete, see "Running the Converter" on page 335.

Post Conversion Procedure for Other SCC-Based IDE Projects

Once you have converted the Visual SourceSafe files to Version Manager archives, complete the steps below to complete the process.

NOTE See the *Version Manager IDE Client Implementation Guide,* as needed, for menu commands and procedures specific to your IDE.

Finalizing converted IDE projects:

1 Use Windows Explorer to browse to the location that contains the converted archives for the IDE project, and delete all files with *.scc extensions from the directory structure.

2 Set Version Manager as your SCC provider by doing one of the following:

- If you have not yet installed Version Manager, install it. The SCC provider will be set to Version Manager during the installation.
- If Version Manager is already installed but the SCC registry key has been overwritten by another provider or manually edited, edit the registry key:
`Hkey_Local_Machine | Software | SourceCodeControlProvider`
by setting the string value: `ProviderRegKey`

to: SOFTWARE\Merant\Integrations\SCC
Then restart the IDE.

3 Open the IDE project in the IDE.

4 Add the IDE project to source control. Since the archives already exist, you will be prompted whether you wish to check the workfiles into the existing archives as new revisions. Yes, this is what you want to do.

5 Test the converted projects. You should now have access to any previous revisions and history through your IDE.

6 Connect additional workstations to the converted project as needed.

Running the Converter

The converter can be run in either interactive mode or non-interactive mode. When you run the converter, use the -# option to create a verbose log file. The log file will help you determine the nature of any errors encountered during the conversion. If you do not use the -# option, you will receive only minimal logging information during the conversion.

IMPORTANT! Before running the converter, ensure that all files to be converted are checked in and all users are logged out of Visual SourceSafe.

To use the converter in interactive mode, at a command-line prompt, enter:

```
vss2vm
```

The converter will then prompt you for the information it needs to do a conversion. The purpose of non-interactive mode is so you can use the converter in a script.

Syntax

Syntax for non-interactive mode:

```
vss2vm -srSS_database [/SS_project]
[-saSS_file][-iduserid [:pwd]
-prVM_PDB_location [-ppVM_project]
[-#]
```

Syntax for interactive mode:

```
vss2vm [-iduserid] [:pwd][-#]
```

Options

IMPORTANT! You must use lowercase when entering command options.

-# This option turns on verbose mode, which prints the verbose output of the converter to the display and to a log file named ss2pvcs.log. The default is not to display output (non-verbose mode; do not create this log file). The output that is displayed is the same information that is written to the log file.

-h Displays help for the vss2vm command and its options.

-id `-id`*`userid`* `[:`*`pwd`*`]`

This option is required only if you specify a Version Manager project database or project that has an access control database enabled, and VLOGIN (Login Dialog) is the login source used to obtain user identification or any login source if the user ID defined in the access control database has a password associated with it. In these cases, you must specify a value for this option.

-pp `-pp`*`VM_project`*

This option specifies a Version Manager project within the Version Manager project database that will contain the converted SourceSafe projects and files. If you specify a Version Manager project, the SourceSafe project being converted is placed beneath the Version Manager project as a subproject. For example, if you are converting a SourceSafe project named checkers and you specify a Version Manager project named games, then the converted project checkers is a subproject of the project games (`/games/checkers`).

-pr `-pr`*`VM_PDB_location`*

You must specify the location of the Version Manager project database that will contain the converted SourceSafe projects and files. For example, `-prd:\SS_conversion`. This is required when using non-interactive mode.

-sa `-sa`*`SS_archive`*

You must specify either a SourceSafe database, a SourceSafe project, or a single SourceSafe file to convert. One of these options is required when using non-interactive mode.

To convert an entire SourceSafe database, specify `-sr$/` on the command line.

To convert a SourceSafe project and any projects beneath that project, specify `-sr$/`*`SS_project`* on the command line. For example, `-sr$/test`.

To convert a single SourceSafe file, specify the project or project database to which the file belongs and the file. For example, `-sa$/test/readme`.

-sr `-sr`*`SS_database`* `[/`*`SS_project`*`]`

-vr This option causes the numbering of Version Manager revisions to start at 1.1 rather than at 1.0 so that the portion of the revision number to the right of the decimal point matches the SourceSafe version number. Thus SourceSafe version **1** equals Version Manager revision 1.**1** and SourceSafe version **5** equals Version Manager revision 1.**5**. See "About Version and Revision Numbers" on page 323.

To run the converter in interactive mode:

NOTE Do not move the converter executable (vss2vm) or the converter will not run.

1 Perform the steps in "Before You Begin" on page 325.

2 Change (cd) to the directory where you want the converter to create the convArch directory (see "Where the Archives Are Located After Conversion" on page 324). If necessary, create the directory.

3 Run the converter from this directory by entering:

```
vss2vm [-iduserid [:pwd] [-#]
```

The converter displays the following prompt:

```
Enter the path to the Version Manager database where the converted
    archives will be placed:
```

4 Enter the path of the Version Manager project database. For example:

```
d:\test
```

5 The converter displays a second prompt:

```
Do you want to specify a Version Manager project where your archives
    will be stored? (y/n):
```

Enter "y" to be prompted for a Version Manager project name. Otherwise, enter "n".

6 If you entered y, the converter displays the following prompt:

```
Enter the Version Manager project where the converted archives will
    be placed:
```

Enter the name of the Version Manager project. For example:

```
Temp
```

The converter now displays the following prompt:

```
Choose an option:
   1.) Convert entire VSS database
   2.) Convert one VSS project or subproject
   3.) Convert one VSS file
```

Do one of the following:

- Enter 1 to convert the entire SourceSafe database. Valid input value is $/.

  Every file and project in the database is converted. The converter creates a subdirectory for each SourceSafe project that exists in the database and places the converted archives into the directories. (See "Where the Archives Are Located After Conversion" on page 324.)

- Enter 2 to convert one project or subproject in the SourceSafe database. You must provide a project or a subproject name. Valid input values are:

  $/*project*
  $/*project*/*subproject*

  The converter prompts you for the name of the project to convert; enter the project name.

- Enter 3 to convert one file in the SourceSafe database. You must provide a filename. Valid input values are:

  $/*project*/*filename*
  $/*project*/*subjproject*/*filename*

  The converter prompts you for the name of the file; enter the file name.

To run the converter in non-interactive mode:

NOTE Do not move the converter executable (vss2vm) or the converter will not run.

1. Perform the steps in "Before You Begin" on page 325.
2. Change (cd) to the directory where you want the converter to create the convArch directory (see "Where the Archives Are Located After Conversion" on page 324). If necessary, create the directory.
3. Run the converter from this directory by either:
   - Entering a non-interactive vss2vm command line (see Syntax for non-interactive mode on page 335).

     For example:

     ```
     vss2vm -sr$/test -prd:\sample
     ```

     This example converts a SourceSafe project named test and places it in the Version Manager project database on `d:\sample`.
   - Creating a script that contains a vss2vm command line. The following is an example of a Version Manager PCLI script named conversion.pcli:

     ```
     set -vSSprj $1
     set -vVMpdb $2
     run -e vss2vm -sr'$/'$SSprj -pr$VMpdb
     ```

Internationalized Strings

This utility has been designed to use internationalized strings if the strings are contained within the message catalog (pvcsvm.cat). In order for the converter to see the catalog file, you must set the environment variable NLSPATH to the location of the catalog file. By default, the catalog file is located in the bin directory of your Version Manager installation location (for example, `C:\Program Files\Serena\vm\common\bin\win32`).

To set the NLSPATH variable:

1. From the Windows Start menu, select Control Panel | System. The System Properties dialog box appears.
2. Select the Advanced tab.
3. Click the **Environment Variables** button. The Environment Variables dialog box appears.
4. Click the **New** button. The New Variable dialog box appears.
5. Enter NLSPATH in the **Variable name** field.
6. Enter the location of the pvcsvm.cat file in the **Variable value** field.
7. Click the **OK** button on each dialog box.

To verify that the string is in the message catalog, run `vss2vm -#` from the command prompt.

If the strings are displayed in English, your catalog file does not contain the appropriate strings or the `NLSPATH` is incorrect.

NOTE If you are converting localized Visual SourceSafe projects, please contact Serena support before you start the conversion. They will provide you with a message catalog file specific to the language you are using.

The Log Files

The converter creates four log files during the conversion of SourceSafe files.

By default, these files are located in the convArch directory and named: ss2pvcs.log, ConvErr.log, shared.log, and dgb_path.log.

The converter creates convArch directory beneath the directory from which you run the converter.

After the conversion is complete, the contents of the convArch directory are deleted, with the exception of .log and.cfg files.

If the conversion should fail or if it is interrupted before completion, the contents of convArch are not deleted. If you run the conversion again without deleting the convArch directory, another directory, named convArch0 is created and the conversion is done in this directory. Each time you run the conversion without deleting the existing convArch directories, a new convArch directory is created (convArch1, convArch2, etc.).

ss2pvcs.log

The ss2pvcs.log file contains information such as:

- The names of the SourceSafe files that were converted.
- The location of the Version Manager archive to which each file was converted.
- The actions the converter took during conversions such as searching for archives, importing archives, creating projects, etc.

convERR.log

The convERR.log file is generated by the part of the program that imports the archives into the Version Manager project database. Normally, this file will empty, but if the converter is running in verbose mode, some information may be placed in it. The information in this file is provided mainly for informational purposes, with no wording to indicate errors. For example, this log file may contain path and file names.

Shared.log

The shared.log file contains references to shared files and where they are shared from:

"*path**filename*" shared from "parentpath\filename"

A text file is also generated to indicate a shared file. Refer to "Where the Archives Are Located After Conversion" on page 324.

dgb_path.log

This file is used by Serena support.

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 342, 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 342.	254 (full path including the file name)
pvcs_bindir	As listed in "Prohibited Characters for Files and Directories" on page 342.	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 342, 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

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W