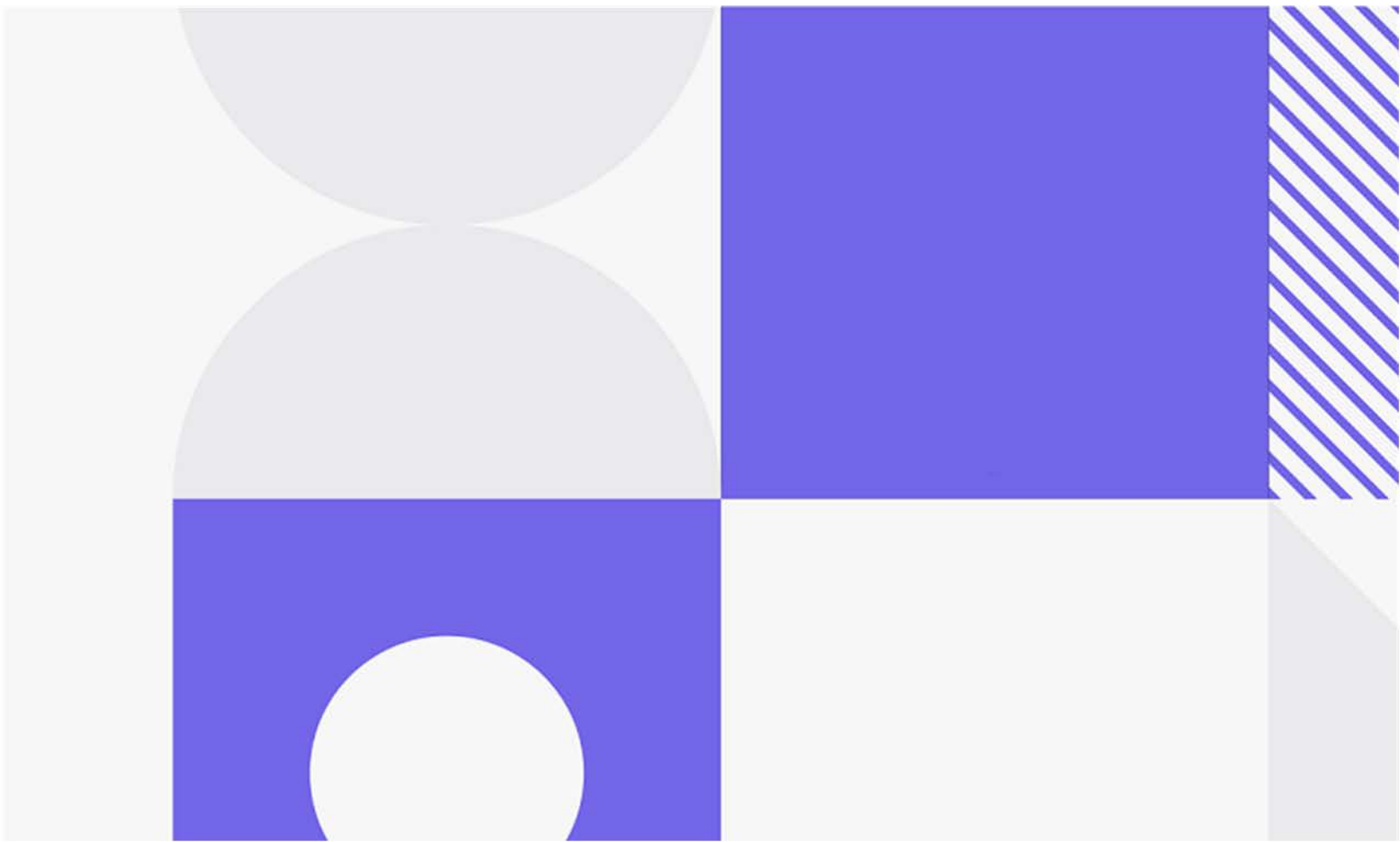




# PVCS Version Manager

Software version: 25.4

## IDE Client Implementation Guide



Copyright © 2025 Open Text.

The only warranties for products and services of Open Text and its affiliates and licensors ("Open Text") are as may be set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Open Text shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Product version: 25.4

Last updated: December 8, 2025

The most recent edition of this manual (with errata included) can be downloaded here:  
<https://www.microfocus.com/documentation/pvcs-version-manager/25.4/DEVINTUG.pdf>

# Table of Contents

---

<b>Part 1</b>	<b>The Version Manager IDE Client . . . . .</b>	<b>9</b>
	Introduction . . . . .	10
<i>Chapter 1</i>	<b>Overview of Version Manager Source Control. . . . .</b>	<b>11</b>
	Introduction . . . . .	12
	Source Control Concepts. . . . .	12
	Project Databases . . . . .	12
	Projects and Subprojects . . . . .	12
	Archives. . . . .	12
	Revisions . . . . .	13
	Workfiles . . . . .	13
	Locks . . . . .	13
	Version Labels . . . . .	13
	Promotion . . . . .	13
	Branches . . . . .	14
	Sharing . . . . .	14
	Workspaces . . . . .	14
	Available Source Control Information . . . . .	14
	Viewing Properties of Files Under Source Control. . . . .	14
	Monitoring Source Control Activity. . . . .	14
	Viewing Historical Archive or Revision Activity. . . . .	14
	Comparing Files or Revisions . . . . .	15
	How Version Manager Integrates with IDEs. . . . .	15
<i>Chapter 2</i>	<b>Setting Up Source Control with SCC IDEs . . . . .</b>	<b>17</b>
	Introduction . . . . .	18
	Recommended Workflow . . . . .	18
	Administrators . . . . .	18
	All Users . . . . .	18
	About Selecting a Source Control Provider . . . . .	19
	Selecting an SCC Provider . . . . .	19
	Testing an SCC Provider. . . . .	19
	Stopping the PVCSCCLIServ Service . . . . .	20
	Creating and Configuring Project Databases . . . . .	20
	About Version Manager Workspaces . . . . .	21
	Launching the Version Manager Desktop Client. . . . .	21
	About Setting Defaults for Version Manager Options . . . . .	22
	Setting Defaults . . . . .	22
	About Creating Source Control Projects . . . . .	23
	About Adding Files to Source Control . . . . .	24
	Returning Files to Source Control . . . . .	24
	Advanced Add Options . . . . .	24

	About Sharing Files Across Projects . . . . .	25
	Sharing Files. . . . .	25
	About Removing Files from Source Control . . . . .	26
<b>Chapter 3</b>	<b>Using Source Control . . . . .</b>	<b>27</b>
	Introduction . . . . .	28
	Logging In to Version Manager Projects . . . . .	28
	About Getting Files. . . . .	28
	Advanced Get Options . . . . .	29
	Getting IDE Projects from Source Control. . . . .	30
	About Checking Out Files . . . . .	30
	Advanced Checkout Options . . . . .	31
	About Undoing Checkout. . . . .	32
	Advanced Undo Checkout Options . . . . .	32
	About Checking In Files . . . . .	33
	Advanced Check-In Options . . . . .	33
	About Version Labels . . . . .	34
	Assigning Version Labels . . . . .	34
	Renaming Version Labels . . . . .	36
	Deleting Version Labels . . . . .	37
	About Promotion Groups. . . . .	38
	Checking Out Revisions Assigned to a Promotion Group. . . . .	38
	Assigning a Promotion Group to Revisions . . . . .	39
	Promoting Revisions to the Next Promotion Group. . . . .	40
	Changing a Promotion Group . . . . .	41
	Removing a Promotion Group . . . . .	42
<b>Chapter 4</b>	<b>Accessing Source Control Information . . . . .</b>	<b>45</b>
	Introduction . . . . .	46
	About Properties . . . . .	46
	Reviewing Properties . . . . .	46
	Monitoring Source Control Activity with Pulse . . . . .	47
	Configuring Pulse . . . . .	47
	About Starting Pulse . . . . .	48
	Viewing Source Control Activity. . . . .	49
	Suspending Project Activity Monitoring . . . . .	50
	Closing Pulse . . . . .	50
	About History Reports . . . . .	50
	Generating History Reports. . . . .	51
	About Difference Reports . . . . .	52
	Generating Difference Reports . . . . .	52
<b>Part 2</b>	<b>IDE Reference . . . . .</b>	<b>55</b>
	Introduction . . . . .	56

<b>Chapter 5</b>	<b>ColdFusion Studio</b>	<b>57</b>
	Introduction	58
	Accessing Supported Features	58
	Setting Up Source Control Projects	59
	Setting Up Projects for Access by Multiple-Users	59
	Selecting a Source Control Provider	60
	Mapping Projects to Source Control	60
	Adding Files to Source Control	61
	Removing Files from Source Control	62
	Using Source Control	62
	Getting Files	63
	Checking Out Files	63
	Undoing Checkout	63
	Checking In Files	63
<b>Chapter 6</b>	<b>PowerBuilder</b>	<b>65</b>
	Introduction	66
	About Version Manager Project Structure	66
	Accessing Supported Features in PowerBuilder	67
	Setting Up Source Control Projects in PowerBuilder	68
	Connecting PowerBuilder Workspaces to Source Control	68
	Adding Objects to Source Control	70
	Configuring Workstations in a Multi-User Environment	71
	Removing Objects from Source Control	72
	Disconnecting Workspaces from Source Control	72
	Using Source Control with PowerBuilder	73
	Getting Objects	73
	Checking Out Objects	73
	Undoing Checkout	74
	Checking In Objects	74
	Adding New Objects	75
	Adding New Targets or PBLs	75
<b>Chapter 7</b>	<b>Rational Application Developer (Eclipse 3 and 4)</b>	<b>77</b>
	Introduction	78
	Accessing Supported Features	78
	Setting Up Source Control Projects	79
	Excluding Files and Directories from Source Control	80
	Connecting Projects to Source Control	80
	Connecting Additional Workstations to a Source Control Project	82
	Adding New Files to Source Control	83
	Disconnecting Projects from Source Control	84
	Removing Files from Source Control	84
	Using Source Control	84
	Viewing Source Control Status	84
	Getting Files	85
	Checking Out Files	86
	Locking Files	86

Undoing Checkout . . . . .	87
Checking In Files . . . . .	87
Using Rename or Move (Refactoring) . . . . .	88
Using Local Mode . . . . .	89
Working Offline . . . . .	90
Synchronizing Your Workspace with Source Control . . . . .	91
Comparing with Local History . . . . .	94
Replacing with Local History . . . . .	94

## Chapter 8

### **Rational Application Developer Rich Integration (Eclipse 3 and 4) . . . . .**

**95**

Introduction . . . . .	96
Accessing Supported Features . . . . .	97
Integration Overview . . . . .	98
Working Offline . . . . .	99
SBM Integration . . . . .	99
Collaborative Process Overview . . . . .	99
Using Workspaces . . . . .	100
Working on Files Without Locking Them . . . . .	101
Checking Out Files with Locks . . . . .	102
Setting Up Source Control Projects . . . . .	102
Excluding Files and Directories from Source Control . . . . .	103
Migrating Projects from the Previous Source Control Integration . . . . .	103
Adding Projects to Source Control . . . . .	104
Connecting Additional Workstations to an Existing Source Control Project . . . . .	106
Disconnecting Projects from Source Control . . . . .	108
Using Source Control . . . . .	108
Viewing Connection Information . . . . .	108
Viewing Source Control Status . . . . .	109
Working in the History View . . . . .	109
Assigning Version Labels . . . . .	111
Getting Files . . . . .	112
Checking Out Files . . . . .	113
Undoing Checkout . . . . .	114
Checking In Files . . . . .	115
Using Rename or Move (Refactoring) . . . . .	118
Comparing and Synchronizing Your Workspace with Source Control . . . . .	118
Comparing with the Latest Revision . . . . .	122
Comparing with Local History . . . . .	123
Comparing Workfiles with Each Other . . . . .	123
Replacing with Local History . . . . .	124
Replacing with Latest Revision . . . . .	125
Associating and Working on SBM Issues . . . . .	125
Issue Management Workflow . . . . .	126
Setting Up Your IDE Folder . . . . .	127
Changing SBM Connection Information . . . . .	128
Displaying Reports and Issues . . . . .	128
Submitting and Modifying Issues . . . . .	129
Associating Issues with Files . . . . .	130
Setting Default Options . . . . .	131

	Source Control Options . . . . .	131
	Issue Management Options . . . . .	137
<b>Chapter 9</b>	<b>Visual Studio SCC Integration . . . . .</b>	<b>139</b>
	Introduction . . . . .	140
	Accessing Supported Features . . . . .	140
	About Visual Basic Files . . . . .	141
	Setting Up Source Control Projects . . . . .	142
	Upgrading to Visual Studio 2005 from Visual Studio .NET 2003. . . . .	142
	Configuring Source Control Behavior . . . . .	142
	Configuring Web Projects . . . . .	142
	Excluding or Removing Files from Source Control . . . . .	143
	Adding Visual Studio Files to Source Control . . . . .	143
	Connecting Additional Workstations to a Source Control Project . . . . .	145
	Using Source Control . . . . .	146
	Getting Files . . . . .	146
	Checking Out Files . . . . .	146
	Undoing Checkout . . . . .	147
	Checking In Files . . . . .	148
<b>Chapter 10</b>	<b>Visual Studio Rich Integration . . . . .</b>	<b>149</b>
	Introduction . . . . .	150
	Accessing Supported Features . . . . .	151
	About the Source Control Toolbar . . . . .	152
	Visual Studio Rich Integration Overview . . . . .	152
	Solutions Business Manager Integration . . . . .	153
	Supported Project Types . . . . .	153
	Rebinding a Solution . . . . .	153
	Collaborative Process Overview . . . . .	153
	Using Workspaces . . . . .	154
	Working on Files Without Locking Them . . . . .	155
	Checking Out (Locking) Files . . . . .	156
	Migrating and Converting Visual Studio Solutions . . . . .	157
	Migrating from Visual Studio 2003 to Visual Studio RIDE . . . . .	157
	Migrating from Visual Studio SCC to Visual Studio RIDE . . . . .	158
	Migrating from Visual Studio 2005 to Visual Studio RIDE . . . . .	160
	Working with Web Projects . . . . .	161
	Working with Branches . . . . .	161
	Viewing Branched Files . . . . .	162
	How Should I Branch My Files? . . . . .	163
	Automatic Label-Based Branching . . . . .	163
	Manual Branching . . . . .	165
	Editing Revisions on a Branch . . . . .	166
	Checking In Branched Files . . . . .	167
	Setting Up Source Control Projects . . . . .	167
	Adding Solutions and Projects to Version Manager . . . . .	168
	Opening Solutions and Projects from Source Control . . . . .	170
	Opening Solutions not Added using RIDE . . . . .	174

Editing Files . . . . .	174
Reviewing File History . . . . .	175
Getting Specific Files or Folders. . . . .	176
Checking Out Files. . . . .	178
Undoing Checkout. . . . .	180
Editing Files . . . . .	181
Refreshing File Status . . . . .	181
Reviewing Local Changes . . . . .	181
Checking In Files. . . . .	182
Labeling Revisions. . . . .	186
Promoting Revisions . . . . .	188
Working While Offline . . . . .	188
Setting Default Options for Dialog Boxes . . . . .	189
Configuring Client/Server-Side Processing . . . . .	191
Setting Encoding and Display Options . . . . .	192
Comparing and Synchronizing Workspaces . . . . .	193
About the Merge Process . . . . .	193
Important Refactoring Considerations . . . . .	193
Comparing Workspaces . . . . .	194
Getting All Updates from Version Manager . . . . .	195
Committing Local Changes to Version Manager. . . . .	197
Synchronizing Workspaces . . . . .	198
Comparing Files and Resolving Conflicts . . . . .	199
About File Comparison . . . . .	199
Comparing Files . . . . .	199
Reviewing and Resolving Conflicts. . . . .	201
Associating and Working on SBM Issues . . . . .	204
Issue Management Workflow . . . . .	204
Setting Up Your IDE Folder. . . . .	205
Defining Association Options. . . . .	206
Logging into SBM . . . . .	207
Displaying Reports and Issues . . . . .	208
Submitting and Modifying Issues. . . . .	208
Associating Issues. . . . .	209
<b>Appendix A: Naming Conventions and Restrictions . . . . .</b>	<b>211</b>
General Naming Conventions and Restrictions. . . . .	212
Prohibited Characters for Files and Directories . . . . .	212
Naming Considerations for Cross-Platform Environments . . . . .	212
Specific Naming Conventions and Restrictions. . . . .	213
<b>Index. . . . .</b>	<b>215</b>



## Part 1

---

# The Version Manager IDE Client

Overview of Version Manager Source Control	11
Setting Up Source Control with SCC IDEs	17
Using Source Control	27
Accessing Source Control Information	45

# Introduction

Contents	This part of the manual contains conceptual and procedural information common to setting up and using the Version Manager IDE client with any IDE.
Purpose	The purpose of this part of the manual is to provide a conceptual and procedural overview of how to set up and use the Version Manager IDE client outside the context of a specific IDE.
Unsupported features	Some IDEs do not support all of the features described in this part of the manual. For information on which features are supported and how to access them, see <a href="#">Part 2, "IDE Reference," on page 55</a> .
Additional information	Use this part of the manual in conjunction with these additional sources of information.

For more information about...	See the...
Setting up and configuring Version Manager	Version Manager Administrator's Guide
Version Manager features and concepts	Version Manager User's Guide
Setting up and using your IDE with source control	Documentation and online help provided by the vendor of your IDE

## Chapter 1

---

# Overview of Version Manager Source Control

Introduction	12
Source Control Concepts	12
Available Source Control Information	14
How Version Manager Integrates with IDEs	15

# Introduction

Purpose	This chapter is an introduction to key source control concepts and the options available to users of Version Manager. This chapter describes Version Manager features, such as nested project support and project-wide version labeling, that are now available from within supported IDEs.
For more information	For more detailed conceptual information about source control and working with Version Manager, see the <i>Version Manager User's Guide</i> .
IDE-specific information	For information about the Version Manager features supported in your IDE, see <a href="#">Part 2, "IDE Reference,"</a> on page 55.

## Source Control Concepts

Source control	Source control is a way to manage changes to the individual components of software (or other content) being developed by a team. You can control access and manage change to any file by placing it under source control.
----------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Project Databases

Definition	A <i>project database</i> is a hierarchical collection of projects, subprojects, and files under source control. A project database defines a common source control configuration for all projects and subprojects contained within it. See <a href="#">"Creating and Configuring Project Databases"</a> on page 20.
------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Projects and Subprojects

Definition	<i>Projects</i> are logical groupings of subprojects and files under source control. <i>Subprojects</i> are projects contained within other projects.
------------	-------------------------------------------------------------------------------------------------------------------------------------------------------

### Archives

Definition	<p>An <i>archive</i> is a Version Manager file that stores the changes you make to a workfile. Whenever you check in a modified workfile, its archive is updated. Archives also store the following:</p> <ul style="list-style-type: none"><li>■ A description of the changes</li><li>■ The ID of the user who made the changes</li><li>■ The date and time when the changes were made</li><li>■ Version label and promotion group information</li></ul> <p>An archive is created for each file you add to source control.</p>
------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Revisions

Definition	A <i>revision</i> is a record within an archive that stores the changes you made to a workfile and checked in on a particular occasion. When you check out a revision, Version Manager re-creates that version of the workfile in the specified workfile location.
Initial and tip revisions	The oldest revision is called the <i>initial revision</i> and is, by default, numbered 1.0. For each new revision you check in, the revision number increments by one; for example, 1.1 to 1.2. The newest revision in the archive is called the <i>tip</i> .
Default revision	The <i>default revision</i> is the revision that is acted on if no other is selected. The default revision is, by default, the tip revision. Through the Version Manager desktop client, you can set the default revision to a specific version label or revision.

## Workfiles

Definition	A <i>workfile</i> is any file that you check in to Version Manager to create a new revision or archive. When you check out a revision, Version Manager copies it as a workfile to your workfile location.
------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Locks

Definition	A <i>lock</i> is a way to keep other team members from modifying a revision while you are working on it. They can always view or get the file, but they cannot check in changes and overwrite the locked revision.
Duration of lock	When you check out a revision, Version Manager locks it. The lock stays in place until you either check in the workfile or undo the checkout on the revision.

## Version Labels

Definition	A <i>version label</i> is a tag used to identify a specific revision within an archive. Version labels enable you to efficiently work with revisions from multiple archives that belong together, such as all of the revisions that make up a beta version of an application. See <a href="#">"About Version Labels" on page 34</a> .
------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Promotion

Definition	<i>Promotion</i> is a means to control development based on milestones in the development cycle, from the design phase to final release. Promotion must be set up through the Version Manager desktop client.
Promotion model	A <i>promotion model</i> is a hierarchy of milestones in a development cycle that enforces active development work to the lowest level of the model.
Promotion groups	Each milestone in a promotion model is represented by a <i>promotion group</i> . Examples of promotion groups include Development, QA, and Production.  At checkout, you can select revisions based on promotion groups. At check-in, you can assign promotion groups to new revisions. From the Options dialog box, you can assign promotion groups to revisions, promote revisions to the next group, change which promotion group is assigned to revisions, and remove promotion groups from revisions. See <a href="#">"About Promotion Groups" on page 38</a> .

## Branches

- Definition A *branch* is a separate line of development consisting of one or more revisions that diverge from a revision on the main line of development (trunk) or on another branch. You can create branches when checking in new revisions.
- Why use branching? Branching lets you develop alternate variations of a file in parallel with the continued development of the revision from which it was branched.

## Sharing

- Definition *Sharing* lets you access a file from multiple Version Manager projects while keeping all changes in a single shared archive. You share an archive when you want to use and edit a given file from within multiple IDE projects. See ["About Sharing Files Across Projects" on page 25](#).

## Workspaces

- Definition A *workspace* is a collection of work settings defined for a project database, which includes the work settings for all of the projects and archives contained within the project database. These work settings include workfile location, default version, base version, and branch version. See ["About Version Manager Workspaces" on page 21](#).

# Available Source Control Information

You can access four types of information about the files you have placed under source control.

## Viewing Properties of Files Under Source Control

- Properties dialog box You can use the *Version Manager Properties dialog box* to view information on the following: archives, revisions, version labels, and promotion groups. See ["About Properties" on page 46](#).

## Monitoring Source Control Activity

- Pulse *Pulse* allows users who are logged into the same projects to monitor certain source control events. For example, Pulse notifies you when another user has added a new file or checked in changes to an existing file. You can also view results messages for your own source control actions. See ["Monitoring Source Control Activity with Pulse" on page 47](#).

## Viewing Historical Archive or Revision Activity

- History report A *history report* summarizes information about archives and/or revisions. You can select revisions for the report by version label, promotion group, date, author, owner, and lock. See ["About History Reports" on page 50](#).

## Comparing Files or Revisions

**Difference report** A *difference report* allows you to compare two files side-by-side to see the additions, deletions, and changes made to the files. You can compare specific revisions, workfiles with revisions, or two workfiles. See ["About Difference Reports" on page 52](#).

## How Version Manager Integrates with IDEs

Version Manager supports two conventions for integrating with IDEs: SCC and Eclipse. The following list shows the method used by each supported IDE:

- Source Code Control (SCC):
  - ColdFusion Studio
  - PowerBuilder
  - Rational Rose
  - Visual Studio
- Eclipse:
  - Eclipse
  - Rational Application Developer

In addition to the above, Version Manager leverages the issue management features of TeamTrack to support rich integrations to the following IDEs:

- Eclipse 3 and Rational Application Developer 6 & 7  
(See ["Rational Application Developer Rich Integration \(Eclipse 3 and 4\)" on page 95](#).)
- Visual Studio  
(See ["Visual Studio Rich Integration" on page 149](#).)



**NOTE** See the Version Manager readme for a list of the specific IDE versions supported in this release of Version Manager.

Additional  
information

For information on setting up Version Manager with your IDE, see the following:

For more information about ...	See ...
Procedures common to integrating with SCC compliant IDEs	<a href="#">Chapter 2, "Setting Up Source Control with SCC IDEs" on page 17</a>
Procedures for integrating with specific IDEs	<a href="#">Part 2, "IDE Reference," on page 55</a>





## Chapter 2

---

# Setting Up Source Control with SCC IDEs

Introduction	18
Recommended Workflow	18
About Selecting a Source Control Provider	19
Creating and Configuring Project Databases	20
About Setting Defaults for Version Manager Options	22
About Creating Source Control Projects	23
About Adding Files to Source Control	24
About Sharing Files Across Projects	25
About Removing Files from Source Control	26

# Introduction

Contents and purpose	This chapter contains conceptual and procedural information common to setting up the Version Manager IDE client with supported SCC compliant IDEs. The purpose of this chapter is to help administrators create and configure project databases and projects.
IDE-specific information	For information specific to your IDE, see <a href="#">Part 2, "IDE Reference," on page 55</a> .

## Recommended Workflow

We recommend that a project lead or administrator create and set up source control project databases and projects. Once these are established, all users can begin using source control from within their IDE.

### Administrators

**We recommend that Project leads or administrators first follow these steps:**

- 1 Create a Version Manager project database which will contain the source control project associated with the IDE project. Use the Version Manager desktop client to create the project database. See ["Creating and Configuring Project Databases" on page 20](#).
- 2 Add the IDE project and files to source control. See [Part 2, "IDE Reference," on page 55](#).
- 3 Configure the source control project in the Version Manager desktop client. Configuration includes setting up security and promotion groups. For more information on project configuration, see the *Version Manager Administrator's Guide*.

### All Users

**All users can then perform the following source control tasks:**

- Define user settings.
- Get or check out and edit files. See ["About Getting Files" on page 28](#) or ["About Checking Out Files" on page 30](#).
- Check in files. Users can assign version labels and revision numbers to revisions during check-in. See ["About Checking In Files" on page 33](#).
- View source control information:
  - Archive and revision properties. See ["About Properties" on page 46](#).
  - Project activity. See ["Monitoring Source Control Activity with Pulse" on page 47](#).
  - History and difference reports. See ["About History Reports" on page 50](#) and ["About Difference Reports" on page 52](#).

IDE-specific information	For information on accessing these functions, see <a href="#">Part 2, "IDE Reference," on page 55</a> .
--------------------------	---------------------------------------------------------------------------------------------------------

# About Selecting a Source Control Provider

If multiple IDE clients are installed on your system, you can choose which one is active. For example, if the Version Manager Source Code Control (SCC) Interface 6.0 is installed to your system and you wish to continue using it with existing 5.3/6.0 projects, you can make it the active source control provider.

By default, the Version Manager IDE client becomes the active source control provider when it is installed.

## Selecting an SCC Provider

To select a provider, complete the following steps:

- 1 Exit any application you intend to use with a IDE client.
- 2 Select Programs | Serena | Version Manager | Version Manager IDE Client | Version Manager SCC Admin from the Start menu. The SCC Admin tool appears.

Available source control providers are listed under **Available Source Control Providers**. Information about the selected provider appears in the Description frame.

- 3 Select the provider you wish to activate from the list.

To work with this project format...	Select this source control provider...
Version Manager 8.0 or higher	Version Manager
Version Manager 6.5 to 7.5	PVCS Source Control
Version Manager 5.3/6.0	PVCS Version Manager
Dimensions 10 or higher	Dimensions
Dimensions 6 to 9	Dimensions



**NOTE** If PVCS Version Manager SourceBridge or Trackerlink is installed to your system, it appears as an available source control provider. Selecting it activates the source control provider that it is configured to work with. For more information, see the TeamTrack and/or Tracker documentation.

- 4 Click **OK**.

Restart your IDE

The provider you selected will be active the next time you launch your IDE. Every time you change source control providers, you must restart your IDE for the change to take effect.



**NOTE** For information on the **CLI Service** button, see ["Stopping the PVCSCLIServ Service" on page 20](#).

## Testing an SCC Provider

You can test the currently active provider to make sure it initiates correctly.

### To test an SCC provider:

- 1 Exit any application you intend to use with a IDE client.
- 2 Select Programs | Serena | Version Manager | Version Manager IDE Client | Version Manager SCC Admin from the Start menu. The SCC Admin tool appears.



**IMPORTANT!** If the SCC provider you want to test is not the currently active provider, follow the procedure ["Selecting an SCC Provider" on page 19](#) and exit the SCC Admin tool before returning to this procedure.

- 3 Click the **Test SCC** button.
- 4 One of the following will occur:
  - **Success:** A dialog from your SCC provider appears. Dismiss it. In the Test column, a green check mark appears next to each item indicating that SCC initiated successfully.
  - **Failure:** In the Test column, a red check mark appears next to the steps that failed.




**NOTE** This information may be requested by technical support if you call with a problem.

## Stopping the PVCSCCLIServ Service

The PVCSCCLIServ service is launched if your IDE uses the same JVM as Version Manager. This ensures that each JVM is run in a separate process space. Once this service is launched, it will continue to run until you reboot or manually unload it.

### To stop the PVCSCCLIServ service:

- 1 Select Programs | Serena | Version Manager | Version Manager IDE Client | Version Manager SCC Admin from the Start menu. The SCC Admin tool appears.
-  2 Click the button with the red dot. If the dot is brown, the service is not running.
- 3 A prompt appears asking if you really want to stop the service. Click **Yes**.

## Creating and Configuring Project Databases

About project databases      A Version Manager project database is a hierarchical collection of projects, subprojects, and versioned files. It defines a common configuration for all projects and subprojects contained within it.

Default project database      When you install the Version Manager IDE client, you can create a default project database for your source control projects. By default, this database is located at:

%ALLUSERSPROFILE%\Application Data\Serena\VM\vmdevint

For example:

C:\Documents and Settings\All Users\Application Data\Serena\VM\vmdevint.

You can set up this database for use with your source control projects, or create and configure a different database using the Version Manager desktop client.

**Configuration options** You can set up security, promotion models, and version labels for your project from the Version Manager desktop client. You can also configure workspace settings such as default version, default promotion group, branch version, and base branch.

**For more information** See the *Serena ChangeMan Version Manager Administrator's Guide* for information on planning, creating, and configuring project databases and projects.

## About Version Manager Workspaces

**Definition** A Version Manager workspace is a collection of work settings defined for a project database which includes all of the projects and archives contained within the project database. With Version Manager IDE client projects, you can set the default version, default promotion group, branch version, and base branch.



**NOTE** You cannot override the existing project workfile location with workspace settings. If you will use the Version Manager desktop client or project command line interface with your IDE projects, set your workspace to reference the existing workfile directories. This will ensure that files will always check out to the correct location.

**Public, private and root workspaces** You can define both *public* and *private* workspaces. The *root* workspace is the default public workspace. The root workspace is the active workspace for all users for whom private workspaces are not defined.

**Workspaces in a team environment** Private workspaces are typically created by individual project team members so they can customize the project work settings without affecting other team members. For settings in a particular workspace to take effect when a user logs into a Version Manager project, that workspace must be made the default for that user. If no default private workspace is set, the work settings defined in the root workspace will take effect.



**IMPORTANT!** You must exit and restart your IDE for new workspace settings to take effect.

**For more information** See the *Version Manager User's Guide*.

## Launching the Version Manager Desktop Client

**Configuring projects** To configure your Version Manager projects, you can launch the Version Manager desktop client from within your IDE.

### To launch the Version Manager desktop client:

- 1 Open the Version Manager Options dialog box. (For the IDE-specific menu command, see [Part 2, "IDE Reference," on page 55.](#))
- 2 Select the General tab.
- 3 Click the **Version Manager** button in the Launch group.

For more information See the *Version Manager Administrator's Guide*.

## About Setting Defaults for Version Manager Options

To save time and improve workflow, you can configure the Version Manager IDE client to reflect the way you normally work. This section describes how to set the following defaults:

- The default behavior for check-in and undo checkout operations
- Which program is used to display history reports
- Which Version Manager workspace to use
- Which Version Manager project database to open by default for operations such as version labeling

Scope of settings The settings described in this section are not project specific. Any changes to these settings will affect your working environment and persist from project to project.

### Setting Defaults

**To set defaults:**

- 1 Open the Version Manager Options dialog box. (For the IDE-specific menu command, see [Part 2, "IDE Reference," on page 55](#).)
- 2 Under the Version Manager tab do any of the following:

- |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Default project database | <ul style="list-style-type: none"><li>■ To specify which project database will appear by default every time you add projects to source control, get projects from source control, or assign or modify version labels, enter the project database name and path in the <b>Default Project Database</b> field, or click the Browse button to select one. You can override this selection at the time of the operation.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| History report viewer    | <ul style="list-style-type: none"><li>■ To specify a text viewer for viewing history reports, enter a text viewer executable name and path in the <b>History Report Viewer</b> field, or click the Browse button to select one. If no viewer is selected, Version Manager uses the default Windows .txt file viewer.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Workspaces               | <ul style="list-style-type: none"><li>■ Specify the Version Manager workspace to use. Your IDE controls the actual workfile location regardless of the Version Manager workspace you select, but the following values, as defined in the workspace, will be in effect:<ul style="list-style-type: none"><li>• <b>Default Version:</b> The revision Version Manager operates on (for actions such as checking out) when you do not specify a revision number or version label. For example, you could set the default version to be a specific floating label on a branch.</li><li>• <b>Base Version:</b> Used to facilitate automatic branching.</li><li>• <b>Branch Version:</b> Used to facilitate automatic branching.</li><li>• <b>Default Promotion Group:</b> The lowest-level promotion group of the promotion model, if one is in effect.</li></ul></li></ul> |

In the **Workspace** field you can:

- Select the **Default Workspace** as defined in Version Manager.
- Select the **Root Workspace** as defined in Version Manager.
- Enter or select a workspace that is defined in Version Manager. The last 5 entries are retained in the list.

To enter a nested workspace, enter it as:  
ParentWorkspace/ChildWorkspace.

To enter a private workspace, select the **Private** check box and enter the user ID of the owner of the workspace in the **Owner** field.



**NOTE** If a private workspace is nested below a public workspace, you cannot access it from the IDE client.

For more information on defining and using workspaces, see the *Serena ChangeMan Version Manager Administrator's Guide*.

#### Check-in options

- To specify default behavior when you check in a workfile that is unchanged or older than the previous revision, select one of the following from the **If workfile unchanged or older** drop-down menu in the Check In Options group:
  - **Prompt** asks what you want to do if the workfile is unchanged.
  - **Check in**, which is the default option, checks in the workfile even if it is unchanged.
  - **Don't check in** does not check in the workfile.

#### Undo checkout options

- To specify what happens to workfiles by default when you undo a checkout, select one of the following from the **Replace options** drop-down list in the Undo Check Out Options group:
  - **Replace with latest revision**, which is the default option, replaces the workfile with the tip revision.
  - **Replace with original** replaces the workfile with a fresh copy of the revision that was originally checked out.
  - **Leave alone** leaves the workfile as it is at the time of the operation.
- To make workfiles writable by default after you undo a checkout, select the **Make workfile writable** check box. This option is available only if you chose to replace the workfile via the **Replace options** drop-down list.

3 Click **OK**.

## About Creating Source Control Projects

Once you have created and saved a project in your IDE, you can add it to an existing Version Manager project database. By default, when you add an IDE project to source control, a new Version Manager project is created.

#### IDE-specific instructions

For instructions specific to your IDE, see [Part 2, "IDE Reference," on page 55](#).

## About Adding Files to Source Control

By default, the following occurs when you add files to source control:

- If any project files are located in nested subdirectories under the root working directory of the IDE project, Version Manager creates nested subprojects that mirror the directory structure of the IDE project's workspace.
- Archives are created in the Version Manager project directories for each file that is added. By default, the name of each archive matches its originating workfile with the addition of an -arc suffix. Custom suffixes can be defined in the project database or project configuration.
- For each archive, the initial revision is checked in and no version label is assigned to the initial revision.

IDE-specific instructions

For instructions specific to your IDE, see [Part 2, "IDE Reference," on page 55](#).

Advanced options

If your IDE supports advanced options, you can assign a promotion group or version label to the initial revision.

## Returning Files to Source Control

If you re-add a file that was previously removed from source control, one of the following will occur:

- If the original archive is still present in the project archive directory, you will be prompted to check the file into the existing archive as a new revision.
- If the archive has been moved or deleted from the project archive directory, a new archive will be created, and the file will be checked in as the initial revision.

## Advanced Add Options

After you click the Advanced button in your IDE's add dialog box, the Advanced Add dialog box appears.

**To set advanced add options:**

General tab options

- 1 If you are adding files to a project database or project with a defined promotion model, you can assign a promotion group to the initial revision by entering a lowest-level promotion group in the **Lowest-level promotion group** field, or by clicking the Browse button to select one. (Use the Version Manager desktop client to define a promotion model.)

Advanced tab options

- 2 Under the Advanced tab, do any of the following:
  - To assign a version label to the initial revision, enter the version label in the **Version Label** field or click the Browse button to select one.
  - To keep the version label assigned to the latest revision in the archive, select the **Float label with tip** check box.
- 3 Click **OK**.
- 4 Your IDE's add dialog box may reappear; click **OK**.



The selected files are added to source control, creating archives in the archive directory specified for the project.

**Shared file** If you want to share a file with multiple Version Manager projects, see the next section.

## About Sharing Files Across Projects

**Why share?** It may be necessary for multiple development projects to access and edit the same file. To do this, multiple projects within a Version Manager project database can share a single Version Manager archive.

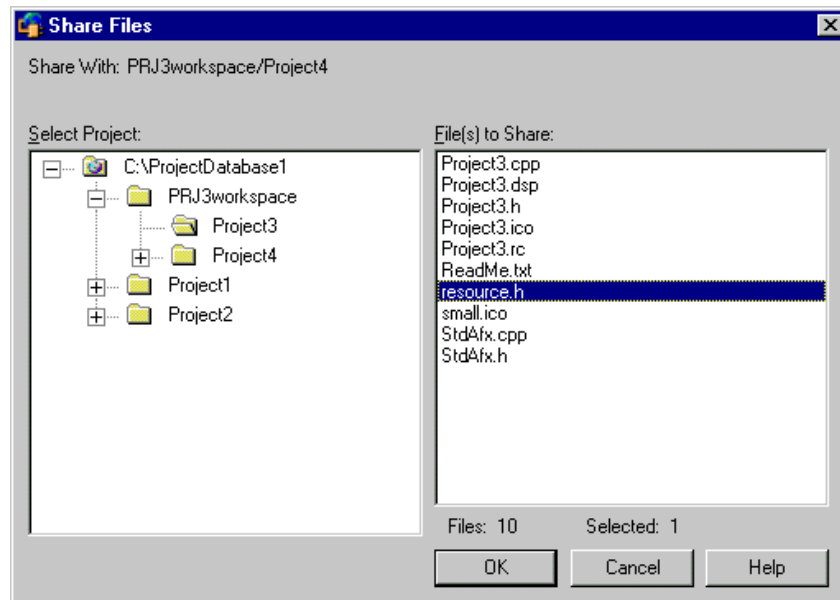
**How share works** When you share an archive, Version Manager copies the tip revision to your working directory. You can then check out, modify, and check in the file like any other. The archive is not physically copied, but changes to the file are stored in the shared archive.

**NOTE** You cannot share files across projects that are located in separate project databases. To share files across projects, the projects must be located in the current project database.

## Sharing Files

**To share a file:**

- 1 Open the Share Files dialog box. (For the IDE-specific menu command, see [Part 2, "IDE Reference,"](#) on page 55.)



The Share Files dialog box appears with the current Version Manager project displayed.

- 2 Under **Select Project**, select the project that contains the file(s) you want to share. Click the plus signs to expand the project database tree. The **File(s) to Share** pane displays all files in the selected project.
- 3 Under **File(s) to Share**, select the files whose archives you want to access.

- 4 Click **OK**. The latest revision of each file is copied to your project working directory.

## About Removing Files from Source Control

When you remove a file from source control, Version Manager does not delete the workfile or Version Manager archive. The IDE project remains unchanged.

IDE-specific  
information

For the IDE-specific menu command, see [Part 2, "IDE Reference," on page 55](#).

Returning files to  
source control

For information on returning files to source control, see ["Returning Files to Source Control" on page 24](#).

## Chapter 3

---

# Using Source Control

Introduction	28
Logging In to Version Manager Projects	28
About Getting Files	28
About Checking Out Files	30
About Undoing Checkout	32
About Checking In Files	33
About Version Labels	34
About Promotion Groups	38

## Introduction

**Contents** This chapter contains information common to performing the following operations with supported IDEs, including default behaviors and advanced options:

- Logging in
- Getting files
- Checking out files
- Undoing a checkout
- Checking in files
- Managing version labels

**Purpose** The purpose of this chapter is to help your development team effectively use the source control features available through the Version Manager IDE client.



**NOTE** Your IDE may not support some of the features described in this chapter, such as advanced options and certain default behaviors.

**IDE-specific information** For information specific to your IDE, see [Part 2, "IDE Reference," on page 55](#).

## Logging In to Version Manager Projects

Depending on how security is configured for the Version Manager project, you may be prompted for a user ID and password when opening a project.

**If the Version Manager Login dialog box appears, do the following:**

- 1** Enter your user ID in the **User Name** field and your password in the **Password** field.  
The **Location** field displays the name of the project database that contains the project you are logging in to.
- 2** Click **OK**.

## About Getting Files

**Defaults** When you get a file, the following occurs by default:

- A read-only copy of the default revision is created in the workfile location.
- If a writable workfile exists in the current workfile location, a prompt displays before overwriting the workfile. If a read-only workfile exists in the current workfile location, there is no prompt for confirmation.
- The archive remains unlocked, allowing other users to check out and edit the file.

IDE-specific instructions For instructions specific to your IDE, see [Part 2, "IDE Reference," on page 55](#).

Advanced options If your IDE supports advanced options, you can select a different revision or browse for a revision associated with a particular version label or promotion group in the Advanced Get dialog box.

## Advanced Get Options

After you click the Advanced button in your IDE's get dialog box, the Advanced Get dialog box appears.

### To select advanced get options:

General tab options

1 Under the General tab, do any of the following:

- To select a revision other than the default revision, enter a revision number or version label in the **Revision** field or click the Browse button to select one.



**NOTE** To specify a version label that begins with a numeral, precede the version label with a backslash (\).

- To specify what to do if a writable copy of the file already exists in the working directory, select one of the following from the **If Workfile Exists** drop-down list:
  - **Prompt:** asks what you want to do if a duplicate file is detected during a get
  - **Overwrite:** replaces the existing workfile with the new file
  - **Don't Overwrite:** does not add the file to the workfile location
- To get a revision by promotion group, enter a promotion group in the **Promotion Group** field, or click the Browse button to select one. The files must belong to a project or project database that has a defined promotion model.
- To get a writable copy of the workfile so that you can edit it, select the **Make workfile writable** check box.
- **Lookup:** (Applies only if a promotion model is in effect.) Select one of the following options:
  - **Lookup revision based on Revision:** The revision specified in the **Revision** field will be acted on. You can enter/select a revision number, version label, or promotion group to specify the desired revision.  
  
If you select **[Default Revision]** in the **Revision** field, the revision specified by the workspace settings or configuration file will be acted on; if no default value is found, the Tip of the Trunk will be acted on.
  - **Lookup revision based on Promotion Group:** The revision assigned to the promotion group in the **Promotion Group** field will be retrieved. If such a revision is not found, the operation will climb the promotion model and act on the revision assigned to the lowest currently assigned group in the promotion model.

Advanced tab options

2 Under the Advanced tab of the Advanced Get dialog box, do any of the following:

- To select a revision by date, select the **Check out by date** check box and then select one of the following:
    - **Revision newer than workfile:** gets the revision only if it is newer than the date of its corresponding workfile
    - **Revision newer than:** gets a revision that was last checked in after the specified date and time
    - **Revision checked in before:** gets a revision that was last checked in before the specified date and time
  - To update the timestamp of the file to the current date and time, select the **Set workfile time to current time** check box.
- 3 Click **OK**.
- 4 Your IDE's get dialog box may redisplay; click **OK**.

The selected revisions are copied to the workfile location.

## Getting IDE Projects from Source Control

Getting a project from source control copies the latest revision of every file in the project to a directory of your choice. This is useful if you must update your working copy of the project with all recent changes made by other team members.

IDE-specific instructions	Some IDEs do not support this feature. For instructions specific to your IDE, see <a href="#">Part 2, "IDE Reference," on page 55</a> .
---------------------------	-----------------------------------------------------------------------------------------------------------------------------------------

## About Checking Out Files

Defaults	<p>To edit a file that is under source control, you must first check it out. When you check out a file, the following occurs by default:</p> <ul style="list-style-type: none"><li>■ The default revision is locked.</li><li>■ A writable copy of the default revision is created in the workfile location.</li><li>■ If a writable workfile exists in the current workfile location, a prompt displays before the workfile is overwritten. If a read-only workfile exists in the current workfile location, there is no prompt for confirmation.</li><li>■ The promotion group currently assigned to the revision is retained, if a promotion model is in effect.</li></ul>
Shared files	When you check a file out from a shared archive, Version Manager copies a revision of the file to your working directory and locks the shared archive. Changes to the file are stored in the file's shared archive.
IDE-specific instructions	For instructions specific to your IDE, see <a href="#">Part 2, "IDE Reference," on page 55</a> .
Advanced options	If your IDE supports advanced options, you can select a different revision or browse for a revision associated with a particular date, promotion group, or version label.

## Advanced Checkout Options

After you click the Advanced button in your IDE's checkout dialog box, the Advanced Check Out dialog box appears.

### To select advanced checkout options:

General tab  
options

1 Under the General tab, do any of the following:

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



**NOTE** To specify a version label that begins with a numeral, precede the version label with a backslash (\).

- To specify what to do if a writable copy of the file already exists in the working directory, select one of the following from the **If Workfile Exists** drop-down list:
  - **Prompt:** displays a prompt asking what to do when a duplicate workfile exists
  - **Overwrite:** replaces the duplicate file with the new file
  - **Don't overwrite:** does not add the file to the workfile location
- If the selected files are associated with a project database or project with a defined promotion model, enter a promotion group to associate with the workfile, or click the Browse button in the **Lowest-level promotion group** field to select one.
- **Lookup:** (Applies only if a promotion model is in effect.) Select one of the following options:
  - **Based on Revision:** The revision specified in the **Revision** field will be acted on. You can enter/select a revision number, version label, or promotion group to specify the desired revision.

If you select **[Default Revision]** in the **Revision** field, the revision specified by the workspace settings or configuration file will be acted on; if no default value is found, the Tip of the Trunk will be acted on.

The promotion group specified in the **Lowest-level promotion group** field will be assigned to the revision.

- **Based on Promotion group:** The revision assigned to the promotion group specified in the **Lowest-level promotion group** field will be acted on.

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

Advanced tab  
options

2 Under the Advanced tab of the Advanced Check Out dialog box, do any of the following:

- To select a revision by date, select the **Check out by date** check box and then select one of the following:

- **Revision newer than workfile:** checks out the revision only if it is newer than the date of its corresponding workfile
  - **Revision newer than:** checks out a revision that was last checked in after the specified date and time
  - **Revision checked in before:** checks out a revision that was last checked in before the specified date and time
- To update the timestamp of the file to the current date and time, select the **Set workfile time to current time** check box.
- 3 Click **OK**.
  - 4 Your IDE's checkout dialog box may redisplay; click **OK**.

Writable copies of the selected revisions are checked out to the workfile location and the revisions are locked.

## About Undoing Checkout

**Defaults** Undoing a checkout unlocks a revision without updating the associated archive with changes. This allows other users to check out the revision. When you undo a checkout, the following occurs by default:

- The revision is unlocked.
- The workfile is replaced with a read-only copy of the latest revision.

**IDE-specific instructions** For instructions specific to your IDE, see [Part 2, "IDE Reference," on page 55](#).

**Advanced options** If your IDE supports advanced options, you can choose to leave the workfile writable, replace it with the original, or leave it alone.

## Advanced Undo Checkout Options

After you click the Advanced button in your IDE's undo checkout dialog box, the Advanced Undo Check Out dialog box appears.

### To select advanced undo checkout options:

- 1 To specify what to do with the current workfile in the working directory, select one of the following from the **Replace options** drop-down list:
  - **Replace with latest revision:** replaces the workfile with a copy of the latest revision of the file. Any changes you made to the workfile will be lost.
  - **Replace with original:** replaces the workfile with a copy of the locked revision of the file. Any changes you made to the workfile will be lost.
  - **Leave alone:** retains the workfile in its current state.
- 2 To leave a writable copy of the workfile in the working directory, select the **Make workfile writable** check box.
- 3 Click **OK**.



- 4 Your IDE's undo checkout dialog box may redisplay; click **OK**.

The archive is unlocked.

## About Checking In Files

**Defaults** Checking in a workfile preserves the changes to the file in a new revision. By default, the following occurs:

- A new revision is created and assigned the next number in sequence.
- A read-only workfile is left in the workfile location.
- The revision that was checked out is unlocked.



**NOTE** By default, the IDE client will check in a file even if it is unmodified or older than the latest revision. To change this default, see ["Setting Defaults" on page 22](#).

**IDE-specific instructions**

For instructions specific to your IDE, see [Part 2, "IDE Reference," on page 55](#).

**Advanced options**

If your IDE supports advanced options, you can assign a version label, force a branch, lock the new revision, or choose what happens if the workfile is unchanged or older than the previous revision.

## Advanced Check-In Options

After you click the Advanced button in your IDE's check-in dialog box, the Advanced Check In dialog box appears.

### To select advanced check-in options:

**General tab options**

- 1 Under the General tab, do any of the following:
  - To specify what to do if any of the files you are checking in are unchanged or older than the previous revisions, select one of the following from the **If workfile unchanged or older** drop-down list:
    - **Prompt:** displays a prompt asking what you want to do if the workfile is unchanged
    - **Check in:** checks in the file even if it is unchanged
    - **Don't check in:** does not check in the file
  - To lock the revision that is created when the file is checked in, select **Keep revision locked after check in**.

**Advanced tab options**

- 2 Under the Advanced tab of the Advanced Check In dialog box, do any of the following:
  - If you have multiple revisions of the file checked out, enter the revision in the **Revision** field that you want to check the file into, or click the Browse button to select one.

- Enter a new revision number in the **Revision** field to override the default increment. For example, if the archive currently contains two revisions (1.0 and 1.1), you can check the file in as revision 1.5.
  - To create a branch from the revision you are checking the file into, select the **Force branch** check box. A branch is a separate line of development consisting of one or more revisions that diverge from the main line of development (trunk).
  - To assign a version label to the revision, enter the version label in the **Version Label** field, or click the Browse button to select one.
  - To keep the version label associated with the latest (tip) revision of the current trunk or branch, select the **Float label with tip** check box.
  - To specify what to do if an identical version label is already assigned to a revision within the selected archive, select one of the following from the **If Version Label Exists** drop-down list:
    - **Prompt:** displays a prompt asking what you want to do if an identical version label exists
    - **Reassign:** reassigns the identical version label to the revision you are checking in
    - **Don't reassign:** cancels check-in
- 3 Click **OK**.
- 4 Your IDE's check-in dialog box may redisplay; click **OK**.
- The file is checked in.

## About Version Labels

- |                                                                                                                                                            |                                                                                                                                                                                                              |
|------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| What are version labels?                                                                                                                                   | Version labels are tags used to identify a specific revision within an archive. Typically, version labels are used to identify the revisions that make up a specific product release, such as "Beta Test 1". |
| Assign a version label to a specific revision when you want to distinguish the revision from other revisions within the same archive or group of archives. |                                                                                                                                                                                                              |
| Using version labels                                                                                                                                       | Use the Version Manager Options dialog box to perform version label functions such as assigning, renaming, reassigning, and deleting version labels.                                                         |



**NOTE** You can also assign version labels when checking in files. See ["About Checking In Files" on page 33](#).

## Assigning Version Labels

You can assign a version label to specific revisions in one or more archives, projects, or an entire project database.

**To assign a version label:**

- 1 Open the Version Manager Options dialog box. (For the IDE-specific menu command, see [Part 2, "IDE Reference," on page 55.](#))
- 2 Select the General tab and click the Assign button in the Manage Version Labels group. The Assign Version Label dialog box appears.
- 3 The default project database displays under **Select Project**. If you wish to work with a different project database, click the **Open Database** button. The Select Project Database dialog box appears.

Do one of the following:

- To open a project database located on a listed Version Manager File Server, select it from the **Project Databases** list and click **OK**.

To add a Version Manager File Server to the list, double-click an empty cell in the **File Server** list and type the name of the system that hosts the file server. Once you press the ENTER key, the entry will be auto-completed (`http://SystemName:8080/serenafs/FileServer`) and tested. To specify an https server or a non-standard port, double-click on the auto-completed entry, edit it appropriately, and press the ENTER key.

- To open a project database located on your local file system or the network, click the **Browse** button and navigate to the location via the resulting dialog box.
- 4 Under **Select Project**, select the project or project database that contains the files you want to label.
  - 5 To list all of the files in a project database or in subprojects below the selected project, select the **Include files in subprojects** check box.



**NOTE** You must use this check box if you selected a project or project database that contains no files at the root level.

- 6 Select the files you want to label from under **Files to Label**.



**TIP** To operate on all listed files, click OK without selecting any files. A prompt asks if you wish to operate on all files. Click Yes.

- 7 In the **Assign Version Label** field, enter the version label name, or click the Browse button to select one.
- 8 In the **To Revision** field, enter the number of the revision to assign the label to, or click the Browse button to select one. If you do not select a revision, the label is assigned by default to the tip revision of every file.
- 9 To allow the version label to move with the latest (tip) revision of each file, select the **Float label with tip** check box.
- 10 To specify what to do if an identical version label is already assigned to a revision in the selected archive, select one of the following from the **If Version Label Exists** drop-down list:
  - **Prompt:** displays a prompt asking what you want to do if an identical version label exists

- **Reassign:** reassigns the identical version label to the revision
- **Don't reassign:** does not assign a version label to the revision

11 Click **OK**.

## Renaming Version Labels

You can rename existing version labels in one or more archives, a project, or an entire project database.

### To rename a version label:

- 1 Open the Version Manager Options dialog box. (For the IDE-specific menu command, see [Part 2, "IDE Reference," on page 55](#).)
- 2 Select the General tab and click the **Rename** button in the Manage Version Labels group. The Rename Version Label dialog box appears.
- 3 The default project database displays under **Select Project**. If you wish to work with a different project database, click the **Open Database** button. The Select Project Database dialog box appears.

Do one of the following:

- To open a project database located on a listed Version Manager File Server, select it from the **Project Databases** list and click **OK**.

To add a Version Manager File Server to the list, double-click an empty cell in the **File Server** list and type the name of the system that hosts the file server. Once you press the ENTER key, the entry will be auto-completed (`http://SystemName:8080/serenafs/FileServer`) and tested. To specify an https server or a non-standard port, double-click on the auto-completed entry, edit it appropriately, and press the ENTER key.

- To open a project database located on your local file system or the network, click the **Browse** button and navigate to the location via the resulting dialog box.
- 4 Under **Select Project**, select the project or project database that contains the label you want to rename.
  - 5 To list all of the files in a project database or in subprojects below the selected project, select the **Include files in subprojects** check box.



**NOTE** You must use this check box if you selected a project or project database that contains no files at the root level.

- 6 Select the files with the label you want to rename from under **Files to Label**.



**TIP** To operate on all listed files, click **OK** without selecting any files. A prompt asks if you wish to operate on all files. Click **Yes**.

- 7 In the **Rename From** field, enter the version label that you want to rename, or click the Browse button to select one.
- 8 In the **To** field, enter the new version label, or click the Browse button to select one.

- 9 Click **OK**.

## Deleting Version Labels

You can delete version labels when they are no longer needed.

### To delete a version label:

- 1 Open the Version Manager Options dialog box. (For the IDE-specific menu command, see [Part 2, "IDE Reference," on page 55.](#))
- 2 Select the General tab and click the **Delete** button in the Manage Version Labels group. The Delete Version Label dialog box appears.
- 3 The default project database displays under **Select Project**. If you wish to work with a different project database, click the **Open Database** button. The Select Project Database dialog box appears.

Do one of the following:

- To open a project database located on a listed Version Manager File Server, select it from the **Project Databases** list and click **OK**.

To add a Version Manager File Server to the list, double-click an empty cell in the **File Server** list and type the name of the system that hosts the file server. Once you press the ENTER key, the entry will be auto-completed (`http://SystemName:8080/serenafs/FileServer`) and tested. To specify an https server or a non-standard port, double-click on the auto-completed entry, edit it appropriately, and press the ENTER key.

- To open a project database located on your local file system or the network, click the **Browse** button and navigate to the location via the resulting dialog box.
- 4 Under **Select Project**, select the project or project database that contains the label you want to delete.
  - 5 To list all of the files in a project database or in subprojects below the selected project, select the **Include files in subprojects** check box.



**NOTE** You must use this check box if you selected a project or project database that contains no files at the root level.

- 6 Select the files from which you want to delete a label from under **Files to Select**.



**TIP** To operate on all listed files, click OK without selecting any files. A prompt asks if you wish to operate on all files. Click Yes.

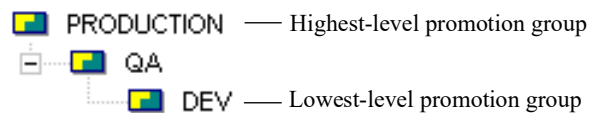
- 7 In the **Version Label** field, enter the version label name, or click the Browse button to select one.
- 8 Click **OK**.

## About Promotion Groups

**Before you begin...** Before you can begin working with promotion groups, you must have a promotion model set up for your project database. A *promotion model* is a hierarchy of milestones in a development cycle. These milestones are represented by promotion groups. For information on setting up a promotion model, see the *Administrator's Guide*.

**What is a promotion group?** A promotion group is a milestone within a promotion model hierarchy. When a promotion model is created for a project database, you assign revisions to the lowest-level promotion group. As development matures and reaches specified milestones, authorized users can promote revisions within the promotion model hierarchy. Development is considered complete when a revision reaches the highest-level promotion group, the top of the promotion model hierarchy.

The following figure is an example of a simple, but typical, promotion model.



**Why use promotion groups?** Promotion groups are meant for tracking development as a revision moves through the different milestones in your development process. They help you regulate software changes by making it necessary to associate locked revisions, which are the only revisions that can be edited, with the lowest-level promotion group. You can also use promotion groups to control access to source code by integrating a promotion model with various security options.

Promotion groups are useful in development environments where formal procedures are in place for moving software from one stage to the next, and where Project Team Members have different but well defined responsibilities and tasks.

**Promote permissions** Project Leaders or Administrators are typically the only users who have the authority to promote revisions. To be able to promote revisions, you must have promote permissions assigned to you by your Administrator.

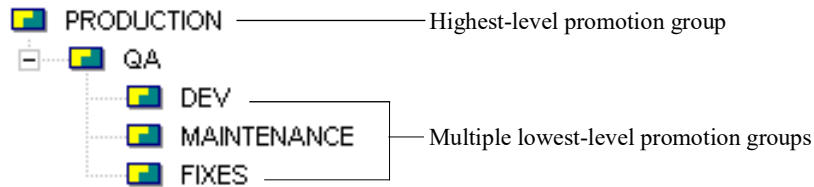
## Checking Out Revisions Assigned to a Promotion Group

An important rule to remember with promotion groups: you can only check out revisions to the lowest-level promotion group, the level reserved for development work. Regardless of what promotion level a revision has reached, when you check out and lock a revision, you must assign the revision to a lowest-level promotion group to continue development.

Workspace settings may be used to define a Default Promotion Group. If a Default Promotion Group is not defined for the active workspace, Version Manager will either:

- Prompt you to select a lowest-level promotion group if more than one is defined within a promotion model, or

- Use the lowest-level promotion group if only one is defined within a promotion model.



For a comprehensive list of promotion model rules, see the *Administrator's Guide*.

## Assigning a Promotion Group to Revisions

You can assign a promotion group to revisions by selecting a revision, a single versioned file, multiple versioned files, a project, a folder, or a project database.

### To assign a promotion group:

- 1 Open the Version Manager Options dialog box. (For the IDE-specific menu command, see [Part 2, "IDE Reference," on page 55](#).)
- 2 Select the General tab and click the **Assign** button in the Manage Promotion Groups group. The Assign Promotion Group dialog box appears.
- 3 The default project database displays under **Select Project**. If you wish to work with a different project database, click the **Open Database** button. The Select Project Database dialog box appears.

Do one of the following:

- To open a project database located on a listed Version Manager File Server, select it from the **Project Databases** list and click **OK**.

To add a Version Manager File Server to the list, double-click an empty cell in the **File Server** list and type the name of the system that hosts the file server. Once you press the ENTER key, the entry will be auto-completed (`http://SystemName:8080/serenafs/FileServer`) and tested. To specify an https server or a non-standard port, double-click on the auto-completed entry, edit it appropriately, and press the ENTER key.

- To open a project database located on your local file system or the network, click the **Browse** button and navigate to the location via the resulting dialog box.
- 4 Under **Select Project**, select the project or project database that contains the files you want to act on.
  - 5 To list all of the files in a project database or in subprojects below the selected project, select the **Include files in subprojects** check box.



**NOTE** You must use this check box if you selected a project or project database that contains no files at the root level.

- 6 Select the files you want to act on from under **Select Files**.



**TIP** To operate on all listed files, click OK without selecting any files. A prompt asks if you wish to operate on all files. Click Yes.

- 7 Specify a promotion group in the **Assign Promotion Group** field or browse to select one.
- 8 To act on a revision other than the default revision, enter it in the **To Revision** field or browse to select it.
- 9 To specify what to do if an identical promotion group is already assigned to a revision in the archive, select one of the following from the **If promotion group exists** drop-down list:
  - **Prompt:** Displays a prompt asking what you want to do if an identical promotion group exists.
  - **Reassign:** Reassigns the promotion group to the selected revision.
  - **Don't reassign:** Does not assign a promotion group to the revision.
- 10 Click **OK**.

## Promoting Revisions to the Next Promotion Group

You can promote revisions to the next promotion group by selecting a revision, a single versioned file, multiple versioned files, a project, a folder, or a project database.

### To promote revisions to the next promotion group:

- 1 Open the Version Manager Options dialog box. (For the IDE-specific menu command, see [Part 2, "IDE Reference," on page 55](#).)
- 2 Select the General tab and click the **Promote** button in the Manage Promotion Groups group. The Promote Promotion Group dialog box appears.
- 3 The default project database displays under **Select Project**. If you wish to work with a different project database, click the **Open Database** button. The Select Project Database dialog box appears.

Do one of the following:

- To open a project database located on a listed Version Manager File Server, select it from the **Project Databases** list and click **OK**.

To add a Version Manager File Server to the list, double-click an empty cell in the **File Server** list and type the name of the system that hosts the file server. Once you press the ENTER key, the entry will be auto-completed (`http://SystemName:8080/serenafs/FileServer`) and tested. To specify an https server or a non-standard port, double-click on the auto-completed entry, edit it appropriately, and press the ENTER key.

- To open a project database located on your local file system or the network, click the **Browse** button and navigate to the location via the resulting dialog box.
- 4 Under **Select Project**, select the project or project database that contains the files you want to act on.



- 5 To list all of the files in a project database or in subprojects below the selected project, select the **Include files in subprojects** check box.



**NOTE** You must use this check box if you selected a project or project database that contains no files at the root level.

- 6 Select the files you want to act on from under **Select Files**.



**TIP** To operate on all listed files, click OK without selecting any files. A prompt asks if you wish to operate on all files. Click Yes.

- 7 Specify a promotion group in the **Promote From** field or browse to select one.
- 8 To allow reassignment of a promotion group across branches, select the **OK to move across branches** check box. Otherwise, the promotion will fail if a revision on another branch is already assigned the next highest promotion group.
- 9 Click **OK**.

## Changing a Promotion Group

You can change promotion groups by selecting a revision, a single versioned file, multiple versioned files, a project, a folder, or a project database.



**NOTE** You should not use the **Change** option as a means of promoting revisions because it does not enforce the promotion model hierarchy assigned to your project database.

### To change a promotion group:

- 1 Open the Version Manager Options dialog box. (For the IDE-specific menu command, see [Part 2, "IDE Reference," on page 55](#).)
- 2 Select the General tab and click the **Change** button in the Manage Promotion Groups group. The Change Promotion Group dialog box appears.
- 3 The default project database displays under **Select Project**. If you wish to work with a different project database, click the **Open Database** button. The Select Project Database dialog box appears.

Do one of the following:

- To open a project database located on a listed Version Manager File Server, select it from the **Project Databases** list and click **OK**.

To add a Version Manager File Server to the list, double-click an empty cell in the **File Server** list and type the name of the system that hosts the file server. Once you press the ENTER key, the entry will be auto-completed (`http://SystemName:8080/serenafs/FileServer`) and tested. To specify an https server or a non-standard port, double-click on the auto-completed entry, edit it appropriately, and press the ENTER key.

- To open a project database located on your local file system or the network, click the **Browse** button and navigate to the location via the resulting dialog box.

- 4 Under **Select Project**, select the project or project database that contains the files you want to act on.
- 5 To list all of the files in a project database or in subprojects below the selected project, select the **Include files in subprojects** check box.



**NOTE** You must use this check box if you selected a project or project database that contains no files at the root level.

- 6 Select the files you want to act on from under **Select Files**.



**TIP** To operate on all listed files, click OK without selecting any files. A prompt asks if you wish to operate on all files. Click Yes.

- 7 Specify a promotion group in the **Change From** field or browse to select one.
- 8 Specify a promotion group to change to in the **To** field or browse to select one.
- 9 Click **OK**.

## Removing a Promotion Group

You can remove promotion groups by selecting a revision, a single versioned file, multiple versioned files, a project, a folder, or a project database.



**NOTE** Even if you remove a promotion group from a revision, as long as a promotion model is in effect for your project database, every time you check out a revision, it will be associated with the lowest-level promotion group.

### To remove a promotion group from a revision:

- 1 Open the Version Manager Options dialog box. (For the IDE-specific menu command, see [Part 2, "IDE Reference," on page 55](#).)
- 2 Select the General tab and click the **Remove** button in the Manage Promotion Groups group. The Remove Promotion Group dialog box appears.
- 3 The default project database displays under **Select Project**. If you wish to work with a different project database, click the **Open Database** button. The Select Project Database dialog box appears.

Do one of the following:

- To open a project database located on a listed Version Manager File Server, select it from the **Project Databases** list and click **OK**.

To add a Version Manager File Server to the list, double-click an empty cell in the **File Server** list and type the name of the system that hosts the file server. Once you press the ENTER key, the entry will be auto-completed (`http://SystemName:8080/serenafs/FileServer`) and tested. To specify an https server or a non-standard port, double-click on the auto-completed entry, edit it appropriately, and press the ENTER key.

- To open a project database located on your local file system or the network, click the **Browse** button and navigate to the location via the resulting dialog box.

- 4 Under **Select Project**, select the project or project database that contains the files you want to act on.
- 5 To list all of the files in a project database or in subprojects below the selected project, select the **Include files in subprojects** check box.



**NOTE** You must use this check box if you selected a project or project database that contains no files at the root level.

- 6 Select the files you want to act on from under **Select Files**.



**TIP** To operate on all listed files, click OK without selecting any files. A prompt asks if you wish to operate on all files. Click Yes.

- 7 Specify a promotion group in the **Promotion Group** field or browse to select one.
- 8 Click **OK**.



## Chapter 4

---

# Accessing Source Control Information

Introduction	46
About Properties	46
Monitoring Source Control Activity with Pulse	47
About History Reports	50
About Difference Reports	52

## Introduction

Purpose	<p>This chapter describes how to access four types of information about the files you have placed under source control:</p> <ul style="list-style-type: none"><li>■ The properties of archives and revisions</li><li>■ The source control activity of other team members and the results of your own activity</li><li>■ The history of source control activity in archives and revisions</li><li>■ The differences between two files, or two revisions, or a file and a revision</li></ul>
IDE-specific information	<p>Some IDEs do not support these features. For information on which features are supported by your IDE, see <a href="#">Part 2, "IDE Reference," on page 55</a>.</p>

## About Properties

You can review the source control properties of any file under source control.

### Reviewing Properties

**To review the source control properties of a file:**

- 1 Select the file that corresponds to the Version Manager archive you want information about.
- 2 Open the Version Manager Properties dialog box. (For the IDE-specific menu command, see [Part 2, "IDE Reference," on page 55](#).) The archive appears in the left pane of the dialog box.
- 3 Click the plus sign to expand the view of the archive. All revisions, version labels, and promotion groups appear in the expanded view.

Displaying tabs	<p>The content of the Version Manager Properties dialog box varies depending on what you select in the left pane. When the dialog box appears, the archive is selected by default, and the Versioned File tab is visible. When you select a revision in the archive, the Revision, Version Labels, and Promotion Groups tabs appear.</p>
Versioned files tab	<ul style="list-style-type: none"><li>■ Under the Versioned Files tab, the following information is provided:<ul style="list-style-type: none"><li>• The name of the file</li><li>• The name and location of the associated Version Manager archive</li><li>• The date the archive was created</li><li>• The user who created the archive</li><li>• The users who currently have locks on revisions in the archive</li><li>• The description entered when the file was added to the project</li></ul></li></ul>
Revision tab	<ul style="list-style-type: none"><li>■ Under the Revision tab, the following information is provided:<ul style="list-style-type: none"><li>• The revision number</li></ul></li></ul>

- The date and time the file was last modified for the specified revision
  - The user who checked in the revision
  - The user who currently has a lock on the revision
  - The change description that was entered when the revision was checked in
- Version Labels tab
- The Version Labels tab displays a list of all version labels applied to the selected revision.
- Promotion Groups tab
- The Promotion Groups tab displays a list of all promotion groups applied to the selected revision.

## Monitoring Source Control Activity with Pulse

**About Pulse** Pulse allows users who are logged into the same projects to monitor certain source control events. For example, Pulse notifies you when another user has added a new file or checked in changes to an existing file. If you are working with multiple Version Manager projects in multiple development environments, you can display activity in all open projects, or limit the display to activity in particular environments.

Pulse also displays results messages for all source control actions that you perform from your current development environment. For example, if you check out a file from within Visual Basic, Pulse displays the success or failure of that checkout.

### Configuring Pulse

Before you begin using Pulse, configure it in the Options dialog box.

#### To configure Pulse:

- 1 Open the Options dialog box. (For the IDE-specific menu command, see [Part 2, "IDE Reference," on page 55.](#))
- 2 Select the General tab. Pulse options appear in the Pulse group.

You can start and stop the display of source control activity in your IDE from the Options dialog box. See the next section for more information on starting and displaying Pulse.

- 3 Do any of the following:

- Select the frequency with which Pulse scans open projects for changes from the **Check for changes** drop-down list. By default, Pulse checks for changes every ten minutes. This option applies to project-wide activity monitoring only.



**NOTE** If you are working with very large Version Manager projects, frequent scanning for changes may impact performance.

- To automatically start Pulse when you launch the Version Manager IDE client, select the **Launch Pulse (r) on startup** check box.

- To display changes to all files in open Version Manager projects and subprojects, select the **Report on files in subprojects** check box. If this option is not selected, Pulse displays changes to files in the root Version Manager project only.



**NOTE** If you change the **Report on files in subprojects** check box, you must restart your IDE for the change to take effect.

## About Starting Pulse

Automatic startup	Pulse begins monitoring source control activity as soon as it is launched. You can configure Pulse to start when ever you launch the Version Manager IDE client. See <a href="#">"Configuring Pulse" on page 47.</a>
One IDE	If Pulse is not running or you have stopped monitoring the source control activity of your IDE, you can start or resume monitoring from the Options dialog box.
Multiple IDEs	To monitor activity in multiple IDEs running on your system, start Pulse from within each IDE. Pulse displays the activity of only the IDEs from which it was started. For example, if you are working with Version Manager projects in both Visual Basic and Visual C++ but start Pulse from only Visual Basic, Pulse will display activity in Visual Basic projects only.





**NOTE** If you have suspended monitoring of projects across all IDEs, clicking the **Start Monitoring Project Files in this IDE** button has no effect. You must un-suspend activity monitoring to resume display. See ["Suspending Project Activity Monitoring" on page 50.](#)

### Starting Pulse

#### To start Pulse:

- 1 Open the Options dialog box. (For the IDE-specific menu command, see [Part 2, "IDE Reference," on page 55.](#))
- 2 Select the General tab.
- 3 In the Pulse group, click the **Start Monitoring Project Files in this IDE** button.

### Displaying Pulse

When you start Pulse, an icon (  ) appears in the taskbar status area of your desktop. The icon changes (  ) to notify you when source control activity is detected. If the Pulse window is not visible when the icon changes, double-click the icon. The window appears.

### Hiding Pulse

To hide the Pulse window, click the Close button. Closing the Pulse window does not suspend monitoring. To re-display the window, double-click the icon.



## Viewing Source Control Activity




Pulse allows you to monitor two types of source control activity:

- Project-wide
  - Activity in all Version Manager projects open in IDEs from which Pulse is running. This includes actions performed from other users' workstations and from other development environments currently running on your system.
- Local
  - Actions you perform from within your current development environment.

### Project-wide Activity

**Project activity tab** The Project Activity tab displays information about activity originating from instances of development environments other than the one that is currently active on your system. For example, if you are sharing a Version Manager project with a user at another workstation, and that user adds a new file to the project, Pulse notifies you. You can also monitor activity across multiple instances of the same environment on the same system.

Icons in the File column identify each event as one of the following:

This icon...	Displays when this event occurs...
	A new file is added to source control
	A file is removed from source control
	A new revision of a file is checked in as the tip, or latest, revision

- Project-wide activity information** For every event that occurs, the Project Activity tab displays the following information:
- **File:** The name of the file that has been added, removed, or modified. Click the File column header to alphabetically sort events by filename.
  - **Project:** The Version Manager project that is affected by the activity. Click the Project column header to alphabetically sort events by project name.
  - **Date/Time:** The date and time that the event was detected. Click the Date/Time column header to sort events by date.
  - **Description:** The event that occurred. When a new revision is added, this column displays both the old revision number and the new revision number. Click the Description column header to alphabetically sort events by description.

**Detailed activity information** You can also view more detailed information about each event, such as the location of a file within a Version Manager project, and the development environment from which a file was added, removed, or modified.

To display detailed information about a source control event, double-click the event under the Project Activity tab. The Project Activity Details window appears.

### Local Results Messages

**Output tab** Under the Output tab, Pulse displays results messages for all source control actions that you perform from the IDE currently active on your system.

## Suspending Project Activity Monitoring

You can stop monitoring project-wide source control activity in specific IDEs, or suspend monitoring in all open development environments.



**NOTE** Suspending the display of project activity does not affect the display of source control results messages under the Pulse Output tab.

### Specific IDEs

#### To stop monitoring in a specific IDE:



- 1 Open the Options dialog box from within the IDE. (For the IDE-specific menu command, see [Part 2, "IDE Reference," on page 55.](#))
- 2 Select the General tab.
- 3 In the Pulse group, click the **Stop/Start Monitoring Project Files in this IDE** button.

Resuming  
monitoring

You can resume monitoring by clicking the **Stop/Start Monitoring Project Files in this IDE** button.

### All Development Environments

#### To suspend monitoring in all open IDEs:

- 1 If the Pulse window is not currently visible, double-click the Pulse icon (  ) in the Taskbar Status area of your desktop.
- 2 Select the Project Activity tab.
- 3 Select the **Suspend monitoring of projects** check box. The icon in the Taskbar Status area of your desktop reflects the suspended status (  ).

Resuming  
monitoring

You can resume monitoring by deselecting the **Suspend monitoring of projects** check box.



**TIP** You can also suspend or resume project monitoring by right-clicking the Pulse taskbar icon and then selecting Suspend from the pop-up menu.

## Closing Pulse

Pulse remains active when you close your IDE. If you want to close Pulse, right-click the Pulse taskbar icon and select Exit from the pop-up menu.

## About History Reports

History reports summarize information about archives and/or revisions that you can use to monitor the development process, review archive histories, and check archive

attributes. For more information about the content of history reports, see the *Version Manager User's Guide*.

## Generating History Reports

### To generate a history report:

- 1 Select a file to be included in the report.
- 2 Open the Show History dialog box. (For the IDE-specific menu command, see [Part 2, "IDE Reference,"](#) on page 55.)



**NOTE** Fields shown in the Show History dialog box vary depending on which report type is selected.

#### General tab options

- 3 Under the General tab, select a report type in the **Report Type** drop-down list:

Full	Comprehensive information (including file, revision, lock, and version label). To select specific revisions, enter the revision number, version label, or promotion group in the <b>Revision</b> field or click the Browse button to select a revision.
File information only	Only archive information (such as creation date, owner, locks, version labels). This report does not contain revision history.
Revision information only	Only revision information on all or selected revisions. To select specific revisions, enter the revision numbers (separated by commas) in the <b>Revision</b> field.
List locked revisions	A list of locked revisions within the selected file(s).
List revisions with version label	A list of revisions that match a specific version label. To select a version label, enter it in the <b>Label</b> field, or click the Browse button to select one.
List revisions in group	A list of revisions that match a specific promotion group. To select a promotion group, enter the promotion group in the <b>Group</b> field, or click the Browse button to select one.
List newest revisions	A list of the latest revisions (if you selected multiple files).
Check tips against version/revision	Information that compares the latest revision with a specified revision. To specify the revision, enter a revision number, version label, or promotion group in the <b>Revision</b> field, or click the Browse button to select one.

#### Advanced tab options

- 4 Under the Advanced tab, you can restrict report information by generating a report based on author, user locks, owners, date range, or a combination of these options.

#### Author(s)

- To generate a report based on a specific author, enter the author(s) name in the **Author(s)** field, or click the Browse button to select an author. Separate multiple authors with a comma (,).

- |            |                                                                                                                                                                                                                                                                             |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Locker(s)  | ■ To generate a report based on user locks, enter the user names in the <b>Locker(s)</b> field, or click the Browse button to select a user. Separate multiple users with a comma (,).                                                                                      |
| Owner(s)   | ■ To generate a report based on a specific owner, enter the owner names in the <b>Owner(s)</b> field, or click the Browse button to select an owner. Separate multiple owners with a comma (,).                                                                             |
| Date Range | ■ To generate a report based on a date range, select the <b>Date Range</b> check box, then specify dates and times in the <b>From</b> and <b>To</b> fields.                                                                                                                 |
| Files tab  | <b>5</b> Under the Files tab, review the files for which the history report will be generated.<br>The Files tab displays a list of all of the files you selected with a check beside each one. You can change your selection by checking and unchecking files in this list. |
|            | <b>6</b> When you have chosen the options you want, click <b>OK</b> . The history report will appear. (If you are generating a report on a large number of files, it may take a few moments.)                                                                               |

## About Difference Reports

Version Manager launches a separate utility, the Merge Tool, to compare files and generate difference reports. You can compare:

- A revision and a workfile
- Revisions in a single archive
- Revisions in two different archives
- Two workfiles

You can use difference reports to:

- Identify specific differences between files or revisions.
- Determine what changes have been made to a file before you check it in.
- Confirm which revision is newer if you are unsure of a timestamp.

## Generating Difference Reports


**To generate a difference report:**

- 1** Select a file to be included in the report.
- 2** Open the Show Differences dialog box. (For the IDE-specific menu command, see [Part 2, "IDE Reference," on page 55.](#))

The appearance of the Show Differences dialog box varies depending on the files you select and the type of comparison you want.

- 3** Select the type of comparison you want to perform from the **Compare** drop-down list.

If you select this report type...	You must choose...
A revision and a workfile	A versioned file and revision number, and a workfile
Revisions in a single versioned file	A versioned file and revision number, and a second revision number
Revisions in two different versioned files	Two versioned files and two revision numbers
Two workfiles	Two workfiles

- 4** To change the selected files or revisions in the First File and Second File groups, click the Browse buttons.
- 5** Select the **Ignore white space** check box if you want to ignore trailing, intervening, and leading white spaces, tabs, and form feeds.
- 6** Click **OK**. The Merge Tool is launched in a separate window.
- 7** To view the differences, scroll through the files to compare the colored text blocks. You can also click the **Next Difference** button (  ) from the Merge Tool window to go directly to the differences.

For information on advanced differencing features and interpreting difference reports, see the *User's Guide*.



## Part 2

---

# IDE Reference

ColdFusion Studio	57
PowerBuilder	65
Rational Application Developer (Eclipse 3 and 4)	77
Rational Application Developer Rich Integration (Eclipse 3 and 4)	95
Rational Rose	141
Visual Studio SCC Integration	139
Visual Studio Rich Integration	149

# Introduction

**Contents and purpose** This part of the manual contains chapters specific to setting up and using the Version Manager IDE client with supported IDEs. The purpose of this part of the manual is to help you set up and use the Version Manager IDE client with your IDE.

**Additional information** Use this part of the manual in conjunction with these additional sources of information:

For more information about...	See...
Source control concepts	<a href="#">Chapter 1, "Overview of Version Manager Source Control" on page 11</a>
Setting up and configuring the Version Manager IDE client to work with SCC compliant IDEs	<a href="#">Chapter 2, "Setting Up Source Control with SCC IDEs" on page 17</a>
Setting up and configuring the Version Manager IDE client to work with web-based IDE projects	<a href="#">Chapter 3, "Setting Up Source Control with COM IDEs" on page 35</a>
Default and advanced options	<a href="#">Chapter 3, "Using Source Control" on page 27</a>
Viewing information about items under source control	<a href="#">Chapter 4, "Accessing Source Control Information" on page 45</a>
Setting up and using your IDE with source control	The documentation provided by the vendor of your IDE



## Chapter 5

---

# ColdFusion Studio

Introduction	58
Accessing Supported Features	58
Setting Up Source Control Projects	59
Using Source Control	62

# Introduction

**Purpose** This chapter has four purposes:

- List the Version Manager features available through Adobe® ColdFusion® Studio and provide a quick reference to accessing those features
- Note any features described in Part 1 of this manual that do not apply to this IDE
- Help administrators set up source control projects and add files to source control
- Help your development team access files that are under source control from within ColdFusion Studio

**For more information** See [Part 1, "The Version Manager IDE Client," on page 9](#) for information about:

- Source control concepts
- Source control defaults
- Advanced source control features

## Accessing Supported Features

**What is supported?** ColdFusion Studio supports a subset of the source control features available through the Version Manager IDE client. Limitations specific to this IDE include the inability to:

- View archive properties
- Use advanced options
- Share archives across projects

**Accessing features** Access the menu commands listed below in one of two ways, depending on the type of file you are operating upon:

- To work with a ColdFusion Studio project file, right-click the project icon in the Project pane of the Projects tab.
- To work with ColdFusion Studio workfiles, select the files in the File pane of the Projects tab and right-click.

To...	Select...	For more information see...
Get revisions	Source Control   Get Latest Version	<a href="#">"Getting Files" on page 63</a>
Check out revisions	Source Control   Check Out	<a href="#">"Checking Out Files" on page 63</a>
Undo checkout of revisions	Source Control   Undo Check Out	<a href="#">"Undoing Checkout" on page 63</a>
Check in revisions	Source Control   Check In	<a href="#">"Checking In Files" on page 63</a>
Manage version labels	(Use the Version Manager desktop client)	<i>The Version Manager User's Guide</i>
View properties of revisions or archives	(Use the Version Manager desktop client)	<i>The Version Manager User's Guide</i>
Monitor source control activity	Source Control   Run Source Control Application	<a href="#">"Monitoring Source Control Activity with Pulse" on page 47</a>

To...	Select...	For more information see...
Generate a history report	Source Control   Show History	<a href="#">"About History Reports" on page 50</a>
Generate a difference report	Source Control   Show Differences	<a href="#">"About Difference Reports" on page 52</a>
Access the Version Manager Options dialog	Source Control   Run Source Control Application	<a href="#">"About Setting Defaults for Version Manager Options" on page 22</a>
Map (add) a project file to source control	Source Control   Map Project to Source Control	<a href="#">"Mapping Projects to Source Control" on page 60</a>
Remove a project file from source control	Source Control   Remove Project File from Source Control	<a href="#">"Removing Files from Source Control" on page 62</a>
Add workfiles to source control	Source Control   Add File to Source Control	<a href="#">"Adding Files to Source Control" on page 61</a>
Remove workfiles from source control	Source Control   Remove File from Source Control	<a href="#">"Removing Files from Source Control" on page 62</a>

## Setting Up Source Control Projects

Contents	This section contains information about setting up the Version Manager IDE Client to work with ColdFusion Studio.
Prerequisites	Before proceeding, you must use the Version Manager desktop client to create a project database that will contain the source control projects associated with the ColdFusion Studio project (if you don't already have one).
For more information	See <a href="#">Chapter 2, "Setting Up Source Control with SCC IDEs" on page 17</a> .

### Setting Up Projects for Access by Multiple-Users

If multiple users will access the project, you must configure it with this in mind.

#### To connect multiple users to a project:

- 1 Create the ColdFusion project in a location accessible to all users.
- 2 Create the Version Manager project database in a location accessible to all users.
- 3 Add the ColdFusion project to source control. (See the following sections for details.)
- 4 Using the ColdFusion IDE, open the project from each development system.



#### NOTE

- Use a unique user ID to log in to each development system. If everyone uses the same O/S login, anyone can check in a file you have checked out.
- After adding a new file to the project, check in the ColdFusion project file (.apf). Each user must then reopen the project in order to see the new file.

## Selecting a Source Control Provider

ColdFusion Studio allows you to select a source control provider for each project individually.

**To select a source control provider:**

- 1 From the Projects tab, right-click the project icon in the Projects pane and select Source Control | Choose Source Control Provider. The Choose Source Control Provider dialog box appears.
- 2 Select **Version Manager** from the **Providers** list.
- 3 Click **OK**.

## Mapping Projects to Source Control

Once you have created and saved a ColdFusion project, you can map (add) it to an existing Version Manager project database.



### IMPORTANT!

- Each ColdFusion project must reside in its own directory. If there are multiple ColdFusion projects in a single directory and you place them under source control, source control will fail.
- All files in a ColdFusion project must be located under the root project working directory--the directory containing the ColdFusion project file (.apf). Any files outside of the project directory structure will not be added to source control.

To map and add a project to source control:

- 1 From the Projects tab, right-click the project icon in the Projects pane and select Source Control | Map Project to Source Control. The Add Project to Source Control dialog box appears.
- 2 The default project database displays under **Source Control Project**. If you wish to add the files to a different project database, click the **Open Database** button. The Select Project Database dialog box appears.

Do one of the following:

- To open a project database located on a listed Version Manager File Server, select it from the **Project Databases** list and click **OK**.

To add a Version Manager File Server to the list, double-click an empty cell in the **File Server** list and type the name of the system that hosts the file server. Once you press the ENTER key, the entry will be auto-completed (`http://SystemName:8080/serenafs/FileServer`) and tested. To specify an https server or a non-standard port, double-click on the auto-completed entry, edit it appropriately, and press the ENTER key.

- To open a project database located on your local file system or the network, click the **Browse** button and navigate to the location via the resulting dialog box.
- 3 Do one of the following:
    - To add to an existing source control project, select one from under **Source Control Project**. Proceed to [Step 4](#).

■ **To create a new Version Manager project:**

**a** Under Select Source Control Project, select the location in the project database where you want to create the new project.

**b** Click the **Create Project** button. The Create Source Control Project dialog box appears.

The Project Database Information group displays the name and location of the current project database, and the location of the new project within the database.

**c** By default, the new Version Manager project uses the same name as the IDE project. If necessary, enter a different name in the **Project Name** field.

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 (< >).

**d** Click **OK**. The Add Project to Source Control dialog box reappears with the new project displayed under **Source Control Project**.

**4** Click **OK**. The project file is *mapped* to source control. Continue to the next step to *add* the project file to source control.

**5** From the Projects tab, right-click the project icon in the Projects pane and select Source Control | Add Project File to Source Control. The Comment dialog box appears.

**6** In the **Comment** field, enter a description of the project file.

**7** Click **OK**.

Continue to the next section to add the files in the project to source control.



**IMPORTANT!** For each project that is mapped to source control, Version Manager creates a file named `projectname.cache`. This file is located in the working directory of the ColdFusion project. If this file is lost, you must remap the ColdFusion project to restore the project to source control.

## Adding Files to Source Control



**NOTE** You must add the project (.apf) file to source control before you can add other files to the source control project. See ["Mapping Projects to Source Control" on page 60](#).

**To add a file to a source control project:**

**1** Check out the project (.apf) file by right-clicking the project icon in the Project pane of the Projects tab and selecting Source Control | Check Out.

**2** Do one of the following:


- If you are adding a file to the ColdFusion Studio project, a prompt displays asking if you want to add the file to source control. Click **Yes**.
- If the file is already part of the ColdFusion project but you have not yet added it to source control, right-click the file icon in the File pane of the Projects tab and select Source Control | Add File to Source Control.

The Comment dialog box appears.

- 3 In the **Comment** field, enter a description of the file. The description will be applied to all selected files.



**TIP** To use unique descriptions for each file, leave the Comment field blank and click OK. The Change Description dialog box will appear for each file in turn.

- 4 Click **OK**. A red dot (  ) appears next to the file icon in the File pane to indicate that the file is under source control.

## Removing Files from Source Control

When you remove a file from source control, Version Manager does *not* delete the workfile or Version Manager archive; it simply removes the association between the IDE file and archive. You can always add the file back to source control later.

Remove workfile

**To remove a ColdFusion workfile file from source control:**

- 1 In the File pane of the Projects tab, select the file(s) you wish to remove from source control.
- 2 Right-click and select Source Control | Remove File from Source Control. A confirmation message appears.
- 3 Click **Yes**. The selected files are removed from source control.

Remove project

**To remove a ColdFusion project (.apf) file from source control:**

- 1 In the Project pane of the Projects tab, select the project you wish to remove from source control.
- 2 Right-click and select Source Control | Remove Project File from Source Control. A confirmation message appears.
- 3 Click **Yes**. The selected project file is removed from source control.

Returning files to source control

For information on returning files to source control, see ["Returning Files to Source Control" on page 24](#).

## Using Source Control

Contents

This section contains procedural information about viewing and editing files that are under source control.

For more information

See [Chapter 3, "Using Source Control" on page 27](#).



**TIP** To speed your work, set the IDE and Version Manager defaults to reflect the way you usually work.

## Getting Files

When you get files, read-only copies of the latest revisions are placed in the workfile location.


### To get a revision:

- 1 In the File pane of the Projects tab, select the file(s) and right-click.
- 2 Select Source Control | Get Latest Version.

## Checking Out Files

When you check out a file, the latest (tip) revision is locked and a writable workfile is created in the workfile location.


### To check out a file:

- 1 In the File pane of the Projects tab, select the file(s) and right-click.
- 2 Select Source Control | Check Out. A green check mark (✓ ) appears next to the file icon in the File pane.

## Undoing Checkout

When you undo a checkout, the archive is unlocked and a read-only workfile is left in the workfile location. No changes are checked into the archive.

### To undo a checkout:

- 1 In the File pane of the Projects tab, select the file(s) and right-click.
- 2 Select Source Control | Undo Check Out. The green check mark (✓ ) is removed from the file icon in the File pane to indicate that the file is unlocked.

## Checking In Files

By default, the following occurs when you check in a file:

- A new revision is created and assigned the next number in sequence.
- A read-only workfile is left in the workfile location.
- The archive is unlocked.


### To check in a file:

- 1 Save any changes to the ColdFusion files you wish to check in.
- 2 In the File pane of the Projects tab, select the file(s) and right-click.
- 3 Select Source Control | Check In. The Comment dialog box appears.

- 4 Enter a description of the changes you made in the **Comment** field. This description will be applied to all selected files.



**TIP** To apply a unique description to each file, deselect the **Apply To All Files** check box.

- 5 Click **OK**. The green checkmark (  ) is removed from the file icon in the File pane.



## Chapter 6

---

# PowerBuilder

Introduction	66
About Version Manager Project Structure	66
Accessing Supported Features in PowerBuilder	67
Setting Up Source Control Projects in PowerBuilder	68
Using Source Control with PowerBuilder	73

# Introduction

Purpose	<p>This chapter has four purposes:</p> <ul style="list-style-type: none"><li>■ List the Version Manager features available through Sybase® PowerBuilder™, and provide a quick reference to accessing those features</li><li>■ Note any features described in Part 1 of this manual that do not apply to this IDE</li><li>■ Help administrators set up source control projects and add files to source control</li><li>■ Help your development team access files that are under source control from within PowerBuilder</li></ul>
For more information	<p>See <a href="#">Part 1, "The Version Manager IDE Client," on page 9</a> for information about:</p> <ul style="list-style-type: none"><li>■ Source control concepts</li><li>■ Source control defaults</li><li>■ Advanced source control features</li></ul>

## About Version Manager Project Structure

Default project structure	<p>By default, when you add a project to source control, Version Manager creates a hierarchical project structure that mirrors the physical location of the files in your working directories. Because PowerBuilder stores objects in libraries rather than directories, Version Manager creates versioned files for all objects directly under the root project, rather than hierarchically based on application structure.</p>
Example	<p>If your application <i>App1</i> contains three objects, each in separate PBLs, all three objects will appear in one Version Manager project:</p> <div><div><p><b>PowerBuilderApplication</b></p><pre>App1 ├── PBL1 │   └── Object1 ├── PBL2 │   └── Object2 └── PBL3     └── Object3</pre></div><div><p><b>Version Manager Project</b></p><pre>App1 ├── Object1 ├── Object2 └── Object3</pre></div></div>
Archive organization	<p>Version Manager organizes archives based on project structure. In the example above, Version Manager would place the archives associated with the three objects under the <i>App1</i> archive directory.</p>
Mirroring PBL structure	<p>If you want to organize your versioned files into subprojects based on the applications PBL structure, we recommend that you use version labels. When you add objects to source control, you can group objects by assigning version labels that correspond to PBL names.</p>

With the Version Manager desktop client, you can select versioned files based on version labels and copy them into subprojects.



**NOTE** PowerBuilder allows you to create each target in its own subdirectory beneath the workspace directory, but it does not do so by default. We recommend that you use this feature, as Version Manager will then automatically create subprojects for each target.

### Using Version Labels to Organize Projects

You can use version labels to selectively copy your versioned files into multiple projects that mirror your application's library structure.

**To organize projects based on PBLs, complete the following steps:**

- 1** As you add objects to source control, assign version labels that correspond to the names of the libraries the objects belong to. To be sure that the labels are always easily visible, set the labels to *float* to the tip. Floating version labels are always associated with the latest revision in an archive.
- 2** Once all of the objects in the application have been added to source control, open the new project in the Version Manager desktop client.
- 3** Create the subprojects into which you will copy the versioned files. For example, create subprojects that correspond to each PBL in your application (PBL1, PBL2, PBL3 and so forth).
- 4** In the Project pane, select the Version Manager project you created from within PowerBuilder. All versioned files in the project appear in the File pane.
- 5** Use the Version Label Filter dialog box to customize the File pane to display only versioned files to which the specified label is assigned. For example, if you assigned the label PBL1 to all objects in PBL1, filter the File pane to display only versioned files containing the label PBL1.
- 6** Copy the selected versioned files from the file pane to the subproject that corresponds to the PBL the objects belong to.
- 7** Repeat steps 4-6 for each PBL in the application.

For information on creating projects, filtering the File pane, and copying versioned files, see the *User's Guide*.

## Accessing Supported Features in PowerBuilder

What is supported?

PowerBuilder supports the full set of source control features available through the Version Manager IDE client, except for the sharing of archives among projects. See the following table.



**NOTE** The Entry menu is available only if the Library Painter window is open.

To...	Select...	For more information see...
Get revisions	Right-click   Get Latest Version	<a href="#">"Getting Objects" on page 73</a>
Check out revisions	Right-click   Check Out	<a href="#">"Checking Out Objects" on page 73</a>
Undo checkout of revisions	Right-click   Undo Check Out	<a href="#">"Undoing Checkout" on page 74</a>
Check in revisions	Right-click   Check In	<a href="#">"Checking In Objects" on page 74</a>
Manage version labels	Entry   Source Control   Advanced Options	<a href="#">"About Version Labels" on page 34</a>
View properties of revisions or archives	Entry   Source Control   Source Control Properties	<a href="#">"About Properties" on page 46</a>
Monitor source control activity	Entry   Source Control   Advanced Options	<a href="#">"Monitoring Source Control Activity with Pulse" on page 47</a>
Generate a history report	Entry   Source Control   Show History	<a href="#">"About History Reports" on page 50</a>
Generate a difference report	Entry   Source Control   Show Differences	<a href="#">"About Difference Reports" on page 52</a>
Access the Version Manager Options dialog	Entry   Source Control   Advanced Options	<a href="#">"About Setting Defaults for Version Manager Options" on page 22</a>
Connect workspaces to source control	Right-click   Properties	<a href="#">"Setting Up Source Control Projects in PowerBuilder" on page 68</a>
Add objects to source control	Right-click   Add to Source Control	<a href="#">"Adding Objects to Source Control" on page 70</a>
Disconnect workspaces from source control	Right-click   Properties	<a href="#">"Disconnecting Workspaces from Source Control" on page 72</a>
Remove objects from source control	Entry   Source Control   Remove from Source Control	<a href="#">"Removing Objects from Source Control" on page 72</a>

## Setting Up Source Control Projects in PowerBuilder

For best results, you should create a hierarchy of directories so each target is located in its own directory beneath the workspace directory. Version Manager will then create nested source control projects that reflect the structure of your workspace. This avoids problems associated with having identically named files in several targets and imposes a logical hierarchy on the otherwise flat file structure of PowerBuilder projects.

### Connecting PowerBuilder Workspaces to Source Control


**To connect a PowerBuilder Workspace to source control:**

- 1 Right-click on the Workspace object in the System Tree pane. A pop-up menu appears.
- 2 Select Properties. The Properties of Workspace dialog box opens to the Source Control tab.

- 3 Select **Version Manager** from the **Source Control System** drop-down menu.
  - 4 Click the browse button to the right of the **Project** field. The Add Project to Source Control dialog box appears.
- Select a project database
- 5 The default project database displays under **Source Control Project**. If you wish to add the files to a different project database, click the **Open Database** button. The Select Project Database dialog box appears.
- Do one of the following:
- To open a project database located on a listed Version Manager File Server, select it from the **Project Databases** list and click **OK**.  
To add a Version Manager File Server to the list, double-click an empty cell in the **File Server** list and type the name of the system that hosts the file server. Once you press the ENTER key, the entry will be auto-completed (http://SystemName:8080/serenafs/FileServer) and tested. To specify an https server or a non-standard port, double-click on the auto-completed entry, edit it appropriately, and press the ENTER key.
  - To open a project database located on your local file system or the network, click the **Browse** button and navigate to the location via the resulting dialog box.
- Select a project
- 6 Do one of the following:
    - To connect to an existing source control project, select one from under **Source Control Project**. Proceed to [Step 7](#).
    - **To create a new Version Manager project:**
      - a In the project database tree, select the location where you want to create the new project.
      - b Click the **Create Project** button. The Create Source Control Project dialog box appears.  
The Project Database Information group displays the name and location of the current project database, and the location of the new project within the database.
      - c Enter a name for the source control project in the **Project Name** field.  
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 (<>).
      - d Click **OK**. The Add Project to Source Control dialog box reappears with the new project displayed under **Source Control Project**.
  - 7 Click **OK**. The Properties of Workspace dialog box reappears.
  - 8 Enter the path to the root directory for this workspace in the **Local Root Directory** field or click the browse button to select it.



**IMPORTANT!** All objects that are part of the PowerBuilder workspace must be located in the local root directory or in a subdirectory beneath it.

- 9 Optionally, complete the other fields on the Properties of Workspace dialog box to configure how PowerBuilder interacts with source control. For more information on how to use these settings, see the PowerBuilder documentation.
- 10 Click **OK**. A green plus sign (  ) appears to the left of each object icon to indicate that the workspace is connected to source control but the objects are not yet under source control.

## Adding Objects to Source Control

### To add objects to source control:


- 1 Right-click on the Workspace object in the System Tree pane. A pop-up menu appears.
- 2 Select Add to Source Control. The Add to Source Control dialog box appears.
- 3 Do any of the following:
  - Change which objects to add to source control by unchecking or checking the checkboxes next to each object name.
  - Enter a comment in the **Comment** field.
- 4 Click **OK**. The Advanced Add dialog box appears.
- 5 Do any of the following:
  - Under the Advanced tab, enter a version label in the **Version Label** field, or click the browse button to select one.

If you plan to use the Version Manager desktop client to re-organize your versioned files into individual projects based on your application's library structure, assign a version label that corresponds to the library each object belongs to. For example, if you are adding objects from a library called PBL1 to source control, enter *PBL1* in the **Version Label** field. See ["Using Version Labels to Organize Projects" on page 67](#).



**TIP** If you create each target in its own subdirectory beneath the workspace directory, the Version Manager IDE client will automatically create subprojects based on that directory structure.

- To cause the version label to always be associated with the latest revision in the archive, select the **Float label with tip** check box.
  - Under the General tab, enter a lowest-level promotion group to associate with the initial revisions of the files, or browse to select one.
- 6 Click **OK**. If the objects you are adding to source control do not already contain workfile descriptions, the Change Description dialog box appears.
  - 7 Enter a description for the current object in the **Description** field.
  - 8 Click **OK**. Complete the Advanced Add and Change Description dialog boxes for each target you are adding to source control.

A green dot (  ) appears to the left of each object icon to indicate that that object is under source control and is not checked out.

## Configuring Workstations in a Multi-User Environment

After connecting a PowerBuilder workspace to source control and adding the objects within it to source control, you can make it available to multiple users. To do so, you must complete the following steps:

- 1 Copy the directory structure of the PowerBuilder workspace to the workstation. Include only the workspace and target directories, the PowerBuilder Library (.PBL) files, and the PowerBuilder Target (.PBT) files.



### IMPORTANT!

- Do not include the workspace (.PBW) file. The .PBW file contains absolute paths.
- The directory structure of the PowerBuilder workspace must be the same on each system.

- 2 Create a new PowerBuilder workspace in the workspace directory you copied to the workstation.
- 3 Add the copied target (.PBT) files to the new workspace.
- 4 Connect the new workspace to the existing source control project. See the next section for more information on this step.

### Connecting Workstations to the Existing Source Control Project

#### To connect a workstation to source control:


- 1 Right-click on the Workspace object in the System Tree pane. A pop-up menu appears.
- 2 Select Properties. The Properties of Workspace dialog box opens to the Source Control tab.
- 3 Select **Version Manager** from the **Source Control System** drop-down menu.
- 4 Click the browse button to the right of the **Project** field. The Add Project to Source Control dialog box appears.
- 5 The default project database displays under **Source Control Project**. If you wish to add the files to a different project database, click the **Open Database** button. The Select Project Database dialog box appears.

Select a project database

Do one of the following:

- To open a project database located on a listed Version Manager File Server, select it from the **Project Databases** list and click **OK**.

To add a Version Manager File Server to the list, double-click an empty cell in the **File Server** list and type the name of the system that hosts the file server. Once you press the ENTER key, the entry will be auto-completed (`http://SystemName:8080/serenafs/FileServer`) and tested. To specify an https server or a non-standard port, double-click on the auto-completed entry, edit it appropriately, and press the ENTER key.

- To open a project database located on your local file system or the network, click the **Browse** button and navigate to the location via the resulting dialog box.
- Select a project
- 6 From under **Source Control Project**, select the source control project to which to connect the workspace.
  - 7 Click **OK**. The Properties of Workspace dialog box reappears.
  - 8 Enter the path to the root directory for this workspace in the **Local Root Directory** field or click the browse button to select it.
  - 9 Optionally, configure how PowerBuilder interacts with source control. See the PowerBuilder documentation for more information on how to use these settings.
  - 10 Click **OK**. Symbols appear to the left of each object icon to indicate that the objects are under source control:
    - A green dot (  ) indicates that the object is checked in.
    - A red X indicates that the object is checked out by another user.

## Removing Objects from Source Control

When you remove an object from source control, Version Manager does *not* delete the workfile or Version Manager archive; it simply removes the association between the object and archive. You can always add the object back to source control later.

### To remove objects from source control:

- 1 In the Library Painter, select the object or objects you wish to remove from source control.
- 2 Select Entry | Source Control | Remove from Source Control. The Remove from Source Control dialog box appears with a list of objects to remove from source control.
- 3 Make sure the correct objects are selected and click **OK**. The selected objects are removed from source control.

Returning files to source control For information on returning files to source control, see ["Returning Files to Source Control" on page 24](#).



**NOTE** To re-add objects to source control that you have previously removed from source control, use the Workspace tree. PowerBuilder may close unexpectedly if you initiate the add operation from the Library Painter.

## Disconnecting Workspaces from Source Control

Disconnecting a PowerBuilder workspace from source control does not affect the Version Manager project. You can always re-connect the workspace to the Version Manager project later.

### To disconnect a workspace from source control:

- 1 Right-click the workspace object in the System Tree. A pop-up menu appears.
- 2 Select Properties. The Properties of Workspace dialog box appears.



- 3 Select **None** from the **Source Control System** field.
- 4 Click **OK**. The workspace is disconnected from source control.

## Using Source Control with PowerBuilder

Contents This section contains procedural information about viewing and editing files that are under source control.

For more information See [Chapter 3, "Using Source Control" on page 27](#).

### Getting Objects

When you get an object, a read-only copy of the selected revision is placed in the target PBL.

#### To get a revision:

- 1 Right-click the object and select Get Latest Version from the resulting pop-up menu. The Get Latest Version dialog box appears with a list of objects.
- 2 Select or deselect objects from the list as needed.
- 3 Do one of the following:

Use advanced options ■ To override the default get options, click the **Advanced** button. The Advanced Get dialog box appears. (For information on advanced options, see ["About Getting Files" on page 28](#).)

Accept defaults ■ To accept the default get options, click **OK**.

### Checking Out Objects

When you check out an object, the revision is locked and a writable object is created in the target PBL.

#### To check out an object:

- 1 Right-click the object and select Check Out from the resulting pop-up menu. The Check Out dialog box appears with a list of objects.
- 2 Select or deselect objects from the list as needed.
- 3 Do one of the following:

Use advanced options ■ To override the default checkout options, click the **Advanced** button. The Advanced Check Out dialog box appears. (For information on advanced options, see ["About Checking Out Files" on page 30](#).)

Accept defaults ■ To accept the default options, click **OK**.

A green checkmark (✓) appears to the left of each object icon to indicate that the objects are checked out by you.



**NOTE** A red X indicates that the object is checked out by another user.

## Undoing Checkout

By default when you undo a checkout, the revision is unlocked without updating the associated Version Manager archive with changes and your local copy of the object is replaced with the latest revision.

### To undo a checkout:

- 1 Right-click the object and select Undo Check Out from the resulting pop-up menu. The Undo Check Out dialog box appears with a list of objects.
- 2 Select or deselect objects from the list as needed.
- 3 Do one of the following:

Use advanced options

- To override the default undo checkout options, click the **Advanced** button. The Advanced Undo Check Out dialog box appears. (For information on advanced options, see ["About Undoing Checkout" on page 32.](#))

Accept defaults

- To accept the default options, click **OK**.

A green dot (•) appears to the left of each object icon to indicate that the objects are no longer checked out.

## Checking In Objects

By default when you check in an object, the revision is unlocked and a new revision is created in the archive and assigned the next number in sequence.

### To check in an object:

- 1 Right-click the object and select Check In from the resulting pop-up menu. The Check In dialog box appears with a list of objects.
- 2 Select or deselect objects from the list as needed.
- 3 In the **Comment** field, enter comments describing the changes you made to the object or objects.



**NOTE** To use a unique description for each object, leave the **Comment** field blank. After you complete the Check In dialog box, the Change Description dialog box will appear for each file in turn.


- 4 Do one of the following:

Use advanced options


- To override the default check-in options, click the **Advanced** button. (For information on advanced options, see ["About Checking In Files" on page 33.](#))

Accept defaults

- To accept the default options, click **OK**. The object or objects are checked in.

A green dot (  ) appears to the left of each object icon to indicate that the objects are checked in.

## Adding New Objects

When you create new objects in a source controlled workspace, a green plus sign (  ) appears to the left of each object icon to indicate that the objects are not yet under source control. To add new objects to source control, follow the procedure ["Adding Objects to Source Control" on page 70](#).



**NOTE** To re-add objects to source control that you have previously removed from source control, use the Workspace tree. PowerBuilder may close unexpectedly if you initiate the add operation from the Library Painter.

Once the objects are added to source control, other users can access them by getting or checking out the target that contains them.

## Adding New Targets or PBLs

If you add a new target (.PBT) or library (.PBL) to a source controlled workspace, you must:

- 1 Add it to source control. See ["Adding Objects to Source Control" on page 70](#).
- 2 Distribute it to each workstation, as was done when the workspace was first set up for use by multiple users. See ["Configuring Workstations in a Multi-User Environment" on page 71](#).



## Chapter 7

---

# Rational Application Developer (Eclipse 3 and 4)

Introduction	78
Accessing Supported Features	78
Setting Up Source Control Projects	79
Using Source Control	84

# Introduction

**Purpose** This chapter has four purposes:

- List the Version Manager features available through the SCC integration to Rational Application Developer (Eclipse 3 and 4) and provide a quick reference to accessing those features



**NOTE** For information on the rich integration to Eclipse-based IDEs, see ["Rational Application Developer Rich Integration \(Eclipse 3 and 4\)" on page 95](#).

- Note any features described in Part 1 of this manual that do not apply to this IDE
- Help administrators set up source control projects and add files to source control
- Help your development team access files that are under source control from within Eclipse

**For more information** See [Part 1, "The Version Manager IDE Client," on page 9](#) for information about:

- Source control concepts
- Source control defaults
- Advanced source control features

## Accessing Supported Features

**What is supported?** Eclipse 3-based IDEs support the full set of source control features available through the Version Manager IDE client, except for the sharing of archives among projects.

To...	Select...	For more information see...
Get revisions	Right-click   Team   Get	<a href="#">"Getting Files" on page 85</a>
Check out revisions	Right-click   Team   Checkout	<a href="#">"Checking Out Files" on page 86</a>
Lock revisions without overwriting work files	Right-click   Team   Lock	<a href="#">"Locking Files" on page 86</a>
Undo checkout of revisions	Right-click   Team   Undo Checkout	<a href="#">"Undoing Checkout" on page 87</a>
Check in revisions	Right-click   Team   Checkin	<a href="#">"Checking In Files" on page 87</a>
Manage version labels	PVCS VM SCC   Run PVCS VM SCC Client	<a href="#">"About Version Labels" on page 34</a>
View properties of revisions or archives	Right-click   Team   Show PVCS VM SCC Properties	<a href="#">"About Properties" on page 46</a>
Monitor source control activity	PVCS VM SCC   Run PVCS VM SCC Client	<a href="#">"Monitoring Source Control Activity with Pulse" on page 47</a>
Generate a history report	Right-click   Team   Show History	<a href="#">"About History Reports" on page 50</a>
Generate a difference report	Right-click   Compare With   PVCS VM SCC Team Provider	<a href="#">"About Difference Reports" on page 52</a>

To...	Select...	For more information see...
Compare workspace with local history	Right-click   Compare With   Local History	<a href="#">"Comparing with Local History" on page 94</a>
Replace workspace with local history	Right-click   Replace With   Local History	<a href="#">"Replacing with Local History" on page 94</a>
Access the Version Manager Options dialog	PVCS VM SCC   Run PVCS VM SCC Client	<a href="#">"About Setting Defaults for Version Manager Options" on page 22</a>
Connect projects to source control	Right-click   Team   Share Project	<a href="#">"Connecting Projects to Source Control" on page 80</a>
Add new files to source control	Right-click   Team   Add	<a href="#">"Adding New Files to Source Control" on page 83</a>
Disconnect projects from source control	Right-click   Team   Disconnect Project	<a href="#">"Disconnecting Projects from Source Control" on page 84</a>
Remove files from source control	Right-click   Team   Remove	<a href="#">"Removing Files from Source Control" on page 84</a>
Open project from source control	PVCS VM SCC   Open from PVCS VM SCC Team Provider	<a href="#">"Connecting Additional Workstations to a Source Control Project" on page 82</a>
Edit files locally without checking them out (local mode)	Right-click   Team   Local Mode	<a href="#">"Using Local Mode" on page 89</a>
Revert local mode files to source control	Right-click   Team   Revert Controlled	<a href="#">"Using Local Mode" on page 89</a>
Enter and exit offline mode	See the text.	<a href="#">"Working Offline" on page 90</a>
Rename or move objects (refactoring)	See the text.	<a href="#">"Using Rename or Move (Refactoring)" on page 88</a>
Refresh source control status	Right-click   Team   Refresh Project Status	<a href="#">"Viewing Source Control Status" on page 84</a>
Synchronize your workspace with source control	Right-click   Team   Synchronize Project	<a href="#">"Synchronizing Your Workspace with Source Control" on page 91</a>

## Setting Up Source Control Projects

**Contents** This section contains information about setting up the Version Manager IDE client to work with Eclipse-based IDEs.

**Prerequisites** Before proceeding, you must use the Version Manager desktop client to create a project database that will contain the source control projects associated with the IDE project (if you don't already have one).



**IMPORTANT!** Eclipse requires JRE 1.3 or 1.4, but it will attempt to use the first JRE found in the path. No JRE is included with Eclipse.

## Excluding Files and Directories from Source Control

You can configure Eclipse to exclude specified files and directories from source control. This can minimize the size of the project database. For example, you could exclude all files with a .tmp file extension or exclude the bin directory and its contents.



**IMPORTANT!** Specify any files and directories you wish to exclude from source control before adding the project they are in to source control.

### To exclude specified files and directories from source control:

- 1 Select Window | Preferences. The Preferences dialog box appears.
- 2 Click the plus sign next to **Team** and select **Ignored Resources**.
- 3 Examine the **Ignore Patterns** list for the file type or directory you wish to exclude from source control. If it is not listed, do the following:
  - a Click the **Add** button. The Enter Ignore Pattern dialog box appears.
  - b Enter a pattern that defines a file type or directory to ignore. Use wildcards if needed:
    - Asterisks (\*) represent one or more characters.
    - Question marks (?) represent one character.
  - c Click **OK**.
- 4 In the **Ignore Patterns** list, ensure that a check mark appears next to each file type and directory you want to exclude from source control.
- 5 Click **OK**.

## Connecting Projects to Source Control

Before you can place IDE files under source control, you must connect your IDE project to source control.



**IMPORTANT!** Specify any files you wish to exclude from source control before adding the project they are in to source control. See ["Excluding Files and Directories from Source Control" on page 80](#)".

### To connect to a Version Manager project database:

- 1 Right-click the project icon in the Package Explorer or Navigator. A pop-up menu appears.
- 2 Select Team | Share Project. The Share Project dialog box appears.
- 3 Select **PVCS VM SCC Team Provider** from the **Select a repository type** list.
- 4 Click the **Next** button. The Create or Select a PVCS VM SCC Team Provider Project dialog box appears.
- 5 Click the **Create/Select** button. The Add Project to Source Control dialog box appears.



- 6 The default project database displays under **Source Control Project**. If you wish to add the files to a different project database, click the **Open Database** button and browse to select one or click the **Browse Database** button to browse a list of project databases published by your Version Manager File Servers if you have any.
- 7 Do one of the following:
  - To add to an existing source control project, select one from under Source Control Project. Proceed to [Step 8](#).
  - To create a new Version Manager project:
    - a Under Source Control Project, select the location in the project database where you want to create the new project.
    - b Click the **Create Project** button. The Create Source Control Project dialog box appears.  
  
The Project Database Information group displays the name and location of the current project database, and the location of the new project within the database.
    - c By default, the new Version Manager project uses the same name as the IDE project. If necessary, enter a different name in the **Project Name** field.  
  
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 (< >).
    - d Click **OK**. The Add Project to Source Control dialog box reappears with the new project displayed under **Source Control Project**.
- 8 Click **OK**. The Create or Select a PVCS VM SCC Team Provider Project dialog box reappears.
- 9 Click **Finish**. Select Team | Add. The Add to Source Control dialog box appears with a list of selected files to be added.
- 10 Do any of the following:
  - To change which files are selected, select and deselect files in the **Files in** list.




**NOTE** You must add the .project file to source control.

- To check out and lock the files immediately after adding them to source control, select the **Keep Checked Out** check box.
  - To add any empty folders to source control, select the **Add Empty Folders** check box. The empty folders will then be included when developers open the project from source control.
- 11 Enter a description of the files in the **Comment** field. The description will be applied to all selected files.



**TIP** To use a unique description for each file, leave the **Comment** field blank. After you complete the Add to Source Control dialog box, the Change Description dialog box will appear. Enter a description for each file in turn.

**12** Do one of the following:

- |                      |                                                                                                                                                                                                                                                                                                                                                           |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Use advanced options | ■ To assign a version label or promotion group to the initial revision, click the <b>Advanced Options</b> button. The Advanced Add dialog box appears. (For information on advanced options, see <a href="#">"Advanced Add Options" on page 24.</a> )                                                                                                     |
| Accept defaults      | ■ To accept the default add options, click <b>OK</b> . The selected files are added to source control, creating archives in the specified archive directory. A gold cylinder (  ) is added to the icon of each item to indicate that the item is under source control. |

## Connecting Additional Workstations to a Source Control Project

Once an Eclipse project is under source control, you can open it from any workstation that has access to the Version Manager project database. There are two ways to connect additional workstations to a source control project:

- Open the project from the PVCS VM SCC Team Provider.
- Export and import a Project Set file.

### ***Open from the PVCS VM SCC Team Provider***

#### **To add existing repository projects to your workspace:**

- 1** Select PVCS VM SCC | Open from PVCS VM SCC Team Provider. The Get Project from Source Control dialog box appears with a list of projects in the current project database.

If the database containing the project you wish to open is not displayed, click the **Open Database** button to browse for the correct database.

- 2** Select the project you wish to open.
- 3** If the Login dialog box displays, enter your user ID and password.
- 4** Enter a location for all of your project files in the **Workfile Location** field, or click the browse button to select a location.
- 5** Click **OK**.

### ***Export/Import a Project Set File***

A Project Set file contains the path information needed to connect other workstations to an existing source control project.

#### **To export a Project Set file:**

- 1** Select File | Export. The Export dialog box appears.
- 2** Select **Team Project Set** from the **Select an export destination** list.
- 3** Click the **Next** button.
- 4** Enter a path and file name in the **File name** field or browse to select a destination.
- 5** Click the **Finish** button. A \*.PSF file is created in the selected directory.

- 6 Distribute the \*.PSF file to each workstation or make it available from a network location.

#### To import a Project Set file:

- 1 Select File | Import. The Import dialog box appears.
- 2 Select **Team Project Set** from the **Select an import source** list.
- 3 Click the **Next** button.
- 4 Enter the path and file name of the \*.PSF file in the **File name** field or browse to select it.
- 5 Click the **Finish** button.

## Adding New Files to Source Control

Once you have connected an Eclipse project to source control, you can add new IDE files to source control at any time. (For details on connecting Eclipse to project databases, see ["Connecting Projects to Source Control" on page 80.](#))

#### To add new files to an IDE project that is already under source control:

- 1 Right-click on the root of the IDE project in the Package Explorer or Navigator. A pop-up menu appears.
- 2 Select Team | Add. The Add to Source Control dialog box appears with a list of selected files to be added.
- 3 Do any of the following:
  - To change which files are selected, select and deselect files in the **Files in** list.
  - To check out and lock the files immediately after adding them to source control, select the **Keep Checked Out** check box.
  - To add any empty folders to source control, select the **Add Empty Folders** check box. The empty folders will then be included when developers open the project from source control.
- 4 Enter a description of the files in the **Comment** field. The description will be applied to all selected files.



**TIP** To use a unique description for each file, leave the **Comment** field blank. After you complete the Add to Source Control dialog box, the Change Description dialog box will appear. Enter a description for each file in turn.

- 5 Do one of the following:

Use advanced options

- To assign a version label or promotion group to the initial revision, click the **Advanced Options** button. The Advanced Add dialog box appears. (For information on advanced options, see ["Advanced Add Options" on page 24.](#))

Accept defaults

- To accept the default add options, click **OK**. The selected files are added to source control, creating archives in the specified archive directory. A gold cylinder (📦) is added to the icon of each item to indicate that the item is under source control.

## Disconnecting Projects from Source Control

When you disconnect a project from source control, Version Manager does not delete the Version Manager archives; it simply removes the association between the IDE project and archives.

### To disconnect a project from source control:

- 1 Right-click the project icon in the Package Explorer or Navigator. A Pop-up menu appears.
- 2 Select Team | Disconnect Project. A prompt appears to confirm that you wish to disconnect the project.
- 3 Click **Yes**.

## Removing Files from Source Control

When you remove a file from source control, Version Manager does not delete the Version Manager archive; it simply removes the association between the IDE file and archive. You can always add the file back to source control later.

### To remove files from source control:

- 1 Select the files you want to remove in the Package Explorer or Navigator and right-click. A pop-up menu appears.
- 2 Select Team | Remove. The Remove from PVCS VM SCC Team Provider dialog box appears with a list of files to be removed.
- 3 Select or deselect files in the list as needed.
- 4 Click **OK**.

Returning files to source control      For information on returning files to source control, see ["Returning Files to Source Control" on page 24](#).










## Using Source Control

Contents      This section contains procedural information about viewing and editing files that are under source control.

## Viewing Source Control Status

Version Manager indicates status and revision information by displaying graphics and text next to the object icons in the Package Explorer and Navigator, as in the following image.

The following table lists the information that can be displayed.

This graphic/text . . .	Indicates . . .
(Online) (Offline)	Whether you are working in online or offline mode.
(1.0)	The revision you got or checked out from source control to your workspace.
[1.2]	The current tip revision in the source control repository if the tip is not the revision in your workspace.
	The object is under source control and it is checked in.
	You have the object checked out.
	Someone else has the object checked out.
	Multiple users have the object checked out and you are not one of them.
	Multiple users have the object checked out and you are one of them.
	The object is in local mode.
	The object contains objects that have been modified locally or for which there are newer revisions in the source control repository.
	The object has been modified locally and is out of synch with the repository.
	The repository contains a newer revision of this object. <b>NOTE</b> If you modify the local object, the graphic will change to a right-pointing arrow.

To refresh status information, right-click on a project and select Team | Refresh Project Status.

### Enabling Icon Glyphs

By default, icon glyphs are enabled.

#### To enable Icon Glyphs:

- 1 Select Window | Preferences. The Preferences dialog box appears.
- 2 Click the plus sign next to **Workbench** and select **Label Decorations**.
- 3 Select **PVCS VM SCC** in the **Available label decorations** list.
- 4 Click **OK**.

## Getting Files

When you get files, read-only copies of the selected revisions are placed in the workfile location.

**To get revisions:**

- 1 In the Package Explorer or Navigator, select the files you want to get and right-click. A pop-up menu appears.
- 2 Select Team | Get. The Get dialog box appears with a list of selected files. You can change your selection by selecting and deselecting the files in this list.
- 3 Do one of the following:

Use advanced  
options

- To override the default get options, click the **Advanced Options** button. The Advanced Get dialog box appears. (For information on advanced options, see ["About Getting Files" on page 28.](#))

Accept defaults

- To accept the default get options, click **OK**. Read-only copies of the selected revisions are placed in the workfile location.

## Checking Out Files

When you check out a file, the revision is locked and a writable workfile is created in the workfile location.

**To check out files:**

- 1 In the Package Explorer or Navigator, select the files you want to check out and right-click. A pop-up menu appears.
- 2 Select Team | Checkout. The Check Out dialog box appears with a list of selected files. You can change your selection by selecting and deselecting the files in this list.
- 3 (Optional) To modify the files locally without first placing a source control lock on them, select the **Make Files Local** checkbox.



**NOTE** Other users can still check out, modify, and check in changes to any files you use in local mode. You must resolve any resulting code conflicts if you decide to check in the changes you make while in local mode. See ["Using Local Mode" on page 89.](#)

- 4 Do one of the following:

Use advanced  
options

- To override the default checkout options, click the **Advanced Options** button. The Advanced Check Out dialog box appears. (For information on advanced options, see ["About Checking Out Files" on page 30.](#))

Accept defaults

- To accept the default checkout options, click **OK**. A check mark (☑) appears next to each file icon to indicate that the files are checked out.

## Locking Files

When you lock a file, the revision is locked in source control but no files are written to the workfile location so your existing workfile is not overwritten. Locking a revision does not change the write attribute of the workfile.

**To lock files:**

- 1 In the Package Explorer or Navigator, select the files you want to lock and right-click. A pop-up menu appears.

- 2 Select Team | Lock. The Check Out dialog box appears with a list of selected files. You can change your selection by selecting and deselecting the files in this list.
- 3 (Optional) To modify the files locally without first placing a source control lock on them, select the **Make Files Local** checkbox.



**NOTE** Other users can still check out, modify, and check in changes to any files you use in local mode. You must resolve any resulting code conflicts if you decide to check in the changes you make while in local mode. See ["Using Local Mode" on page 89](#).

- 4 Do one of the following:

Use advanced options

- To override the default checkout options, click the **Advanced Options** button. The Advanced Check Out dialog box appears. (For information on advanced options, see ["About Checking Out Files" on page 30](#).)

Accept defaults

- To accept the default checkout options, click **OK**. A check mark (☑) appears next to each file icon to indicate that the files are locked.

## Undoing Checkout

When you undo a checkout, the archive is unlocked and a read-only workfile is left in the workfile location. No changes are checked into the archive.

### To undo a checkout:

- 1 In the Package Explorer or Navigator, select the files you want to unlock and right-click. A pop-up menu appears.
- 2 Select Team | Undo Checkout. The Undo Check Out dialog box appears with a list of selected files. You can change your selection by selecting and deselecting the files in this list.
- 3 Do one of the following:

Use advanced options

- To override the default undo checkout options, click the **Advanced Options** button. The Advanced Undo Check Out dialog box appears. (For information on advanced options, see ["About Undoing Checkout" on page 32](#).)

Accept defaults

- To accept the default undo checkout options, click **OK**. The check mark is removed from each file icon to indicate that the files are unlocked.

## Checking In Files

By default, the following occurs when you check in a workfile:

- If you modified the file in local mode and the tip revision is newer than the revision you started with, a prompt appears for the Merge Tool.
- A new revision is created and assigned the next number in sequence.
- A read-only workfile is left in the workfile location.
- The archive is unlocked.

**To check in files:**

- 1 In the Package Explorer or Navigator, select the files you want to check in and right-click. A pop-up menu appears.
- 2 Select Team | Checkin. The Check In dialog box appears with a list of selected files. You can change your selection by selecting and deselecting the files in this list.
- 3 Do any of the following:

Retain lock

- To retain the lock on the files after they are checked in, select the **Keep Checked Out** check box.

Compare files

- To compare two revisions in an archive, revisions in two archives, two workfiles, or a revision and a workfile:
  - a Select a single file from the **Files in** list.
  - b Click the **Differences** button. The Show Differences dialog box appears. For more information see ["About Difference Reports" on page 52](#).

- 4 Enter a description of the changes you made in the **Comment** field. The description will be applied to all selected files.



**TIP** To use a unique description for each file, leave the Comment field blank. After you complete the Add to Source Control dialog box, the Change Description dialog box will appear. Enter a description for each file in turn.

You will not be prompted for a description for unmodified files. Unmodified files will be given a description of "No Change."

- 5 Do one of the following:

Use advanced options

- To override the default check-in options, click the **Advanced Options** button. The Advanced Check In dialog box appears. (For information on advanced options, see ["About Checking In Files" on page 33](#).)

Accept defaults

- To accept the default check-in options, click **OK**. The check mark is removed from each file icon to indicate that the files are checked in.

## Using Rename or Move (Refactoring)

**NOTE**

- You *can* refactor when in offline mode, **but there are significant risks**. If other users have refactored the code while you were offline, your offline refactoring will be unaware of the changes they made. This will result in errors which will require manual fixes.
- Refactoring may cause the archive name to differ from the workfile name--which is not compatible with the Command-Line Interface (CLI). Since the CLI is not project aware, it requires that the names match. However, you can use the Project Command-Line Interface (PCLI), instead. (This does not impact the Version Manager desktop client or the IDE client since they resolve workfile and archive names via the project metadata.)



**To use Rename or Move:**


- 1 Before using Rename or Move on a project accessed by multiple users, all users should check in their changes.
- 2 Open a perspective in which the Package Explorer is available, such as the Java Perspective.
- 3 Select the Package Explorer as the active pane.
- 4 Select the item you wish to rename or move and do one of the following:
  - Select Refactor | Rename (or Move).
  - Right-click and select Refactor | Rename (or Move).
- 5 You may be asked to confirm the action of checking in from a different location than the one to which the file is checked out. You must choose to proceed with this action.
- 6 To work with the modified project, all users must get the updated project from source control. See ["Getting Files" on page 85](#) or ["Synchronizing Your Workspace with Source Control" on page 91](#).

## Using Local Mode

You can edit projects and/or files without checking them out from source control by working on them in local mode. Local mode can be enabled for individual projects or files, unlike offline mode which affects the entire IDE.

**Putting Files into Local Mode****To put files into local mode:**

- 1 In the Package Explorer or Navigator, select the project(s) or file(s) you want to work on in local mode and right-click. A pop-up menu appears.
- 2 Do one of the following:
  - Select Team | Local Mode.
  - Select Team | Checkout then select the **Make Files Local** checkbox on the Check Out dialog box. Complete the checkout operation as normal.

The local mode graphic () appears on the icons of the selected files.



**TIP** You can also put files into local mode by selecting the **Make Files Local** check box on the Check Out dialog box.

**Reverting Local Mode Files to Controlled Mode****To revert files to controlled mode:**

- 1 In the Package Explorer or Navigator, select the local mode project(s) or file(s) you want to revert to controlled mode and right-click. A pop-up menu appears.

- 2 Select Team | Revert Controlled. Depending upon what changes are found, any of the following may occur:

- The Revert to Controlled dialog box appears with a list of files that have not changed while in local mode.

Select or deselect files in this list as needed and click **OK**. The files are removed from local mode.

- The Mark as Checked Out dialog box appears with a list of files that you have edited while in local mode.

**a** Select or deselect files in this list as needed and click **OK**.

**b** If other users have checked in changes to a file while you were offline, the Confirm Create Branch dialog box appears.

Select **Yes** or **Yes to All**.



**NOTE** Later, when you check in the file(s), you will be asked if you wish to merge to the tip revision. If you select **Yes**, the Merge tool launches. If you select **No**, the revision will be checked in as a branch off of the revision you started with.

The selected files are removed from local mode and are shown as being checked out by you. To add your local mode edits to source control, check in the files.

The deselected files are removed from local mode. To overwrite your local mode edits, do a get or checkout.

## Working Offline

You can configure the integration to give you the option of working in offline mode. This allows you to edit projects in your local workspace without checking the files out from source control. You may find this feature useful if you wish to:

- Experiment with code and not check in the resulting changes.
- Continue development while away from your network connection then synchronize your offline changes with source control when you can reconnect.



**NOTE** You *can* refactor when in offline mode, **but there are significant risks**. If other users have refactored the code while you were offline, your offline refactoring will be unaware of the changes they made. This will result in errors which will require manual fixes.

### Enabling the Online/Offline Prompt

You can enable a prompt that appears while the IDE is starting up that asks if you wish to work in online or offline mode.

#### To enable the Online/Offline prompt:

- 1 Select Window | Preferences. The Preferences dialog box appears.
- 2 Select **PVCS VM SCC** from under **Team** in the left pane. The PVCS VM SCC pane appears.
- 3 Select the **Ask to go Offline at startup** check box.

- 4 Click **OK**.

### To Enter/Exit Offline Mode

Once the Online/Offline prompt is enabled, you can enter or exit Offline mode when ever you start the IDE.

#### To enter/exit offline mode:

- 1 Launch the IDE. A prompt appears as the IDE starts up.
- 2 Click **Online** to enter online or **Offline** to enter offline mode. The mode you select will remain in effect until you restart the IDE and make another selection. The current mode is displayed next to each project in the Package Explorer and Navigator as (Online) or (Offline).



**NOTE** If you wish to check in the changes you made while working in offline mode, you must enter online mode then synchronize your local workspace with source control (see the next section).

## Synchronizing Your Workspace with Source Control

In a multi-user environment, you should synchronize your workspace with source control to:

- Remove files from your local workspace that other users have removed from the source control project.
- Add files to your local workspace that other users have added to the source control project.
- Add files to the source control project that you have added to your local workspace.
- Update the content of files in your local workspace that other users have modified and checked into the source control project.
- Update the content of files in the source control project that you have modified in your local workspace.

#### To synchronize your workspace with source control:

- 1 In the Package Explorer or Navigator, select the project(s) you want to synchronize and right-click. A pop-up menu appears.
- 2 Select **Team | Synchronize Project**. The Synchronize dialog box appears.
- 3 Select the type of synchronization operations you want to perform:
  - **Revert files to controlled:** Reverts local mode files in your local workspace to active source control status. If the files have been modified, they will be marked as checked out. If you also select **Refresh project files**, the modified files will be checked in.
  - **Refresh project structure:** Updates your workspace and the source control project by adding or deleting projects and files.
  - **Refresh project files:** Updates the contents of files in your local workspace and the source control project.

- 4 Click **OK**. Depending upon the type of synchronization you selected and what changes are found, any of the following may occur:

Revert files to controlled

- The Revert to Controlled dialog box appears with a list of files that have not changed while in local mode.
  - a Select or deselect files in the list as needed.
  - b Click **OK**. The files are removed from local mode.
- The Mark as Checked Out dialog box appears with a list of files that you have edited while in local mode.
  - a Select or deselect files in the list as needed.
  - b Click **OK**. The files are removed from local mode and are shown as being checked out by you.

To add your local mode edits to source control, check in the files. If you selected **Refresh project files** in the Synchronize dialog box, the files will be checked in by a subsequent dialog box; see below.

Refresh project structure


- The Refresh Project Structure - New Repository Files dialog box appears with a list of files that are in the source control project but not in your local workspace. You can select and deselect files in this list.
  - a Select or deselect files in the list as needed.
  - b Do one of the following:
    - To override the default get options, click the **Advanced Options** button. The Advanced Get dialog box appears. (For information on advanced options, see ["About Getting Files" on page 28.](#))
    - To accept the default get options, click **OK**. Read-only copies of the files are placed in the workfile location.
- The Refresh Project Structure - New Local Files dialog box appears with a list of files that are in your local workspace but not in the source control project.
  - a Do any of the following:
    - To change which files are selected, select and deselect files in the **Files in** list.
    - To check out and lock the files immediately after adding them to source control, select the **Keep Checked Out** check box.
    - To add any empty folders to source control, select the **Add Empty Folders** check box.
  - b Enter a description of the files in the **Comment** field. The description will be applied to all selected files.



**TIP** To use a unique description for each file, leave the **Comment** field blank. After you complete the Add to Source Control dialog box, the Change Description dialog box will appear. Enter a description for each file in turn.

- c Do one of the following:
  - To assign a version label or promotion group to the initial revision, click the **Advanced Options** button. The Advanced Add dialog box appears. (For information on advanced options, see ["Advanced Add Options" on page 24.](#))

Refresh project  
files

- To accept the default add options, click **OK**. A gold cylinder () is added to the icon of each item to indicate that the item is under source control.
- The Refresh Project Structure - Remove from Workspace dialog box appears with a list of files that have been removed from source control but still exist in your local workspace.
  - a Select or deselect files in the list as needed.
  - b Click **OK**.
- The Check in Edited Files dialog box appears.
  - a Select or deselect files in the list as needed.
  - b To check out and lock the files immediately after checking them in, select the **Keep Checked Out** check box.
  - c Enter a description of the edits you made in the **Comment** field.
  - d Do one of the following:
    - To override the default check-in options, click the **Advanced Options** button. The Advanced Check In dialog box appears. (For information on advanced options, see ["About Checking In Files" on page 33.](#))
    - To accept the default check-in options, click **OK**. The check mark is removed from each file icon to indicate that the files are checked in.
- The Get Changed Files dialog box appears.
  - a Select or deselect files in the list as needed.
  - b Do one of the following:
    - To override the default get options, click the **Advanced Options** button. The Advanced Get dialog box appears. (For information on advanced options, see ["About Getting Files" on page 28.](#))
    - To accept the default get options, click **OK**. Read-only copies of the files are placed in the workfile location.

### Enabling Refresh Project Structure Report on Open

You can configure the integration to notify you if the project you are opening has a different structure in the repository as compared to your workspace. If it does, a prompt will appear to remind you to synchronize your workspace with the repository. Click **OK** to dismiss the prompt.



**NOTE** The prompt appears only if files have been added to or removed from the project. The contents of the files are not considered.

#### To enable the refresh project structure on open prompt:

- 1 Select Window | Preferences. The preferences dialog box appears.
- 2 Select **PVCS VM SCC** from under Team in the left pane.
- 3 Select the **Refresh Project Structure Report on open** check box.
- 4 Click **OK**.

## Comparing with Local History

You can compare a workfile with a local history of the changes made to that workfile. A new entry is made in the local history each time you save changes to a file.



**NOTE** To configure how many entries are retained and for how long, select Window | Preferences then select **Local History** from under **Workbench**. For more information, see the WebSphere Studio help.

### To compare with local history:

- 1 In the Package Explorer or Navigator, select the file you want to compare and right-click. A pop-up menu appears.
- 2 Select Compare With | Local History. The Compare with Local History dialog box appears.
- 3 Select a local history entry from the Local History of *filename* pane.
- 4 Use the **Select Next Change** (↓) and **Select Previous Change** (↑) buttons to step through the changes.
- 5 Click **OK** to exit the compare.

## Replacing with Local History

You can replace a workfile with an entry from the local history of changes made to that workfile. A new entry is made in the local history each time you save changes to a file.



**NOTE** To configure how many entries are retained and for how long, select Window | Preferences then select **Local History** from under **Workbench**. For more information, see the WebSphere Studio help.

### To replace with local history:

- 1 In the Package Explorer or Navigator, select the file you want to replace and right-click. A pop-up menu appears.



**NOTE** The file must be checked out.

- 2 Select Replace With | Local History. The Replace from Local History dialog box appears.
- 3 Select a local history entry from the Local History of *filename* pane.
- 4 Use the **Select Next Change** (↓) and **Select Previous Change** (↑) buttons to step through the changes.
- 5 Do one of the following:
  - To replace the workfile with the selected history entry, click the **Replace** button.
  - To close the dialog box without replacing the workfile, click the **Cancel** button.

## Chapter 8

---

# Rational Application Developer Rich Integration (Eclipse 3 and 4)

Introduction	96
Accessing Supported Features	97
Integration Overview	98
Collaborative Process Overview	99
Setting Up Source Control Projects	102
Using Source Control	108
Associating and Working on SBM Issues	125
Setting Default Options	131

# Introduction

Purpose This chapter has the following purposes:

- List the Version Manager and Solutions Business Manager (SBM) features available through the rich integration to Rational Application Developer (Eclipse 3 and 4), and provide a quick reference to accessing those features



**NOTE** For information on the old integration to Eclipse-based IDEs, see "[Rational Application Developer \(Eclipse 3 and 4\)](#)" on page 77.

- Note any features described in Part 1 of this manual that do not apply to this integration



**NOTE** Other than the section on source control concepts, Part 1 of this manual does not apply to this integration.

- Help administrators set up source control projects and add files to source control
- Help your development team access files that are under source control from within Eclipse
- Help your development team associate SBM issues with source controlled files from within Eclipse



**IMPORTANT!** The rich integration 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 use the desktop client to 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 Eclipse. 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.
- Remember that when you open a project from source control that you must specify the Version Manager workspace to use. If you wish to see the files and revisions defined by a different default version, then you must do an open from source control and specify the workspace that is associated with the desired label.



# Accessing Supported Features

What is supported? Eclipse 3 based IDEs support the full set of source control and issue management features available through the Version Manager rich IDE integration.

To...	Select...	For more information see...
Get revisions	Right-click   Team   Get	<a href="#">"Getting Files" on page 112</a>
Check out revisions	Right-click   Team   Checkout	<a href="#">"Checking Out Files" on page 113</a>
Undo checkout of revisions	Right-click   Team   Undo Checkout	<a href="#">"Undoing Checkout" on page 114</a>
Check in revisions	Right-click   Team   Check In	<a href="#">"Checking In Files" on page 115</a>
Label the latest revision	Right-click   Team   Label	<a href="#">"Assigning Version Labels" on page 111</a>
Label a prior revision	Right-click   Team   History	<a href="#">"Assigning Version Labels" on page 111</a>
Promote the latest revision to the next level in the promotion hierarchy	Right-click   Team   Promote	<b>NOTE:</b> <ul style="list-style-type: none"> <li>■ You cannot promote files that are checked out.</li> <li>■ You can promote during check in. See <a href="#">"Checking In Files" on page 115</a>.</li> </ul>
Promote a specific revision to the next level in the promotion hierarchy	(From the History view) Right-click   Promote to next	<a href="#">"Working in the History View" on page 109</a>
View revision history: - Revisions - Version labels - Promotion groups - Dates - Comments	Right-click   Team   History	<a href="#">"Working in the History View" on page 109</a>
Compare workfiles with latest revision	Right-click   Compare With   PVCS VM Revision	<a href="#">"Comparing with the Latest Revision" on page 122</a>
Compare workfiles with local history	Right-click   Compare With   Local History	<a href="#">"Comparing with Local History" on page 123</a>
Compare workfiles with each other	Right-click   Compare With   Each Other	<a href="#">"Comparing Workfiles with Each Other" on page 123</a>
Replace workfiles with local history	Right-click   Replace With   Local History	<a href="#">"Replacing with Local History" on page 124</a>
Replace workfiles with latest revision	Right-click   Replace With   Latest PVCS VM Revision	<a href="#">"Replacing with Latest Revision" on page 125</a>
Connect projects to source control	Right-click   Team   Share Project	<a href="#">"Adding Projects to Source Control" on page 104</a>
Disconnect projects from source control	Right-click   Team   Disconnect	<a href="#">"Disconnecting Projects from Source Control" on page 108</a>

To...	Select...	For more information see...
Import projects from source control	PVCS VM RIDE   Import Project from Version Manager	<a href="#">"Connecting Additional Workstations to an Existing Source Control Project" on page 106</a>
Rename or move objects (refactoring)	See the text.	<a href="#">"Using Rename or Move (Refactoring)" on page 118</a>
Refresh all source control status	PVCS VM RIDE   Refresh All Status	<a href="#">"Viewing Source Control Status" on page 109</a>
Refresh source control status for selected objects	Right-click   Team   Refresh Status	<a href="#">"Viewing Source Control Status" on page 109</a>
Synchronize your entire workspace with source control	PVCS VM RIDE   Compare Workspaces	<a href="#">"Comparing and Synchronizing Your Workspace with Source Control" on page 118</a>
Synchronize the selected objects with source control	Right-click   Team   Compare Workspaces	<a href="#">"Comparing and Synchronizing Your Workspace with Source Control" on page 118</a>
Work with SBM Issues	PVCS VM RIDE   Show Issues	<a href="#">"Associating and Working on SBM Issues" on page 125</a>
Get help	(Click in the dialog or view you want help with, and press the F1 key.)	<b>NOTE</b> F1 will invoke help in most dialogs and views.

## Integration Overview

The Version Manager rich integration provides a powerful set of collaborative tools to help development teams manage their source code. The integration includes:

- **Workspace Comparison and Synchronization**

You can easily compare the state of all files in your local workspace with the corresponding Version Manager project. You can determine what the differences are and whether your workspace or the Version Manager repository need to be updated. With the click of a button, you can then automatically check in all of your changes to Version Manager and get all updates to your local workspace. See ["Comparing and Synchronizing Your Workspace with Source Control" on page 118](#).

- **Automated File Merging**

When you check in or synchronize your workspace, your changes are automatically merged with any recent changes to the same files in the Version Manager repository.

- **File Comparison and Conflict Resolution**

You can compare specific local files to the latest revision of the files in the Version Manager repository. If necessary, you can directly edit the files in order to resolve any conflicts before checking in the files. See ["Comparing and Synchronizing Your Workspace with Source Control" on page 118](#).

- **Pragmatic Locking**

You can check in any file, any time, without first locking it. See ["Working on Files Without Locking Them" on page 101](#).

## Working Offline

You can work on an Eclipse project while disconnected from the network, for example while traveling. Once you are back on the network, you can synchronize your changes with the Version Manager repository.

### To work offline:

- 1 Before you disconnect from the network, get or check out the desired files to your local Eclipse workspace.
- 2 Disconnect from the network.



**NOTE** The Rich Integration to Eclipse does not have an "Offline Mode", as did the older integration. You can now simply unplug the network cable, no special "mode" required.

- 3 Edit the Eclipse project as desired, saving changes to your local Eclipse workspace.
- 4 Reconnect to the network.
- 5 Open the Eclipse project and run Compare Workspaces (Team | Compare Workspaces). See ["Comparing Your Workspace with Source Control" on page 119](#).

## SBM Integration

If your organization uses Solutions Business Manager (SBM) to track development issues, such as defects and tasks, you can submit and modify SBM issues from within Eclipse, and associate issues with specific revisions of files. When you associate issues with files, the revision number is added to the issue.

For detailed information on working with SBM issues, see ["Associating and Working on SBM Issues" on page 125](#).

## Collaborative Process Overview

The Version Manager integration supports a flexible range of collaborative development processes. You can:

- |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Workspaces          | <ul style="list-style-type: none"> <li>■ Use Version Manager workspaces to precisely define which files developers will work on, and to automate the assignment of version labels in order to enforce maintenance of the workspace. Workspaces greatly simplify the process of sharing files in a development project, by ensuring that all developers working within the same branch of development see and update the same files. See <a href="#">"Using Workspaces" on page 100</a>.</li> </ul> |
| Pragmatic locking   | <ul style="list-style-type: none"> <li>■ Use an optimistic, or pragmatic, locking model to allow all users to work on common files without locking them. See <a href="#">"Working on Files Without Locking Them" on page 101</a>.</li> </ul>                                                                                                                                                                                                                                                       |
| Pessimistic locking | <ul style="list-style-type: none"> <li>■ Use a pessimistic locking model to request that users check out files before they modify the files and check in their changes. This prevents other users from checking in changes until the files are unlocked. See <a href="#">"Checking Out Files with Locks" on page 102</a>.</li> </ul>                                                                                                                                                               |

## Using Workspaces

The rich integration to Eclipse makes extensive use of Version Manager workspaces to simplify the collaborative process. Version Manager workspaces represent a collection of specific files, each of which shares a common default version label. Workspaces enable developers to get and work only on files associated with specific development efforts or projects, as defined by the default version label.



**NOTE** Version Manager workspaces also define a default workfile location, but this workfile location does not apply when using the rich integration to Eclipse. Instead, your IDE's workspace defines the workfile location. However, the Version Manager workfile location applies when using other Version Manager clients, such as the web or desktop clients.

The following is an overview of workspace setup and usage for the rich integration to Eclipse:

- 1 **The Version Manager administrator sets up the workspace in Version Manager.** Unique workspaces can be set up for each project or subproject, and even for each developer. The simplest workflow is to define a common workspace that all members on a project team can share, and that defines the correct default version label for the project. If developers will use different workspaces but will work on a common project, the default version label should be set to the same value for each workspace. Because it is the default version label that determines what each developer can see and modify, it is important that the default is common for all developers who will work on the same branch of development. See the *Version Manager Administrator's Guide* for details.



**IMPORTANT!** If developers will work in different workspaces, it is critical to note that any changes to project structure that occur within one workspace will also be evident in another workspace. For example, if a developer working in a workspace called "Major\_Releases" renames several files, then those files will also be renamed in any other workspace that includes them. When defining workspaces, it is critical to carefully evaluate such dependencies.

- 2 **The administrator defines the default version label for the workspace.** The default version label will in turn determine precisely which files belong to the workspace.

For example, if the floating label "Latest" is assigned to all files in the project, and if the workspace should include the most recent versions of all files in the project, define the default version label as "Latest."

Or, if the workspace should include all files in a branch for which the branch version label is "branch\_01," then the default version label for the workspace should be defined as "branch\_01."

See the *Version Manager Administrator's Guide* for details.

- 3 **A developer selects the appropriate workspace and adds the Eclipse project to source control.** By selecting the appropriate workspace, the default version label for that workspace is assigned to all files that are added to Version Manager. For example, if the developer chooses a workspace for which "Latest" is the default version label, then the "Latest" label is assigned to all files.
- 4 **Other developers select the appropriate workspace and import the IDE project from source control.** This choice then determines what files are copied to

their local workspaces. For example, if developers choose a workspace for which "Latest" is the default version label, then only those files to which "Latest" is assigned are copied to their local workspaces.

**5 Developers synchronize workspaces and resolve all changes, automatically checking in / getting files with the default version label.** When a developer resolves all changes:

- Any new local files are added to the Version Manager project, and the default version label for the current workspace is assigned to them.
- Any new files in the Version Manager project are copied to the local workspace if they have the default version label.

## Working on Files Without Locking Them

Also known as *pragmatic* or *optimistic locking*, this model allows multiple users to get and modify files at the same time. Each user's modifications will be auto-merged with other users' changes when the files are checked in. If any merge conflicts result from attempting to auto-merge multiple users' changes, use the Compare Workspaces feature to evaluate and resolve the conflicts before completing check-in.



**NOTE** By default, optimistic locking is enabled. However, the administrator can turn it off for any given project database.

The following steps illustrate this process:

- 1 At the beginning of each work day, make sure that your local workspace has all of the latest updates from the Version Manager repository. To do this, you can synchronize your local workspace with the Version Manager repository (see ["Comparing and Synchronizing Your Workspace with Source Control" on page 118](#)). If the local files already exist and are different from the latest revisions in the Version Manager repository, the revisions are merged. If any merge conflicts result from the attempt to auto-merge the different revisions, double-click to open them in the compare editor and then evaluate and resolve the merge conflicts.
- 2 Edit the files.
- 3 When you have finished your work on the files, check them in to Version Manager. You can either check in specific files (see ["Checking In Files" on page 115](#)) or synchronize your workspace with the Version Manager repository (see ["Comparing and Synchronizing Your Workspace with Source Control" on page 118](#)).



**NOTE** During the actual check-in operation, Version Manager briefly locks the files while adding the new revisions to the repository.

### Example: Collaborative Development

A team of developers work together on files stored in a project called "Java," in a project database called "Source." Their organization allows multiple developers to work on common files at the same time. Their development workflow is as follows:

- 1 At the beginning of every day, each developer can update their local workspace with the latest changes under the "Java" project in Version Manager. The development

team works from sites in multiple nations; This step ensures that all developers have all of the latest updates from all development sites. See ["Comparing and Synchronizing Your Workspace with Source Control" on page 118](#).

- 2 At the end of each day, each developer synchronizes their workspace with the Version Manager repository. If multiple developers have modified the same files, the merge functionality will attempt to auto-merge each user's changes into the new revisions. All merged and changed files are copied to the local workspace. If merge conflicts result from the attempt to auto-merge, the developer performing the check-in must resolve the conflicts, and then synchronize again. See ["Comparing and Synchronizing Your Workspace with Source Control" on page 118](#).

## Checking Out Files with Locks

In this workflow, check out any file that you intend to modify. This locks the file in Version Manager, which prevents other users from checking in changes until you unlock the files. Once you have completed your changes, check in the files. This ensures that changes from multiple users will never result in conflicts.



**IMPORTANT!** Checking in local changes does not synchronize renamed, moved, or deleted objects. You must synchronize your local workspace with the Version Manager repository in order to check in refactored and deleted objects. See ["Comparing and Synchronizing Your Workspace with Source Control" on page 118](#).

### **Example: Locking Files**

As part of a larger development team working on a complex and interdependent code base, Joe and Carol frequently work on the same files. A typical day might go something like this:

- 1 To begin work on a set of files, Joe checks them out from source control (see ["Checking Out Files" on page 113](#)).
- 2 When Carol opens the project, she sees the padlock (🔒) glyph on the file icons, indicating that the files are checked out by someone else (see ["Viewing Source Control Status" on page 109](#)). Using the History view, she sees that Joe has the files checked out (see ["Working in the History View" on page 109](#)). She checks with Joe to see how long he plans to work with the files, then takes an early lunch.
- 3 Joe completes his work and checks in the files, associating them with the SBM issues that spawned the work (["Checking In Files" on page 115](#)).
- 4 Carol comes back from lunch and refreshes her source control status. She sees that the files are now checked in. She checks them out and begins her work.

## Setting Up Source Control Projects

Contents This section contains information about setting up the Version Manager rich integration to Eclipse-based IDEs.

**Prerequisites** Before proceeding, your administrator must use the Version Manager desktop client to create a project database that will contain the source control projects associated with the IDE project (if there isn't one already). See the *Administrator's Guide*.

## Excluding Files and Directories from Source Control

You can configure Eclipse to exclude specified files and directories from source control. This can minimize the size of the project database. For example, you could exclude all files with a .tmp file extension or exclude the bin directory and its contents.



**IMPORTANT!** Specify any files and directories you wish to exclude from source control before adding the project they are in to source control.



**CAUTION!** Some third-party tools create files or folders that should be excluded from source control (any folder that will not function correctly when RIDE (Rich Integrated Development Environment) creates a `_serena` folder under it). One such folder is the `ibmconfig` folder.

### To exclude specified files and directories from source control:

- 1 Select Window | Preferences. The Preferences dialog box appears.
- 2 Click the plus sign next to **Team** and select **Ignored Resources**.
- 3 Examine the **Ignore Patterns** list for the file type or directory you wish to exclude from source control. If it is not listed, do the following:
  - a Click the **Add** button. The Enter Ignore Pattern dialog box appears.
  - b Enter a pattern that defines a file type or directory to ignore. Use wildcards if needed:
    - Asterisks (\*) represent one or more characters.
    - Question marks (?) represent one character.
  - c Click **OK**.
- 4 In the **Ignore Patterns** list, ensure that a check mark appears next to each file type and directory you want to exclude from source control.
- 5 Click **OK**.

## Migrating Projects from the Previous Source Control Integration

If you have existing Eclipse projects under Version Manager source control and you wish to migrate them to the rich integration, do the following:

PVCS VM SCC  
integration

- 1 From the previous integration, check in any changes.



- 2 If you will be using the same IDE installation with the rich integration, you must remove the project from your local workspace:
  - a Right-click on the project and select **Delete** from the pop-up menu. The Confirm Project Delete prompt appears.



**CAUTION!** Say **No** to any prompt asking if you wish to delete from the repository.

- b Select the option to delete contents, and click **Yes**.
- c Confirm the deletion of any read-only resources by clicking **Yes To All**.

- Rich integration
- 3 From the new integration, select PVCS VM RIDE | Import Project from Version Manager. See ["Get Projects from Source Control" on page 106](#).
  - 4 If the project was from Eclipse 1 (WebSphere Studio Application Developer 4), check in the .project file. This takes the place of the .vcm\_meta file that was used with Eclipse 1.



**IMPORTANT!** If you migrate a project from a previous version of Eclipse, do not open it from the old version of the IDE again.



## Adding Projects to Source Control

If your IDE project has not yet been added to source control, you must do so in order to use the integration to Version Manager.



**IMPORTANT!** Specify any files you wish to exclude from source control before adding the project they are in to source control. See ["Excluding Files and Directories from Source Control" on page 103](#).

### To add projects to source control:

- 1 Right-click the project icon in the Package Explorer or Navigator. A pop-up menu appears.
- 2 Select Team | Share Project. The Share Project dialog box appears.
- 3 Select **PVCS VM RIDE** from the **Select a repository type** list.
- 4 Click the **Next** button. The Enter Login Information page of the Share Project wizard appears.
- 5 Do any of the following to specify a Version Manager project database:
  - Enter the path to the root project database directory, or select a recent project database from the drop-down list.
  - Click the **PDB** button  to browse for a project database.
  - Click the **File Servers** button  to choose from all project databases on the Version Manager file server, if a file server is defined in Version Manager. (If no Version Manager file server is defined, ignore this button.)
- 6 Enter your Version Manager user name and password, if one is required.



- 7 To persist your password across sessions, select the **Remember password** checkbox, else if a password is required you will have to login at the beginning of each session.

- 8 Click **Next**. The Select Repository Workspace page appears.

- 9 Select the Version Manager workspace that you will use for the project, then click **Next**.

Your choice of workspace will determine the default version and promotion group for the files. This choice does not affect your workfile location setting; working copies of all of your files will be saved to the IDE workspace. This list displays all public workspaces in the project, as well as all private workspaces to which you have access.

The Select Repository Project page appears.

- 10 Select the location in the project database where Version Manager will create a new project using the name of the IDE project. You can select a project, subproject, or the root of the project database.



**NOTE** To add a new project folder to the tree, right-click on the folder that will be the new folder's parent, and select **Create Project** from the resulting pop-up menu.

- 11 Click **Next**. The Select SBM Database page appears.



**NOTE** If you do not wish to connect to an SBM server, click **Next** and skip to [Step 14](#).

- 12 Enter the URL to an SBM server in the **Host** field or select a recent one from the **Host** drop-down list.



**TIP** Enter the name of the host system then tab to the next field. A default URL will be auto-entered for you in the following format:

`http://tt_server/tmtrack/tmtrack.dll`

Where *tt\_server* is the name of the SBM host.



**NOTE** To use a non-default port number (any port other than 80), append the port number to the server name. For example, if the port number is 89:

`http://tt_server:89/tmtrack/tmtrack.dll`



**NOTE** A specific SBM user privilege is required to run the integration to SBM. See the SBM documentation.

- 13 Enter your SBM user name and password and click **Next**. The Done page appears.
- 14 Review the configuration. To change any of the settings, click the **Back** button.
- 15 Click **Finish**.

## Connecting Additional Workstations to an Existing Source Control Project

Once an Eclipse project is under source control, you can open it from any workstation that has access to the Version Manager project database. There are two ways to connect additional workstations to a source control project:



**IMPORTANT!** In order to reconnect a project to source control, you must:



- 1 Delete the project from your Eclipse workspace.
- 2 Import the project from Version Manager. See ["Get Projects from Source Control" on page 106](#).

**WARNING!** This has absolutely nothing to do with the now obsolete **Offline Mode**. For information on working offline, see ["Working Offline" on page 99](#).

- Get the project from source control. See the next section.
- Export and import a Project Set file. See ["Export/Import a Project Set File" on page 107](#).


### Get Projects from Source Control

**To add existing controlled projects to your workspace:**

- 1 Select PVCS VM RIDE | Import Project from Version Manager. The Get Projects from PVCS VM RIDE wizard appears.
- 2 Do any of the following to specify a Version Manager project database:
  - Enter the path to the root project database directory, or select a recent project database from the drop-down list.
  - Click the **PDB** button  to browse for a project database.
  - Click the **File Servers** button  to choose from all project databases on the Version Manager file server, if a file server is defined in Version Manager. (If no Version Manager file server is defined, ignore this button.)
- 3 Enter your Version Manager user name and password, if one is required.
- 4 To persist your password across sessions, select the **Remember password** checkbox, else if a password is required you will have to login at the beginning of each session.
- 5 Click **Next**. The Select Repository Workspace page appears.
- 6 Select the Version Manager workspace that you will use for the project, then click **Next**.

Your choice of workspace will determine the default version and promotion group for the files. This choice does not affect your workfile location setting; working copies of all of your files will be saved to the IDE workspace. This list displays all public workspaces in the project, as well as all private workspaces to which you have access.

The Select Repository Project page appears.

- 7 Browse the project database folders to find and select the specific project that you want to open in Eclipse. Eclipse projects under Version Manager source control are indicated by folders with blue covers .
- 8 In the **Local Workspace Location** field, enter or browse to select the local workspace location, then click **Next**. The local workspace is the local work directory, where your working copies of the files will be stored.
- 9 The Select SBM Database page appears.



**NOTE** If you do not wish to connect to an SBM server, click **Next** and skip to [Step 12](#).

- 10 Enter the URL to an SBM server in the **Host** field or select a recent one from the **Host** drop-down list.



**TIP** Enter the name of the host system then tab to the next field. A default URL will be auto-entered for you in the following format:

`http://tt_server/tmtrack/tmtrack.dll`

Where *tt\_server* is the name of the SBM host.



**NOTE** To use a non-default port number (any port other than 80), append the port number to the server name. For example, if the port number is 89:

`http://tt_server:89/tmtrack/tmtrack.dll`

- 11 Enter your SBM user name and password and click **Next**. The Done page appears.
- 12 Review the configuration. To change any of the settings, click the **Back** button.
- 13 Click **Finish**.

### **Export/Import a Project Set File**

A Project Set file contains the path information needed to connect other workstations to an existing source control project.



**NOTE** The list navigation in the Export and Import dialog boxes varies depending upon which IDE you are using.

#### **To export a Project Set file:**

- 1 Select File | Export. The Export dialog box appears.
- 2 Select **Team Project Set** (or Team | Team Project Set) from the **Select an export destination** list.
- 3 Click the **Next** button.
- 4 Select the projects to export.
- 5 Select an export destination.

- 6 Click the **Finish** button. A \*.PSF file is created in the selected directory.
- 7 Distribute the \*.PSF file to each workstation or make it available from a network location.

**To import a Project Set file:**

- 1 Select File | Import. The Import dialog box appears.
- 2 Select **Team Project Set** (or Team | Team Project Set) from the **Select an import source** list.
- 3 Click the **Next** button.
- 4 Enter the path and file name of the \*.PSF file in the **File name** field or browse to select it.
- 5 Click the **Finish** button.

## Disconnecting Projects from Source Control

When you disconnect a project from source control, Version Manager does not delete the Version Manager archives; it simply removes the association between your local IDE project and the Version Manager archives.



**IMPORTANT!** In order to reconnect a project to source control, you must:

- 1 Delete the project from your Eclipse workspace.
- 2 Import the project from Version Manager. See ["Get Projects from Source Control" on page 106](#).

**WARNING!** This has absolutely nothing to do with the now obsolete **Offline Mode**. For information on working offline, see ["Working Offline" on page 99](#).

**To disconnect a project from source control:**

- 1 Right-click the project icon in the Package Explorer or Navigator. A Pop-up menu appears.
- 2 Select Team | Disconnect. A prompt appears to confirm that you wish to disconnect the project.
- 3 Click **Yes**.

## Using Source Control

Contents This section contains procedural information about viewing and editing files that are under source control.

### Viewing Connection Information

You can view the Version Manager and SBM connection information for a project.





**To view connection information:**

- 1 Right-click the project and select **Properties** from the resulting pop-up menu. The Properties dialog box appears.
- 2 Select **PVCS VM RIDE** in the left pane.

**Viewing Source Control Status**

Version Manager indicates status and revision information by displaying graphics and text next to the object icons in the Package Explorer and Navigator.

The following table lists the information that can be displayed.

Icon/Text	Description	Meaning
(1.0)	A number in parentheses	This is the revision you got or checked out from source control to your workspace.
[1.2]	A number in square brackets	This is the current tip revision in the source control repository if the tip is not the revision in your workspace.
	A gold cylinder	The object is under source control and it is checked in.
	A check mark	You have the object checked out.
	A padlock	Another user has the object checked out.
	An asterisk	The object has been modified locally and is out of sync with the repository.

To refresh source control status for:

- Selected objects, right-click on the objects in the Package Explorer or Navigator and select Team | Refresh Status from the resulting pop-up menu.
- All projects, select PVCS VM RIDE | Refresh All Status from the Eclipse menu bar.

**Working in the History View**

From the History view, you can:

- Assign, rename, and delete version labels
- View all assigned version labels
- Promote revisions to the next promotion group
- View the promotion group hierarchy
- View the check-in comment of each revision
- View the check-in date of each revision
- Compare revisions
- Get previous revisions from the repository

## Using the History View

### To view revision history:

- 1 In the Package Explorer or Navigator, select the file you want to view and right-click. A pop-up menu appears.
- 2 Select Team | History. The History view opens.



**TIP** You can also drag files from the Package Explorer or Navigator and drop them into the History view.

- 3 Do any of the following to change the display of the History view:



- Click the **Refresh** button to refresh the History view.



- Click the **Group by Revisions** button to organize the History view by revision. This view lists every revision.



- Click the **Group by Labels** button to organize the History view by version label. This view displays only revisions that have version labels.















- Click the **Group by Promotion Group** button to organize the History view by promotion group. This view displays only revisions that are assigned a promotion group. They are displayed as a leaves in the promotion model tree.



- Click the **Menu** button and select **Show Comment** and/or **Show Labels** to show or hide the Comment and Label panes of the History view. These panes provide an easy to read list of every version label and comment associated with the selected revision.

- 4 Do any of the following:

To ...	Select one of these modes ...	And ...
Compare a revision to the latest revision in the repository		Right-click on the revision and select <b>Compare revisions</b> . The revisions open in the compare editor.
Compare a revision to another revision		<ol style="list-style-type: none"> <li>1 Select the revisions you want to compare.</li> <li>2 Right-click on the revisions and select <b>Compare revisions</b>.</li> </ol> The revisions open in the compare editor.
View a revision		Right-click on the revision and select <b>View revision</b> . The revision opens in an editor tab.

To ...	Select one of these modes ...	And ...
Get a revision to your workspace	 	Right-click on the revision and select <b>Get revision</b> . Double-click the file in the Package Explorer or Navigator to open it in an editor tab.
Assign a version label to a revision	 	Right-click on the revision and select <b>Add label</b> . The Label dialog box appears. See <a href="#">"Assigning Version Labels" on page 111</a> .
Rename a version label		<ol style="list-style-type: none"> <li>1 Click on the version label name to enter edit mode.</li> <li>2 Type the new name.</li> <li>3 Press the ENTER key to save the new name and exit edit mode.</li> </ol>
Delete a version label		Click on the revision ( <b>not</b> on the label name) and press the DELETE key.
Promote a revision to the next promotion group	  	Right-click on the revision and select <b>Promote to next</b> .

## Assigning Version Labels

To assign a version label to:

- The latest revision of a single file or multiple files, see ["Labeling the Latest Revision" on page 111](#).
- A prior revision of a single file, see ["Labeling a Previous Revision" on page 111](#).

### Labeling the Latest Revision

To label the latest revision:

- 1 In the Package Explorer or Navigator, select the file, files, folders, or projects you want to label and right-click. A pop-up menu appears.
- 2 Select Team | Label. The Label dialog box appears. See ["Completing the Label Dialog Box" on page 112](#).

### Labeling a Previous Revision

To label a previous revision:

- 1 In the Package Explorer or Navigator, select the file you want to label and right-click. A pop-up menu appears.
- 2 Select Team | History. The History view opens.

- 3 Right-click on the revision you want to label and select **Add label**. The Label dialog box appears. See the next section.

### Completing the Label Dialog Box

- 1 Invoke the Label dialog box as described above. The Label dialog box appears.
  - 2 Enter a name for the version label in the **Version Label** field. Version labels can be up to 254 characters. You can use alpha, numeric, and special characters except for colons (:), asterisks (\*), plus signs (+), minus signs (-), and double-quotation marks (").
- Options
- 3 To modify label options, click the **Options** bar and change any of the following:
    - **Float label with tip:** Select **Yes** to always keep the label assigned to the latest revision of the file. Every time a new revision is checked in, the label will move (or float) to the latest revision.  
To keep the new label assigned to the revision you are assigning it to now, select **No**.
    - **If label exists:** Specify what to do if the label you are assigning is already assigned to a different revision:
      - **Prompt:** Asks what you want to do.
      - **Reassign:** Moves the label to the revision you selected.
      - **Do not Reassign:** Leaves the label where it is, rather than moving it to the revision you selected.



**TIP** To save these options as the new default, select the **Save Settings** check box. For more information on setting default options, see ["Setting Default Options" on page 131](#).

- 4 Click **Label**.



**NOTE** To delete version labels, see ["Working in the History View" on page 109](#).

## Getting Files

When you get files, read-only copies of the latest revisions are placed in the workfile location.

### To get revisions:

- 1 In the Package Explorer or Navigator, select the files you want to get and right-click. A pop-up menu appears.
- 2 Select Team | Get. The Get dialog box appears with a list of selected files. You can change your selection by selecting and deselecting the files in this list.



- Options **3** In the **If workfile is changed** option, specify what to do if your local workfile is modified:
- **Merge:** Merges the contents of the repository revision into your modified local workfile. This is the default, though you can define a new one.



**IMPORTANT!** If there are merge conflicts:

- An error will appear in the Console view.
- Your local workfile will remain as it was.

To get the file, you must resolve the merge conflict (see ["Comparing and Synchronizing Your Workspace with Source Control" on page 118](#)) or select **Overwrite** or **Leave as-is**.

- **Prompt:** Asks what you want to do. Select this option if you want to specify a different choice for some files.
- **Overwrite:** Replaces your modified local workfile with the revision from the repository.
- **Leave as-is:** Retains your modified local workfile.



**TIP** To save these options as the new default, select the **Save Settings** check box. For more information on setting default options, see ["Setting Default Options" on page 131](#).

- 4** Click **Get**.

## Checking Out Files

When you check out a file, the tip (latest) revision is locked and a writable workfile is created in the workfile location.



**NOTE** If a promotion model is in effect, you must set the tip (latest) revision to the lowest level promotion group in order to check out the files.

### To check out files:

- 1** In the Package Explorer or Navigator, select the files you want to check out and right-click. A pop-up menu appears.
- 2** Select Team | Checkout. The Checkout dialog box appears with a list of selected files. You can change your selection by selecting and deselecting files in this list.

- Options **3** In the **If workfile is changed** option, specify what to do if your local workfile is modified:
- **Merge:** Merges the contents of the latest repository revision into your modified local workfile. This is the default, though you can define a new one.



**IMPORTANT!** If there are merge conflicts:

- An error will appear in the Console view.
- The repository revision will not be locked.
- Your local workfile will remain as it was.

To check out the file, you must resolve the merge conflict (see ["Comparing and Synchronizing Your Workspace with Source Control" on page 118](#)) or select **Overwrite** or **Leave as-is**.

- **Prompt:** Asks what you want to do. Select this option if you want to specify a different choice for some files.
- **Overwrite:** Replaces your modified local workfile with the latest revision from the repository.
- **Leave as-is:** Locks the file in the repository but retains your modified local workfile.



**TIP** To save these options as the new default, select the **Save Settings** check box. For more information on setting default options, see ["Setting Default Options" on page 131](#).

- 4** Click **Checkout**. A check mark appears next to each file icon to indicate that the files are checked out.

## Undoing Checkout

When you undo a checkout, the archive is unlocked and no changes are checked into the archive.

### To undo a checkout:

- 1** In the Package Explorer or Navigator, select the files you want to unlock and right-click. A pop-up menu appears.
- 2** Select Team | Undo Checkout. The Undo Checkout dialog box appears with a list of selected files. You can change your selection by selecting and deselecting files in this list.

- Options **3** In the **After file has been unlocked** option, specify what to do with the local workfile:
- **Replace local file with latest revision:** Replaces your local workfile with a read-only copy of the latest revision from the repository.

- **Leave local workspace as-is:** Retains your local workfile as it is.



**TIP** To save these options as the new default, select the **Save Settings** check box. For more information on setting default options, see ["Setting Default Options" on page 131](#).

- 4 Click **Undo Checkout**.

## Checking In Files

**Purpose** When you check in your workfiles, Version Manager stores any changes you made as new revisions in the repository.

**Workflow** **Pessimistic Locking:** Your organization may require that you check out (lock) files before you edit them. This workflow ensures that only one person can work on a given file at a time. This is the way Version Manager has traditionally been used.

**Pragmatic Locking:** On the other hand, your organization may encourage a CVS-like workflow where anyone can work on any file at any time, and no one locks the files. This workflow uses the merge and synchronize features to resolve everyone's changes.

For more information on workflows, see ["Collaborative Process Overview" on page 99](#).

**Merging** By default, if other users have checked in changes since you last updated your local workspace from the repository, your changes and the contents of the latest revision will be merged. This ensures that no changes are lost when working in a Pragmatic Locking workflow. If your changes conflict with the changes already checked in by other users, you must resolve the conflicts in order to check in your changes. See ["Comparing and Synchronizing Your Workspace with Source Control" on page 118](#).



**IMPORTANT!** Checking in local changes does not synchronize renamed, moved, or deleted objects. You must synchronize your local workspace with the Version Manager repository in order to check in refactored and deleted objects. See ["Comparing and Synchronizing Your Workspace with Source Control" on page 118](#).

### To check in files:

- 1 In the Package Explorer or Navigator, select the files you want to check in and right-click. A pop-up menu appears.
- 2 Select Team | Check In. The Check In dialog box appears with a list of selected files. You can change your selection by selecting and deselecting the files in this list.
- 3 Enter a description of the changes you made in the **Description** field. The description will be applied to all selected files.



**TIP** To specify a unique description for each file, leave the **Comment** field blank. After you complete the Check In dialog box, the Description dialog box will appear for each file in turn.

You will not be prompted for a description for unmodified files. If you force the check-in of unmodified files, they will be given a description of "No Change."



**IMPORTANT!** If you click the **Cancel** button on the Description dialog, the files will remain locked--**even** if they were not locked to start with. To unlock the files without overwriting your modified local workfiles, do an Undo Checkout and select the **Leave local workspace as-is** option.

- SBM Associations    **4**    To associate SBM issues with the revisions you are checking in, expand the **SBM Associations** bar and select the issues you wish to associate.



**NOTE** Only activated issues are displayed in the Check In dialog box, and only activated issues can be associated with revisions. To activate SBM issues, see ["Associating Issues with Files" on page 130](#).

SBM issues include information about the revisions they are associated with, such as the check-in date, revision number, and Version Manager user ID. For more information, see ["Associating and Working on SBM Issues" on page 125](#).

- 5**    To deactivate the selected issues after check-in, select the **Deactivate selected issues after checkin** check box.
- Options    **6**    To modify check-in options, click the **Options** bar and change any of the following:

Option	Description
<b>If newer revision in repository</b>	<p>Specify what to do if the revision in the repository is newer than the revision your workfile is based on:</p> <ul style="list-style-type: none"> <li>■ <b>Merge:</b> Combines the contents of the latest repository revision and your local workfile into a new revision and checks it in. This is the out-of-box default.</li> </ul> <p><b>NOTE</b> If there are merge conflicts: an error will appear in the Console view, the repository revision will remain locked, and your local workfile will remain as it was. To check in the file, you must resolve the merge conflict (see <a href="#">"Comparing and Synchronizing Your Workspace with Source Control" on page 118</a>) or select <b>Force Checkin</b>.</p> <ul style="list-style-type: none"> <li>■ <b>Skip:</b> Do not check in the file.</li> <li>■ <b>Force Checkin:</b> Check in the file and create a new revision, without merging.</li> </ul>
<b>If workfile unchanged</b>	<p>Specify what to do if the workfile has not changed:</p> <ul style="list-style-type: none"> <li>■ <b>Prompt:</b> Asks what you want to do. Select this option if you want to specify a different choice for some files.</li> <li>■ <b>Checkin:</b> Checks in your local workfile as a new revision.</li> <li>■ <b>Leave as-is:</b> Does not check in your workfile. If the file is locked, it will be unlocked.</li> </ul>

Option	Description
<b>Use description for all</b>	<p>To apply the <b>Description</b> field to all of the files you are checking in, select <b>Yes</b>.</p> <p>To specify a unique description for each file, select <b>No</b>. The Description dialog box will appear for each file in turn.</p> <p><b>NOTE</b> You will not be prompted for a description for unmodified files. If you force the check-in of unmodified files, they will be given a description of "No Change."</p>
<b>Keep locked</b>	<p>To leave the files locked after check-in, select <b>Yes</b>. To leave the files unlocked after check-in, select <b>No</b>.</p> <p><b>IMPORTANT!</b> You cannot promote locked revisions. If you want to promote the new revisions, do not choose to keep the files locked.</p>
<b>New label</b>	<p>Enter a version label to assign to the new revision. Labels are limited to 254 characters. Do not use a colon (:), double quotes ("), a plus sign (+), or a minus sign (-).</p>
<b>Float label with tip</b>	<p>If you are assigning a label to the new revision, select <b>Yes</b> to always keep the label assigned to the latest revision of the file. Every time a new revision is checked in, the label will move (or float) to the latest revision.</p> <p>To keep the new label assigned to the revision you are checking in now, select <b>No</b>.</p>
<b>If label exists</b>	<p>Specify what to do if the label you are assigning to the new revision is already assigned to a different revision:</p> <ul style="list-style-type: none"> <li>■ <b>Prompt:</b> Asks what you want to do. Select this option if you want to specify a different choice for some files or labels.</li> <li>■ <b>Reassign:</b> Moves the label to the new revision.</li> <li>■ <b>Leave as-is:</b> Leaves the label where it is, rather than moving it to the new revision.</li> </ul>
<b>Promote to next</b>	<p>To promote the new revision to the next group in the promotion hierarchy, select <b>Yes</b>. To retain the current promotion group, select <b>No</b>.</p> <p><b>IMPORTANT!</b> You cannot promote locked revisions. If you want to promote the new revisions, do not choose to keep the files locked.</p>
<b>Get file on Keyword expansion</b>	<p>If the file you are checking in includes Version Manager keywords that will be expanded during check-in, selecting <b>Yes</b> will copy the latest revision, with the keywords expanded, to your local workspace.</p> <p>To retain unexpanded keywords in your workfile, select <b>No</b>.</p>



**TIP** To save these options as the new default, select the **Save Settings** check box. For more information on setting default options, see ["Setting Default Options" on page 131](#).

- 7 Click **Check In**.

## Using Rename or Move (Refactoring)

### Special Considerations

- Once you refactor a project, you must synchronize it with the repository before you can check in those changes.
- Refactoring may cause the archive name to differ from the workfile name--which is not compatible with the Command-Line Interface (CLI). Since the CLI is not project aware, it requires that the names match. However, you can use the Project Command-Line Interface (PCLI), instead. (This does not impact the Version Manager desktop client or the IDE client since they resolve workfile and archive names via the project metadata.)

### To use Rename or Move:

- 1 Open a perspective in which the Package Explorer is available, such as the Java Perspective.
- 2 Select the Package Explorer as the active pane.
- 3 Select the item you wish to rename or move and do one of the following:
  - Select Refactor | Rename (or Move).
  - Right-click and select Refactor | Rename (or Move).
- 4 You may be asked to confirm the action of checking in from a different location than the one to which the file is checked out. You must choose to proceed with this action.
- 5 For other users to work with the modified project, you must synchronize it with the Version Manager repository, and then they must get the updated project from source control or synchronize with source control. See ["Comparing and Synchronizing Your Workspace with Source Control" on page 118](#).

## Comparing and Synchronizing Your Workspace with Source Control

**Purpose** In a multi-user environment, you should synchronize your workspace with source control to:

- Remove files from your local workspace that other users have removed from the Version Manager repository.
- Add files to your local workspace that other users have added to the Version Manager repository.



**NOTE** If a default version is defined for the Version Manager project, adding or removing the default version label to or from a file will have the same effect as adding or removing the file to or from the repository.

- Add files to the Version Manager repository that you have added to your local workspace.
- Remove files from the Version Manager repository that you have deleted from your local workspace.
- Update the content of files in your local workspace that other users have modified and checked into the Version Manager repository.
- Update the content of files in the Version Manager repository that you have modified in your local workspace.



**TIP** While disconnected from the Version Manager repository, you can use the compare feature to see what changes you have made since you last synchronized with the repository. A handy feature, especially if you have already seen the in-flight movie.

**Overview** This process consists of several steps, not all of which may apply to your workflow or a given project on a given day:

- 1 ["Comparing Your Workspace with Source Control" on page 119](#). This shows what files have changed locally and/or in the repository since you last checked out or got files from the repository.
- 2 ["Comparing and Editing the Contents of Workfiles" on page 120](#). This shows the line-by-line differences between a workfile and the latest revision in the repository. It allows you to easily edit the workfile while referencing the latest repository revision and the common ancestor of the workfile and the repository revision.
- 3 ["Resolving Your Changes" on page 121](#). This step commits the incoming and outgoing changes to your local project and the Version Manager repository.

### **Comparing Your Workspace with Source Control**

This shows what files have changed locally and/or in the repository since you last checked out or got files from the repository.

#### **To compare your workspace with source control:**

- 1 In the Package Explorer or Navigator, select the projects and/or packages you want to compare to the repository, and right-click. A pop-up menu appears.



**TIP** To compare all projects in your workspace, select PVCS VM RIDE | Compare Workspaces from the Eclipse menu bar.

- 2 Select Team | Compare Workspaces. The Synchronize view appears.
- 3 Select a synchronize mode. This determines which commands are available and which changes are displayed. To display and operate on:



- Incoming changes and conflicts from the repository, select the **Incoming Mode** button.



- Outgoing changes and conflicts to the repository, select the **Outgoing Mode** button.










- Both incoming and outgoing changes and conflicts, select the **Incoming/Outgoing Mode** button.



- Only conflicts, select the **Conflicts Mode** button.

4 Differences between local objects and repository objects are denoted by the following:

Icon/Text	Description	Meaning
(1.0)	A number in parentheses	This is the revision you got or checked out from source control to your workspace.
[1.2]	A number in square brackets	This is the current tip revision in the Version Manager repository. It appears if the revision in your workspace is not the tip.
	Gray right arrow	Outgoing edit, rename, or move. The local workfile contains changes that are not in the repository.
	Gray right arrow with plus (+) sign	Outgoing addition. The local project contains a workfile that has not yet been added to the project in the repository.
	Gray right arrow with minus (-) sign	Outgoing deletion. The workfile has been deleted from the local project but is still in the project in the repository.
	Blue left arrow	Incoming edit, rename, or move. The file in the repository contains changes that are not in the local workfile.
	Blue left arrow with plus (+) sign	Incoming addition. The repository contains a file that has not yet been added to the project in the local workspace.
	Blue left arrow with minus (-) sign	Incoming deletion. The file has been deleted from the project in the repository but is still in the local project.
	Red double-ended arrow	Conflict. The local workfile and the repository file have both changed.



5 To retain the current Synchronize view, click the **Pin Current Synchronization** button. The next time you select Compare Workspaces, the pinned synchronization will be retained when the new synchronization opens.



6 To switch among your current Synchronize views, click the **Synchronize** drop-down button and select a synchronization from the resulting list.



7 To remove the current synchronization, click the **Menu** button and select **Remove Current Synchronization**.

### Comparing and Editing the Contents of Workfiles






The Compare Editor displays the line-by-line differences between your local workfile, the latest revision in the repository, and the common ancestor of the two. Use this view to edit your workfile and/or determine which changes to keep.

#### To compare a workfile with source control:

- Right-click the file in the Synchronize view and select **Open In Compare Editor** (or double-click the file). The local workfile and the latest revision in the repository open in the compare editor.



## 2 Do any of the following:

- 
    - To display the common ancestor of the workfile and repository revision, click the **Show Ancestor Pane** button.
  - 
    - To hide the common ancestor of the workfile and repository revision, click the **Two-Way Compare** button.
  - 
    - To copy all non-conflicting changes from the repository revision to the local workfile, click the **Copy All Non-Conflicting Changes from Right to Left** button.
  - 
    - To copy the currently selected change from the repository revision to the local workfile, click the **Copy Current Change from Right to Left** button.
  - 
    - To select a different change, click the **Select Next Change** or **Select Previous Change** buttons.
    - Copy content from the repository revision and the ancestor and paste it into the workfile.
    - Edit the contents of the workfile.
- 3 Once you are done making changes to the workfile, right-click in the left pane and select Save from the resulting pop-up menu (or press Ctrl+S).

## Resolving Your Changes


This step resolves the incoming and outgoing changes to your local project and the Version Manager repository.


### To resolve changes:

- 1 Select the appropriate synchronize mode.



**NOTE** The synchronize mode (  ) determines which commands are available and which changes are displayed and acted upon. See ["Comparing Your Workspace with Source Control" on page 119](#).

- 2 Select the object you wish to operate on. If you select a project or folder, the objects within it will be included in the operation if they contain the type of changes that would be affected by the operation.
- 

3 To view a change in the Compare Editor, right-click the change and select **Open in Compare Editor**, or double-click the file, or click the **Go to Next Difference** or **Go to Previous Difference** button. See ["Comparing and Editing the Contents of Workfiles" on page 120](#).
- 

4 To resolve changes, do any of the following:

  - To resolve the selected incoming and outgoing changes and conflicts, click the **Resolve All Changes** button. This button is available in all modes.



Outgoing changes will be checked into the repository, and incoming changes will be copied to your local workspace.

Conflicts will be auto-merged to your local workfile. They will then appear as outgoing changes. Click the **Resolve All Changes** button again to check the merged files into the repository.





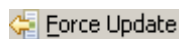
**NOTE** If merge conflicts are found (line-to-line conflicts), the conflicts will remain in the Synchronize view and an auto-merge failure will be noted in the Console view.



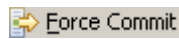
- To copy the selected incoming changes to your local workspace, select the **Update Workspace With Incoming Changes** button. This button is available only in Incoming () and Incoming/Outgoing () modes.




- To check in the selected outgoing changes, select the **Commit Outgoing Changes To Repository** button. This button is available only in Outgoing () and Incoming/Outgoing () modes.




- To overwrite your modified local workfile with the latest revision from the repository, right-click and select **Force Update**. This menu item is available only in Outgoing () mode.



- To check in your local workfile as the latest revision even though there is a newer revision in the repository, right-click and select **Force Commit**. This menu item is available only in Incoming () mode.



**CAUTION!** This in effect skips over the latest revision, leaving behind whatever changes it contained.

- **Mark as Merged:** This removes the conflict from the Synchronize view without making any changes to your local workspace or the repository. This menu item is available only in Conflict () mode.

You may wish to use this feature if you have already resolved the conflict by editing the workfile in the Compare Editor. See ["Comparing and Editing the Contents of Workfiles" on page 120](#).



**NOTE** Depending on your configuration and the resolutions chosen, any of the following may occur:

- If files will be checked in, the Commit dialog box will appear. You may enter a description for each file, or use the same for all.
- If the **Association required** option is enabled, any currently active SBM issues will be associated with any files that get checked in. The issues will remain active. If there are no active issues, a message will display in the Console view, and the check-in will fail.



- 5 To refresh the Synchronize view, click the **Synchronize** button.

## Comparing with the Latest Revision

You can edit your local workfile while comparing it to the latest revision in the repository.

### To compare a workfile with the latest repository revision:

- 1 In the Package Explorer or Navigator, select the file you want to compare and right-click. A pop-up menu appears.

- 2 Select Compare With | PVCS VM RIDE Revision. The local workfile and the latest revision in the repository open in the Compare Editor.

- 3 Do any of the following:



- To copy all non-conflicting changes from the repository revision to the local workfile, click the **Copy All Non-Conflicting Changes from Right to Left** button.



- To copy the currently selected change from the repository revision to the local workfile, click the **Copy Current Change from Right to Left** button.



- To select a different change, click the **Select Next Change** or **Select Previous Change** buttons.
- Copy content from the repository revision and paste it into the workfile.
- Edit the contents of the workfile.

- 4 Once you are done making changes to the workfile, right-click in the left pane and select Save from the resulting pop-up menu.



## Comparing with Local History

You can compare a workfile with a local history of the changes made to that workfile. A new entry is made in the local history each time you save changes to a file.



**NOTE** To configure how many entries are retained and for how long, select Window | Preferences then select **Local History** from under **Workbench**. For more information, see the Eclipse help.

### To compare with local history:






- 1 In the Package Explorer or Navigator, select the file you want to compare and right-click. A pop-up menu appears.
- 2 Select Compare With | Local History. The Compare with Local History dialog box appears.
- 3 Select a local history entry from the Local History of *filename* pane.
- 4 Use the **Select Next Change** (  ) and **Select Previous Change** (  ) buttons to step through the changes.
- 5 Click **OK** to exit the compare.

## Comparing Workfiles with Each Other

### To compare two workfiles with each other:

- 1 In the Package Explorer or Navigator, CTRL-click to select the two files you want to compare, then right-click. A pop-up menu appears.
- 2 Select Compare With | Each Other. The two workfiles open in the Compare Editor.

**3** Do any of the following:

-  ■ To copy all non-conflicting changes from the first workfile to the second workfile, click the **Copy All Non-Conflicting Changes from Left to Right** button.
  -  ■ To copy all non-conflicting changes from the second workfile to the first workfile, click the **Copy All Non-Conflicting Changes from Right to Left** button.
  -  ■ To copy the currently selected change from the first workfile to the second workfile, click the **Copy Current Change from Left to Right** button.
  -  ■ To copy the currently selected change from the second workfile to the first workfile, click the **Copy Current Change from Right to Left** button.
  -  ■ To select a different change, click the **Select Next Change** or **Select Previous Change** buttons.
  - Copy content from one workfile and paste it into the other workfile.
  - Edit the contents of the workfiles.
- 4** Once you are done making changes, click in the pane of the modified file and press CTRL+S.



## Replacing with Local History

You can replace a workfile with an entry from the local history of changes made to that workfile. A new entry is made in the local history each time you save changes to a file.



**NOTE** To configure how many entries are retained and for how long, select Window | Preferences then select **Local History** from under **Workbench**. For more information, see the Eclipse help.

### To replace with local history:

- 1** In the Package Explorer or Navigator, select the file you want to replace and right-click. A pop-up menu appears.
- 2** Select Replace With | Local History. The Replace from Local History dialog box appears.
- 3** Select a local history entry from the Local History of *filename* pane.
- 4** Use the **Select Next Change** () and **Select Previous Change** () buttons to step through the changes.
- 5** Do one of the following:
  - To replace the workfile with the selected history entry, click the **Replace** button.
  - To close the dialog box without replacing the workfile, click the **Cancel** button.

## Replacing with Latest Revision

You can replace your local workfile with the latest revision from the repository.



**NOTE** To replace multiple files with the latest revision, see ["Getting Files" on page 112](#).

### To replace a workfile with the latest repository revision:

- 1 In the Package Explorer or Navigator, select the file you want to replace and right-click. A pop-up menu appears.
- 2 Select Replace With | Latest PVCS VM RIDE Revision. The local workfile is overwritten with the latest revision from the repository.

## Associating and Working on SBM Issues

If your organization uses SBM to track development issues, such as defects and tasks, you can access your issues from within the Version Manager integration to Eclipse. You can submit and modify SBM issues from within Eclipse, and then associate issues with specific files. When you associate issues with files, the versioned file history is added to the issue.

See the following for detailed information on the SBM integration to Eclipse:

- ["Issue Management Workflow" on page 126](#)
- ["Setting Up Your IDE Folder" on page 127](#)
- ["Changing SBM Connection Information" on page 128](#)
- ["Displaying Reports and Issues" on page 128](#)
- ["Submitting and Modifying Issues" on page 129](#)
- ["Associating Issues with Files" on page 130](#)
- ["Issue Management Options" on page 137](#)



**NOTE** The SBM user privilege **Run System Reports** is required in order to use the integration to SBM, else the following error message will appear:

Error reading associations (No permission)

See the SBM Administrator Guide.

## Issue Management Workflow

The following table describes the issue management workflow in Eclipse. You must follow this workflow to successfully display issues and associate them with files.

Step	Description
<b>1</b>	<b>Set up IDE Folder</b> Before you can access issues from within Eclipse, you must set up your IDE folder in SBM. The IDE folder is a special system folder that enables you to display specific issues and listing reports from the rich integrations to Visual Studio and Eclipse. See <a href="#">"Setting Up Your IDE Folder" on page 127</a> .
<b>2</b>	<b>Define Integration Settings</b> In the Version Manager desktop client, you can customize settings that affect issue association in the rich integration to Eclipse, including: <ul style="list-style-type: none"> <li>■ Whether to apply a version label based on the issue number to all revisions associated with an SBM issue</li> <li>■ Whether to require issue associations on check-in</li> <li>■ Whether to automatically add notes about associated issues to the check-in comments of the associated revisions</li> </ul> See <a href="#">"Issue Management Options" on page 137</a> .
<b>3</b>	<b>Connect to the SBM Server</b> When adding a project to, or importing a project from, source control, you can specify an SBM server to use for issue management. If you did not specify an SBM server at that time, or if you need to change the SBM server connection information, See <a href="#">"Changing SBM Connection Information" on page 128</a> . When you connect to an SBM server, you login as a specific user. All issues and reports that are in your IDE folder in SBM are then visible from Eclipse.

Step	Description
4	<p><b>Review, modify, and submit issues</b></p> <p>From the Issues view, you can display all issues that are available via listing reports in your IDE folder, or that have been added directly to your IDE folder. For example, this may include specific reports that list only issues that are assigned to you.</p> <p>You can then modify these issues, and even submit new issues. See <a href="#">"Displaying Reports and Issues" on page 128</a> and <a href="#">"Submitting and Modifying Issues" on page 129</a>.</p>
5	<p><b>Associate issues with file revisions</b></p> <p>The SBM integration to Version Manager also allows you to <i>associate</i> issues with specific revisions of files. By default when you associate issues with files, a <b>Version Control History</b> section is added to the issues to track information about the file revisions, and a version label is assigned to the associated file revisions to identify the issue. See <a href="#">"Associating Issues with Files" on page 130</a>.</p> <p>To associate issues:</p> <ol style="list-style-type: none"> <li>1 <i>Activate the issue.</i> This places the issue in a queue of issues that you can optionally choose to associate with files during check-in.</li> <li>2 <i>Work on the files that are affected by the issue.</i> For example, you may need to edit specific source code files to resolve a problem described in a specific issue.</li> <li>3 <i>Check in the files.</i> When you check in, you have the option to <i>associate</i> the files with any (or all) of the currently activated issues. At check-in, you can choose specifically which issues the files will be associated with. If the files you are checking in effectively end your portion of the work to address the issues, you can also choose to remove the issues from the activated issues list.</li> </ol>

## Setting Up Your IDE Folder

If your user folder is not visible in the PVCS VM RIDE Issues tab of Eclipse (or the IDE folder is not visible in the **Favorites** list in SBM), then you must enable it.

### To enable your IDE folder:

- 1 Launch SBM (enter the URL in a browser).
- 2 Click the **User Profile** link (in older versions it is your user name) in the upper-right corner of the Web client. The Edit User Profile page appears.
- 3 Select the **Display** tab.
- 4 Select the **Auto Folder Items** option.
- 5 Click the **Save Profile** button.

The IDE folder (and other auto folders) will now appear in the **Favorites** list in the Web client, and the contents of the folder will appear in the PVCS VM RIDE Issues tab of Eclipse. You can now specify which issues you want to access from Eclipse by doing any of the following from the Web client:

- Add specific issues directly to your IDE folder.

- Add listing reports to your IDE folder.



**IMPORTANT!** You can access only specific issues and listing reports from within Eclipse. You cannot access other types of reports, or other types of items, such as URLs.

See the *SBM User's Guide* for details on setting up personal or favorites folders.

## Changing SBM Connection Information

When adding a project to, or importing a project from, source control, you can specify an SBM server to enable issue management. If you did not specify an SBM server at that time, or if you need to change the SBM server connection information, complete the following procedure. When you connect to an SBM server, you are logged in as a specific user. All issues and reports that are in your IDE folder in SBM are then visible from Eclipse.

### To connect to an SBM server:

- 1 Right-click the project and select **Properties** from the resulting pop-up menu. The Properties dialog box appears.
- 2 Select **PVCS VM RIDE** in the left pane.
- 3 In the SBM Information frame, click one of the browse buttons. The Change SBM Information dialog box appears.
- 4 Enter the URL to an SBM server in the **Host** field or select a recent one from the **Host** drop-down list.



**TIP** Enter the name of the host system then tab to the next field. A default URL will be auto-entered for you in the following format:

`http://tt_server/tmtrack/tmtrack.dll`

Where *tt\_server* is the name of the SBM host.



**NOTE** To use a non-default port number (any port other than 80), append the port number to the server name. For example, if the port number is 89:

`http://tt_server:89/tmtrack/tmtrack.dll`

- 5 Enter your SBM user name.
- 6 Click **Finish**.
- 7 Click **OK**.

## Displaying Reports and Issues

From the Issues view, you can display all issues that are available via listing reports in your IDE folder, or that have been added directly to your IDE folder. For example, this may include specific reports that list only the issues that are assigned to you.



**To display reports and issues:**


- 1 Select PVCS VM RIDE | Show Issues. The Issues view appears.
- 2 To review your issues:
  - Select your user name in the left pane to list any issues that have been added to your IDE folder.



- Expand your user name to display all reports that are available to you. Any listing reports that have been added to your IDE folder appear here. You can then click any of the reports to display issues.
- Select **Activated Issues** to display any currently activated issues. These are the issues that you will be able to associate with revisions during check-in.





**NOTE** The **Related Issues** list displays all issues that are associated with a particular file. See ["Associating Issues with Files" on page 130](#) for information on using this list.

- 3 To view the contents of an issue, select the issue and click the **View Issue** button .



## Submitting and Modifying Issues

Submit and modify SBM issues to track the status and details of the tasks that you are completing in Eclipse. You can submit new issues for tasks, defects, or other work that needs to be completed, or modify issues to provide input into your work assignments. Depending on the workflow for your organization, you may modify issues in order to move them to another state, for example if you have completed your portion of the task and need to mark it as ready to test.

**To submit an issue:**

- 1 Select PVCS VM RIDE | Issues. The Issues view appears.
- 2 Click the **Submit Issue** button . The Submit Issue tab appears. For more information, see the *SBM User's Guide* or click the **Help**  button to invoke help for this page.

**To modify an issue:**

- 1 Locate the issue you want to update. See ["Displaying Reports and Issues" on page 128](#).
- 2 Select the issue and click the **View Issue** button . The issue opens in a new tab.
- 3 Update the issue as needed. For more information, see the *SBM User's Guide* or click the **Help**  button to invoke help for this page.

## Associating Issues with Files

In addition to providing access to specific issues and reports, The SBM integration to Version Manager also allows you to *relate* issues with specific revisions of files. By default, when you relate an issue with a file:

- A Version Control History section is added to the issue, that tracks the:
  - Name of the associated file
  - Revision number of the associated revision
  - check-in date
  - User who checked in the revision
  - Change description that the user entered when checking in

For example, if Joe related an issue with a file called test.cs, something like the following might appear in the issue after check-in:

```

▣ Version Control History

/Application/app 1/test.cs
Revision 1.3 Checked In by Joe Manager 2/4/2005 3:54:49 AM
Revision 1.2 Checked Out 2/4/2005 3:54:49 AM
minor change
  
```




**IMPORTANT!** In order to see the Version Control History section in SBM issues, you must enable the Version Control History display option from the User Profile dialog box in SBM.


- A version label is assigned to the revision of the file that is associated with the issue. The version label includes information about the issue, such as the issue number.
- Information about the related issue(s) is added to the check-in comment for the new revision.


See ["Issue Management Options" on page 137](#) for information on setting issue association options.

### To associate issues:

- 1 Locate the issues that you will work on and eventually relate to file revisions. See ["Displaying Reports and Issues" on page 128](#).
- 2 Select the issue and click the **Add to Activated Issues** button . The issue is added to your **Activated Issues** list.



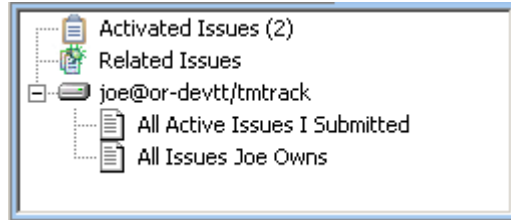
**TIP** To remove an issue from the Activated Issues list, select the issue in the list and click the **Remove from Activated Issues** button .

- 3 At any point, you can review the details of an issue by selecting it and clicking the **View Issue** button .
- 4 Complete the work required to resolve the issue, or your portion of it.
- 5 Check in the file or files that resolve the issue. On the Check In dialog box, under SBM Associations, select the issue that you want to associate with the file or files. Only

issues that are currently activated can be related during check-in. See ["Checking In Files" on page 115](#).

### To display all issues associated with a particular file:

In the Package Explorer or Navigator, right-click the file and select Team | Related Issues. The Issues view appears with the Related Issues list selected in the left pane.



All issues associated with the selected file, appear in the right pane of the Issues view.

Associated TeamTrack Item	File Name	Initiating Revision	Closing Revision	Log Message	Associated Item Table
BUG00107: Sound effect is too loud	/CAD-ECR-1/p1/C3.java	1.3	1.4	For Bob in Denver.	Issues
BUG00108: Icons are too small	/CAD-ECR-1/p1/C3.java	1.3	1.4	For Bob in Denver.	Issues

## Setting Default Options

You can configure the behavior of the source control integration to Version Manager and the issue management integration to SBM. See ["Source Control Options" on page 131](#) and ["Issue Management Options" on page 137](#).

### Source Control Options

You can configure the behavior of the source control dialogs (["Setting Local Source Control Options" on page 131](#)), the status glyphs (["Configuring Icon Glyphs" on page 135](#)), the colors used for console output (["Configuring Console Text Colors" on page 135](#)), and the Synchronize view (["Setting Local Synchronization Options" on page 136](#), and ["Scheduling Synchronization Updates" on page 137](#)).

#### Setting Local Source Control Options

You can configure the default behavior of the source control integration.

#### To configure source control:

- 1 Select Window | Preferences. The Preferences dialog box appears.
- 2 Expand the following nodes in the Preferences tree: Team | PVCS VM RIDE | Versioning.



**IMPORTANT!** If the Team | PVCS VM RIDE node does not appear in the Preferences dialog box, you must enable Core Team Support. In the left pane, select Workbench | Capabilities (or General Capabilities). In the right pane, select Team | Core Team Support (or click the Advanced button and select Team | Team Core Support from the Advanced dialog box).

Assign Label **3** Click **Assign Label** and change any of the following:

Option	Description
<b>Float label with tip</b>	<p>Select <b>Yes</b> to always keep the label assigned to the latest revision of the file. Every time a new revision is checked in, the label will move (or float) to the latest revision.</p> <p>To keep the label assigned to the revision you assign it to, select <b>No</b>.</p>
<b>If label exists</b>	<p>Specify what to do if the label you are assigning to the revision is already assigned to a different revision:</p> <ul style="list-style-type: none"> <li>■ <b>Prompt:</b> Asks what you want to do.</li> <li>■ <b>Reassign:</b> Moves the label to the revision you selected.</li> <li>■ <b>Do not Reassign:</b> Leaves the label where it is, rather than moving it to the revision you selected.</li> </ul>

CheckIn **4** Click **CheckIn** and change any of the following:

Option	Description
<b>If newer revision in repository</b>	<p>Specify what to do if the revision in the repository is newer than the revision your workfile is based on:</p> <ul style="list-style-type: none"> <li>■ <b>Merge:</b> Combines the contents of the latest repository revision and your local workfile into a new revision and checks it in. This is the out-of-box default.</li> </ul> <p><b>NOTE</b> If there are merge conflicts: an error will appear in the Console view, the repository revision will remain locked, and your local workfile will remain as it was. To check in the file, you must resolve the merge conflict (see <a href="#">"Comparing and Synchronizing Your Workspace with Source Control" on page 118</a>) or select <b>Force Checkin</b>.</p> <ul style="list-style-type: none"> <li>■ <b>Skip:</b> Do not check in the file.</li> <li>■ <b>Force Checkin:</b> Check in the file and create a new revision, without merging.</li> </ul>
<b>If workfile unchanged</b>	<p>Specify what to do if the workfile has not changed:</p> <ul style="list-style-type: none"> <li>■ <b>Prompt:</b> Asks what you want to do. Select this option if you want to specify a different choice for some files.</li> <li>■ <b>Checkin:</b> Checks in your local workfile as a new revision.</li> <li>■ <b>Leave as-is:</b> Does not check in your workfile. If the file is locked, it will be unlocked.</li> </ul>

Option	Description
<b>Use description for all</b>	<p>To apply the <b>Description</b> field to all of the files you are checking in, select <b>Yes</b>.</p> <p>To specify a unique description for each file, select <b>No</b>. The Description dialog box will appear for each file in turn.</p> <p>Note, you will not be prompted for a description for unmodified files. Unmodified files will be given a description of "No Change."</p>
<b>Keep locked</b>	<p>To leave the files locked after check-in, select <b>Yes</b>. To leave the files unlocked after check-in, select <b>No</b>.</p> <p><b>IMPORTANT!</b> You cannot promote locked revisions. If you want to promote the new revisions, do not choose to keep the files locked.</p>
<b>New label</b>	<p>Enter a version label to assign to the new revision. Labels are limited to 254 characters. Do not use a colon (:), double quotes ("), a plus sign (+), or a minus sign (-).</p>
<b>Float label with tip</b>	<p>Select <b>Yes</b> to always keep the label assigned to the latest revision of the file. Every time a new revision is checked in, the label will move (or float) to the latest revision.</p> <p>To keep the label assigned to the revision you assigned it to, select <b>No</b>.</p>
<b>If label exists</b>	<p>Specify what to do if the label you are assigning to the new revision is already assigned to a different revision:</p> <ul style="list-style-type: none"> <li>■ <b>Prompt:</b> Asks what you want to do. Select this option if you want to specify a different choice for some files or labels.</li> <li>■ <b>Reassign:</b> Moves the label to the new revision.</li> <li>■ <b>Leave as-is:</b> Leaves the label where it is, rather than moving it to the new revision.</li> </ul>
<b>Promote to next</b>	<p>To promote the new revision to the next group in the promotion hierarchy, select <b>Yes</b>.</p> <p>To retain the current promotion group, select <b>No</b>.</p> <p><b>IMPORTANT!</b> You cannot promote locked revisions. If you want to promote the new revisions, do not choose to keep the files locked.</p>
<b>Get file on Keyword expansion</b>	<p>If the file you are checking in includes Version Manager keywords that will be expanded during check-in, selecting <b>Yes</b> will copy the latest revision, with the keywords expanded, to your local workspace.</p> <p>To retain unexpanded keywords in your workfile, select <b>No</b>.</p>

Checkout **5** Click **Checkout** to specify what to do if your local workfile is modified:

- **Merge:** Merges the contents of the latest repository revision into your modified local workfile. This is the out-of-box default.



**IMPORTANT!** If there are merge conflicts:

- An error will appear in the Console view.
- The repository revision will not be locked.
- Your local workfile will remain as it was.

To check out the file, you must resolve the merge conflict (see ["Comparing and Synchronizing Your Workspace with Source Control" on page 118](#)) or select **Overwrite** or **Leave as-is**.

- **Prompt:** Asks what you want to do. Select this option if you want to specify a different choice for some files.
- **Overwrite:** Replaces your modified local workfile with the latest revision from the repository.
- **Leave as-is:** Locks the file in the repository but retains your modified local workfile

Get **6** Click **Get** to specify what to do if your local workfile is modified:

- **Merge:** Merges the contents of the repository revision into your modified local workfile. This is the out-of-box default.



**IMPORTANT!** If there are merge conflicts:

- An error will appear in the Console view.
- Your local workfile will remain as it was.

To Get the file, you must resolve the merge conflict (see ["Comparing and Synchronizing Your Workspace with Source Control" on page 118](#)) or select **Overwrite** or **Leave as-is**.

- **Prompt:** Asks what you want to do. Select this option if you want to specify a different choice for some files.
- **Overwrite:** Replaces your modified local workfile with the revision from the repository.
- **Leave as-is:** Retains your modified local workfile.

Undo Checkout **7** Click **Undo Checkout** to specify what to do with the local workfile:

- **Replace local file with latest revision:** Replaces your local workfile with a read-only copy of the latest revision from the repository.
- **Leave local workspace as-is:** Retains your local workfile as it is.

### **Configuring Client/Server-Side Processing**

By default, server-side processing is enabled. In most cases, this allows faster operation since less data needs to be transferred back and forth between the Version Manager File Server and your IDE client. However, you can revert to the traditional client-side

processing if your use case requires it. For instance, this feature does not currently support client-side event triggers.

### To configure Client/Server-Side processing:

- 1 Select Window | Preferences. The Preferences dialog box appears.
- 2 Select the following node in the Preferences tree: Team | PVCS VM RIDE | Client/Server.



**IMPORTANT!** If the Team | PVCS VM RIDE node does not appear in the Preferences dialog box, you must enable Core Team Support. In the left pane, select Workbench | Capabilities (or General Capabilities). In the right pane, select Team | Core Team Support (or click the Advanced button and select Team | Team Core Support from the Advanced dialog box).

- 3 Select or deselect the **Enable server-side processing** checkbox.
- 4 Click **Apply** or **OK**.

### Configuring Icon Glyphs

By default, icon glyphs show the status of an individual item. In the case of a Package or Project, the icon reflects the actual status of the object rather than that of any objects it may contain. You can configure icon glyphs so that objects, such as Packages and Projects, reflect the status of the objects they contain. However, this additional information comes at the cost of some performance.

### To configure icon glyphs:

- 1 Select Window | Preferences. The Preferences dialog box appears.
- 2 Select the following node in the Preferences tree: Team | PVCS VM RIDE | Label Decorations.



**IMPORTANT!** If the Team | PVCS VM RIDE node does not appear in the Preferences dialog box, you must enable Core Team Support. In the left pane, select Workbench | Capabilities (or General Capabilities). In the right pane, select Team | Core Team Support (or click the Advanced button and select Team | Team Core Support from the Advanced dialog box).

- 3 Select the **Compute deep folder status** check box to cause objects such as Packages and Projects to display the status of the objects they contain. Or de-select the check box to disable this feature.



**NOTE** Enabling **Compute deep folder status** will slow performance.

- 4 Click **OK**.

### Configuring Console Text Colors

You can choose which colors to use for various types of output into the Console view.

**To configure console colors:**

- 1 Select Window | Preferences. The Preferences dialog box appears.
- 2 Select the following node in the Preferences tree: Team | PVCS VM RIDE | Console.



**IMPORTANT!** If the Team | PVCS VM RIDE node does not appear in the Preferences dialog box, you must enable Core Team Support. In the left pane, select Workbench | Capabilities (or General Capabilities). In the right pane, select Team | Core Team Support (or click the Advanced button and select Team | Team Core Support from the Advanced dialog box).

- 3 Click on the colored button next to a text type to select a new color for that type. You can select colors for the following types of console text:
  - Commands (default is black)
  - Messages (default is blue)
  - Errors (default is red)
  - Warnings (default is gold)
- 4 Click **Apply**.

**Setting Local Synchronization Options**

You can configure the default behavior of the Synchronize view.

**To set synchronization options:**

- 1 Select the **Menu** button in the Synchronize view. A pop-up menu appears.
- 2 Select **Preferences**. The Synchronize Preferences dialog box appears.
- 3 To change whether folders are expanded or collapsed when the Synchronize view opens, select or deselect the **Use Compressed Folders as default Synchronize View layout** check box (or the **Compressed Folders** radio button).
- 4 To display descriptive text for each change type, select the **Show all synchronization information in a resource's text label** check box. To specify whether a different perspective is opened when you synchronize, select one of the following options:
  - **Always:** The perspective specified in the **Perspective** field will open when you synchronize.
  - **Never:** You will remain in whatever perspective you were in at the time you invoked the synchronization.
  - **Prompt:** A dialog will appear asking if you would like to switch to the perspective specified in the **Perspective** field.
- 5 To specify the perspective to open when you synchronize, select one from the **Perspective** field.
- 6 Click **OK**.





## Scheduling Synchronization Updates

You can schedule automatic refreshes of a synchronization. This updates the information displayed in the Synchronize view; it does not commit or resolve changes.

### To schedule synchronization updates:

- 1 Open the synchronization you want to schedule.



**TIP** To switch among pinned (  ) Synchronize views, click the **Synchronize** drop-down (  ) button and select a synchronization from the resulting list.



- 2 Select the **Menu** button in the Synchronize view. A pop-up menu appears.
- 3 Select **Schedule**. The Configure Synchronize Schedule dialog box appears.
- 4 To enable automatic updates for the current synchronization, select the **Using the following schedule** option.
- 5 To specify a time interval between updates, select **hour(s)** or **minute(s)** and enter the value you desire.
- 6 Click **OK**.

## Issue Management Options

You can configure aspects of the global and local behavior of the integration to SBM. See ["Setting Global Issue Management Options" on page 137](#) and ["Setting Local Issue Management Options" on page 138](#).

### Setting Global Issue Management Options

From the Version Manager desktop client, the administrator can define settings for rich IDE integrations to SBM, including:

- Whether to apply a version label with the issue number to all associated revisions
- Whether to require issue associations on check-in
- Whether to automatically add notes about associated issues to the check-in comments for new revisions

### To define integration settings:

- 1 Select the Project Database to which you will apply the settings.
- 2 From the Version Manager desktop client, select Admin | SourceBridge settings. The SourceBridge Settings dialog box appears.

- 3 Set the following options:

Field	Description
<b>Show Issue association dialog on checkin / Association required</b>	Select to require that issues be associated with files at check-in. If you select this option, users will be unable to complete checkins if no issues are currently active. The <b>Show Issue association dialog on checkin</b> option has no effect within the rich integration to Eclipse, but you must select it in order to be able to select the <b>Association required</b> option.
<b>Tag workfile comment with association</b>	<p>Select to add information about the associated issue(s) to the check-in comments for files as they are checked in. Select <b>Before existing comment</b> or <b>After existing comment</b> to determine the placement of this information within the comment.</p> <p>In the <b>Tag</b> field, enter the text that you want to add to the check-in comments. This can include any of a number of keywords that will automatically enter information about the associated issues. These include:</p> <ul style="list-style-type: none"> <li>■ \$id -- Expands to the issue ID number</li> <li>■ \$ownid -- Expands to the user ID of the issue owner</li> <li>■ \$owner -- Expands to the name of the issue owner</li> <li>■ \$project -- Expands to the name of the current project</li> <li>■ \$title -- Expands to the title of the issue</li> </ul>
<b>Use version labels on checkin</b>	Select to apply a version label consisting of the issue number when checking in a file.

- 4 Click **OK**.

### Setting Local Issue Management Options

#### To configure issue management:

- 1 Select Window | Preferences. The Preferences dialog box appears.
- 2 Select the following node in the Preferences tree: Team | PVCS VM RIDE | Issuing | Associations.



**IMPORTANT!** If the Team | PVCS VM RIDE node does not appear in the Preferences dialog box, you must enable Core Team Support. In the left pane, select Workbench | Capabilities. In the right pane, select Team | Core Team Support.

- 3 Select or deselect the **Deactivate selected issues after checkin** check box. By default, issues are not deactivated after check-in.
- 4 Click **OK**.

## Chapter 9

# Visual Studio SCC Integration

---

Introduction	140
Accessing Supported Features	140
About Visual Basic Files	141
Setting Up Source Control Projects	142
Using Source Control	146

# Introduction

**Purpose** This chapter has four purposes:

- List the Version Manager features available through Microsoft® Visual Studio, and provide a quick reference to accessing those features



**NOTE** This chapter applies to the SCC integration (**SCC/COM IDE Client**). If you installed the rich integration to Visual Studio, see [Chapter 10, "Visual Studio Rich Integration" on page 149](#).

- Note any features described in Part 1 of this manual that do not apply to this IDE
- Help you set up source control projects and add files to source control
- Help you access files that are under source control from within the IDE

**For more information** See [Part 1, "The Version Manager IDE Client," on page 9](#) for information about:

- Source control concepts
- Source control defaults
- Advanced source control features

## Accessing Supported Features

**What is supported?** Visual Studio supports the full set of source control features available through the Version Manager IDE client. See the following table.

To...	Select...	For more information see...
Get revisions	File   Source Control   Get	<a href="#">"Getting Files" on page 146</a>
Open a project from source control	File   Source Control   Open From Source Control	<a href="#">"Connecting Additional Workstations to a Source Control Project" on page 145</a>
Check out revisions	File   Source Control   Check Out	<a href="#">"Checking Out Files" on page 146</a>
Undo checkout of revisions	File   Source Control   Undo Checkout	<a href="#">"Undoing Checkout" on page 147</a>
Check in revisions	File   Source Control   Check In	<a href="#">"Checking In Files" on page 148</a>
Manage version labels	File   Source Control   Serena Source Control	<a href="#">"About Version Labels" on page 34</a>
View properties of revisions or archives	File   Source Control   Serena Properties	<a href="#">"About Properties" on page 46</a>
Monitor source control activity	File   Source Control   Serena Source Control	<a href="#">"Monitoring Source Control Activity with Pulse" on page 47</a>
Generate a history report	File   Source Control   History	<a href="#">"About History Reports" on page 50</a>
Generate a difference report	File   Source Control   Compare Versions	<a href="#">"About Difference Reports" on page 52</a>

To...	Select...	For more information see...
Access the Version Manager Options dialog	File   Source Control   Serena Source Control	<a href="#">"About Setting Defaults for Version Manager Options" on page 22</a>
Add non-web solutions to source control	File   Source Control   Add Solution to Source Control	<a href="#">"Adding Visual Studio Files to Source Control" on page 143</a>
Add web and non-web projects to source control	File   Source Control   Add Selected Projects to Source Control	<a href="#">"Adding Visual Studio Files to Source Control" on page 143</a>
Exclude or remove files from source control	File   Source Control   Exclude Selection from Source Control	<a href="#">"Excluding or Removing Files from Source Control" on page 143</a>
Set SCC options for Visual Studio	Tools   Options	<a href="#">"Configuring Source Control Behavior" on page 142</a>
Share an archive across projects	File   Source Control   Share	<a href="#">"About Sharing Files Across Projects" on page 25</a>

## About Visual Basic Files

- Binary and text files** Visual Basic projects include text files that are paired with binary files, but normally the Visual Studio interface does not display the binary files. Though the binary files remain unseen while you work in the IDE, they are kept in sync with the source control actions you apply to the visible files. For example, if you check out the file `Form2.vb`, the binary file `Form2.resx` is also checked out.
- Revision numbers out of sync** However, it is possible for the revision numbers of the files to get out of sync. This occurs because some programming changes that affect text files do not affect binary files. By default when you check in your project after such a change, both files are checked in even if no changes were made to the binary file. However, if you change the default check-in behavior, only the text file will be checked into a new revision and the files will be out of sync.



### IMPORTANT!

- By default, the IDE client will check in a file even if it is unmodified or older than the latest revision. You should not change this default. If the default has been changed, see ["Setting Defaults" on page 22](#) for information on setting it back.
- If you use the Version Manager desktop client to apply source control actions to Visual Basic files, be sure to select both the text files and the binary files or they will become out of sync. You must also choose to check in files that are unchanged.

- TrackerLink users** If you use TrackerLink to associate a Visual Basic file pair with an SCR, it is possible that only the text file will receive the resolution description from Tracker. In this case, the binary file will receive the default change description. This occurs because some programming changes that affect text files do not affect binary files.



**NOTE** To view binary files, click the **Show All Files** button in the Solution Explorer.

## Setting Up Source Control Projects

Contents	This section contains information about setting up the Version Manager IDE client to work with Visual Studio.
Prerequisites	Before proceeding, you must do the following: <ul style="list-style-type: none"><li>■ Select Version Manager as your source control provider (if you have multiple IDE clients installed). See <a href="#">"Selecting an SCC Provider" on page 19</a>.</li><li>■ Use the Version Manager desktop client to create a project database that will contain the source control projects associated with the Visual Studio .NET projects (if you don't already have one).</li></ul>
For more information	See <a href="#">Chapter 2, "Setting Up Source Control with SCC IDEs" on page 17</a> .

### Upgrading to Visual Studio 2005 from Visual Studio .NET 2003

When you open a Visual Studio .NET 2003 solution in Visual Studio 2005, the Visual Studio Conversion Wizard appears. This Microsoft wizard will convert your solutions to the new Visual Studio 2005 format. Once the conversion is complete, you can continue to use the SCC (Source Code Control) integration to Version Manager.



**IMPORTANT!** If you want to migrate to the Rich Version Manager integration, see [Chapter 10, "Visual Studio Rich Integration" on page 149](#).

### Configuring Source Control Behavior

You can configure Visual Studio so that actions you take in Visual Studio:

- Automatically invoke source control operations
- Do not invoke source control operations
- Prompt for permission to invoke source control operations

#### To configure the source control behavior of Visual Studio:

- 1 Select Tools | Options. The Options dialog box appears.
- 2 Select the Source Control folder in the left pane.
- 3 Select or deselect the features you wish to activate or deactivate. For detailed information on each feature, see the Visual Studio documentation.
- 4 Click **OK**.

### Configuring Web Projects

By default, Visual Studio is configured to use the SCC interface (File share) for both web and non-web projects. This is the recommended configuration, but may be unfamiliar to

you since Visual InterDev 6 used the COM interface (FrontPage Extensions) for web projects.

#### To configure web project access:

- 1 Select Tools | Options. The Options dialog box appears.
- 2 Open the **Projects** folder and select **Web Settings**.
- 3 Select the **File share** option.
- 4 Click **OK**.

For more information on working with web projects, see the Visual Studio documentation.

## Excluding or Removing Files from Source Control

You can use the Exclude from Source Control feature in two ways:

- Used before a project is under source control, it will prevent Version Manager from creating archive files for the selected workspace files.
- Used after a project is under source control, it will retain the existing archive files but will prevent the addition of new revisions until the exclusion is removed.

#### To exclude files from source control:

- 1 Select the files in the Project Explorer pane.
- 2 Select File | Source Control | Exclude Selection from Source Control.



**NOTE** If the project file is not already checked out, you are prompted to check it out.

A red circle (🚫) appears to the left of each file icon, if the project is under source control.



**NOTE** To remove an exclusion, repeat the procedure shown above.

## Adding Visual Studio Files to Source Control

#### To add files to source control:

- 1 Do one of the following:
  - To add a web project to source control:
    - a Select the project in the Solution Explorer pane.
    - b Select Add Selected Projects to Source Control.



**IMPORTANT!** Do not add web solutions to source control. Select the web project instead.

- To add a non-web solution and the projects within it to source control, select File | Source Control | Add Solution to Source Control.
- To add a project to source control without placing the solution under source control:
  - a Right-click the project in the Solution Explorer pane. A pop-up menu appears.
  - b Select Add Project to Source Control.
- To add a new project to source control when the solution is already under source control:
  - a Create the new project in a directory below the initial project in the solution. The files of the new project appear in the Pending Checkins pane.
  - b Click the **Check In** button.



**IMPORTANT!** To add a project or file to an existing solution or project that is already under source control, you must first check out the solution or project.

- 2 If you are adding a web project, the following dialog may appear. Click the **Continue** button. For more information, see ["Configuring Web Projects" on page 142](#).
- 3 Select Tools | Serena | Add Project to Serena. The Add Project to Source Control dialog box appears.
- 4 The default project database displays under **Source Control Project**. If you wish to add the files to a different project database, click the **Open Database** button. The Select Project Database dialog box appears.

Do one of the following:

- To open a project database located on a listed Version Manager File Server, select it from the **Project Databases** list and click **OK**.

To add a Version Manager File Server to the list, double-click an empty cell in the **File Server** list and type the name of the system that hosts the file server. Once you press the ENTER key, the entry will be auto-completed (`http://SystemName:8080/serenafs/FileServer`) and tested. To specify an https server or a non-standard port, double-click on the auto-completed entry, edit it appropriately, and press the ENTER key.

- To open a project database located on your local file system or the network, click the **Browse** button and navigate to the location via the resulting dialog box.

- 5 Do one of the following:

- To add to an existing source control project, select one from under **Source Control Project** and click **OK**. Proceed to [Step 6](#).
- **To create a new Version Manager project:**
  - a Under Select Source Control Project, select the location in the project database where you want to create the new project.
  - b Click the **Create Project** button. The Create Source Control Project dialog box appears.



The Project Database Information group displays the name and location of the current project database, and the location of the new project within the database.

- c By default, the new Version Manager project uses the same name as the IDE project. If necessary, enter a different name in the **Project Name** field.

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 (< >).

- d Click **OK**. The Add Project to Source Control dialog box reappears with the new project under **Source Control Project**.

6 Click **OK**. The Change Description dialog box appears.

7 Do any of the following:

- Enter a description of the files in the **Description** field.
- To use the same description for every file, select the **Use description for all** checkbox.
- To use a unique description for each file, do *not* select the **Use description for all** checkbox. The Change Description dialog box will appear for each file in turn.



**NOTE** Visual Basic:

You will be prompted to enter descriptions for both the text files and the binary files in Visual Basic projects, though Visual Studio normally hides the binary files from view.

- 8 Click **OK**. A blue padlock (🔒) appears to the left of each file icon to indicate that the files are checked in to source control.

Sharing files

If an archive will be shared by multiple Version Manager projects, see ["About Sharing Files Across Projects" on page 25](#) for more information.

## Connecting Additional Workstations to a Source Control Project

After a project has been created and added to source control, each developer can access it by opening it from source control, which creates a copy of the project on the developer's system. Once a developer has a local copy of a project, he can perform normal source control operations on it.

### To open a solution or project from source control:

- 1 Select File | Source Control | Open From Source Control. The Get Project from Source Control dialog box appears.

The default database appears under Source Control Project. If the database containing the project you wish to get is not displayed, click the **Open Database** button and browse to select it.

- 2 Select the project you wish to get from the project database.
- 3 Enter a location for all of your project files in the **Workfile Location** field, or click the browse button to select a location.

- 4 Click **OK**. The Open Solution dialog box appears.
- 5 Select the solution or project file you want to open and click **Open**.


## Using Source Control

Contents	This section contains procedural information about viewing and editing files that are under source control.
For more information	See <a href="#">Chapter 3, "Using Source Control" on page 27</a> .

### Getting Files

When you get files, read-only copies of the selected revisions are placed in the workfile location.

#### To get files:

- 1 Select the solution, project, or files you want to get in the Solution Explorer.
- 2 Select File | Source Control | Get. The Get dialog box appears.
- 3 Do any of the following:
  - Select or deselect files in the **Name** list.
  - To invoke the Merge Tool, select a file and click the **Compare Versions** button. For information on differencing, see ["Generating Difference Reports" on page 52](#).
- 4 Do one of the following:
  - To override the default get options, click the **Options** button (). The Advanced Get dialog box appears. (For information on advanced options, see ["About Getting Files" on page 28](#).)
  - To accept the default get options, click **OK**. The selected revisions are copied and read-only workfiles are placed in the workfile location.

Use advanced options

Accept defaults



**NOTE** To get files without invoking the Get dialog box, select File | Source Control | Get Latest Version.

### Checking Out Files

When you check out a file, the revision is locked and a writable workfile is created in the workfile location.

**To check out files:**

- 1 In the Solution Explorer pane, select the files, projects, or solution you want to check out and right-click. A pop-up menu appears.



**IMPORTANT!** Check out the solution and project files if they are under source control and your changes will affect them. Otherwise, your changes may be lost.

- 2 Select Check Out. The Check Out dialog box appears.

- 3 Do any of the following:


- Select or deselect files in the **Name** list.




- To invoke the Merge Tool, select a file and click the **Compare Versions** button. For information on differencing, see ["Generating Difference Reports" on page 52](#).

- 4 Do one of the following:

Use advanced options

- To override the default checkout options, click the **Options** button (). The Advanced Check Out dialog box appears. (For information on advanced options, see ["About Checking Out Files" on page 30](#).)

Accept defaults

- To accept the default checkout options, click the **Check Out** button. A red checkmark () appears next to each file icon.



**NOTE** Ignore the **Comments** field during checkout. You can add a comment during check-in.

## Undoing Checkout

When you undo a check, the archive is unlocked and a read-only workfile is left in the workfile location. No changes are checked into the archive.

**To undo a checkout:**


- 1 In the Solution Explorer pane, select the files, projects, or solution you want to undo a checkout on and right-click. A pop-up menu appears.

- 2 Select Undo Checkout. The Undo Checkout dialog box appears.


- 3 Select or deselect files in the **Name** list as needed.

- 4 Do one of the following:

Use advanced options

- To override the default undo checkout options, click the **Options** button (). The Advanced Undo Check Out dialog box appears. (For information on advanced options, see ["About Undoing Checkout" on page 32](#).)

Accept defaults

- To accept the default undo checkout options, click the **Undo Checkout** button. A blue padlock () appears next to each file icon.

## Checking In Files

By default, the following occurs when you check in a workfile:

- A new revision is created and assigned the next number in sequence.
- A read-only workfile is left in the workfile location.
- The archive is unlocked.

### To check in files:

- 1 In the Solution Explorer pane, select the files, projects, or solution you want to check in and right-click. A pop-up menu appears.
- 2 Select Check In. The Check In dialog box appears.
- 3 Do any of the following:
  - Select or deselect files in the **Name** list.
  - Enter a description of the changes you made to the files in the **Comments** field.



**TIP** To use unique descriptions for each file, leave the **Comments** field blank. After you complete the Check In dialog box, the Change Description dialog box will appear for each modified file in turn.

For Visual Basic projects, you will be prompted to enter descriptions for both the text files and the binary files, though Visual Studio normally hides the binary files from view.




- To invoke the Merge Tool, select a file and click the **Compare Versions** button. For information on differencing, see ["Generating Difference Reports" on page 52](#).




- To automatically check the files out again after the check-in operation, click the **Options** button and select Keep Checked Out.

- 4 Do one of the following:

Use advanced options

- To override the default check-in options, click the **Options** button () and select Advanced. The Advanced Check In dialog box appears. (For information on advanced options, see ["About Checking In Files" on page 33](#).)

Accept defaults

- To accept the default check-in options, click the **Check In** button. A blue padlock () appears next to each file icon (unless you chose to keep the files checked out).

## Chapter 10

# Visual Studio Rich Integration

---

Introduction	150
Accessing Supported Features	151
Visual Studio Rich Integration Overview	152
Collaborative Process Overview	153
Migrating and Converting Visual Studio Solutions	157
Working with Web Projects	161
Working with Branches	161
Setting Up Source Control Projects	167
Editing Files	174
Setting Default Options for Dialog Boxes	189
Comparing and Synchronizing Workspaces	193
Comparing Files and Resolving Conflicts	199
Associating and Working on SBM Issues	204

# Introduction



**NOTE** The information below only applies to the Version Manager rich integration to Visual Studio (RIDE). If you installed the SCC integration to Visual Studio (**SCC/COM IDE Client**), see [Chapter 9, "Visual Studio SCC Integration" on page 139](#) the "Visual Studio SCC Integration" chapter of the *Version Manager IDE Client Implementation Guide*.

This chapter has four purposes:

- List the Version Manager features available through Microsoft® Visual Studio and provide a quick reference to accessing those features.
- Note any features described in Part 1 of this manual that do not apply to this IDE
- Help you set up source control projects and add files to source control
- Help you access files that are under source control from within the IDE

For more information about source control concepts, see [Part 1, "The Version Manager IDE Client," on page 9](#) Part 1 of the *Version Manager IDE Client Implementation Guide*.



**IMPORTANT!** The rich integration 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 use the desktop client to 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 Visual Studio. 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.
- Remember that when you open a project from source control that you must specify the Version Manager workspace to use. If you wish to see the files and revisions defined by a different default version, then you must do an open from source control and specify the workspace that is associated with the desired label.

# Accessing Supported Features

The following table shows how to access features in the rich integration to Visual Studio. (Note, the "Right-Click" menu items are also available from the File | Source Control menu.)

To...	Select...	For more information see...
Get revisions	Right-Click   Get Latest Revision	<a href="#">"Getting Specific Files or Folders" on page 176</a>
Open a project from source control	File   Source Control   Open Project from Source Control	<a href="#">"Opening Solutions and Projects from Source Control" on page 170</a>
Check out revisions	Right-Click   Check Out	<a href="#">"Checking Out Files" on page 178</a>
Undo checkout of revisions	Right-Click   Undo Checkout	<a href="#">"Undoing Checkout" on page 180</a>
Check in revisions	Right-Click   Check In	<a href="#">"Checking In Files" on page 182</a>
Assign version labels	Right-Click   Label	<a href="#">"Labeling Revisions" on page 186</a>
View revision history	Right-Click   History	<a href="#">"Reviewing File History" on page 175</a>
Promote a file	(From the History view) Right-Click   Promote to Next	<a href="#">"Promoting Revisions" on page 188</a>
Work with Solutions Business Manager (SBM) issues	Right-Click   Related Issues	<a href="#">"Associating and Working on SBM Issues" on page 204</a>
Compare your local workspace to source control	Right-Click   Compare Workspaces	<a href="#">"Comparing and Synchronizing Workspaces" on page 193</a>
Compare your local workfile to the latest revision in Version Manager	Right-Click   Compare Revisions	<a href="#">"Comparing Files and Resolving Conflicts" on page 199</a>
Migrate from SCC to the rich integration	File   Source Control   Migrate	<a href="#">"Migrating and Converting Visual Studio Solutions" on page 157</a>
Open the Version Manager desktop client	File   Source Control   Launch Version Manager	<i>User's Guide</i> <i>Administrator's Guide</i>
View and submit SBM issues	View   Issues	<a href="#">"Associating and Working on SBM Issues" on page 204</a>
View SBM issues related to the currently selected file	Right-click   Related Issues	<a href="#">"Associating Issues" on page 209</a>

## About the Source Control Toolbar

The integration includes a toolbar for quick access to certain features. You can dock the toolbar to the Visual Studio toolbars, or leave it free floating.

By default, the toolbar includes the following buttons:

- |   |                     |    |                        |
|---|---------------------|----|------------------------|
| 1 | Check Out           | 6  | History                |
| 2 | Check In            | 7  | View Issues            |
| 3 | Undo Checkout       | 8  | Compare Revisions      |
| 4 | Get Latest Revision | 9  | Compare Workspaces     |
| 5 | File Status         | 10 | Launch Version Manager |

To change which buttons are included in the toolbar, click the drop-down menu and select Add or Remove Buttons | Source Control. Then select or deselect items from the list.

## Visual Studio Rich Integration Overview

The Version Manager rich integration (RIDE) to Microsoft Visual Studio provides a powerful set of collaborative tools to help development teams manage their source code. The integration includes:

- **Workspace Comparison and Synchronization:** You can easily compare the state of all files in your local workspace with the corresponding Version Manager project's workspace. You can determine what the differences are, and whether your workspace or Version Manager need to be updated. With the click of a button, you can then automatically check in all of your changes to Version Manager, and get all updates to your local workspace. See ["Comparing and Synchronizing Workspaces" on page 193](#).
- **Automated File Merging:** When you check in or synchronize your local workspace, your changes are automatically merged with any recent changes to the same files in Version Manager.
- **File Comparison and Conflict Resolution:** You can compare specific local files to the latest revision of the files in Version Manager. If necessary, you can directly edit the files in order to resolve any conflicts before checking files in. See ["Comparing Files and Resolving Conflicts" on page 199](#).
- **Pragmatic Locking:** You can check in any file, any time, without first locking it. This allows you to edit any file without first explicitly locking the file. When you then check in the file, the file is temporarily locked while your local changes are merged with the latest revision in Version Manager (if necessary). Once the new revision is created, the file is then immediately unlocked again, allowing other users to check in their changes.

By default, pragmatic (optimistic) locking is enabled. However, the administrator can turn it off for any given project database, thus requiring users to checkout files before editing.

- **Working Offline:** Even when you are unable to connect to the Version Manager project database, you can continue to work on any projects already in your



workspace, provided that pragmatic locking is in effect or you already have them checked out. See ["Working While Offline" on page 188](#).

## Solutions Business Manager Integration

If your organization uses Solutions Business Manager (SBM) to track development issues, such as defects and tasks, you can access your issues from within the Version Manager integration to Visual Studio. You can submit and modify SBM issues from within Visual Studio, and then associate issues with specific files. When you associate issues with revisions of files, information about the associated revisions is added to the issues.

For detailed information on the SBM integration to Version Manager, see ["Associating and Working on SBM Issues" on page 204](#).

## Supported Project Types

The rich integration to Visual Studio works with all project types that Visual Studio makes available for source control operations.

**IMPORTANT!** All files of a given CAB or C++ project must reside under the root directory of the Visual Studio project. Cutting and pasting files from one C++ project to another inside of Visual Studio violates this requirement.

## Rebinding a Solution

Version Manager RIDE can only open projects and solutions that were added using the integration. For information about rebinding, see the following Knowledgebase article:

<http://knowledgebase.serena.com/InfoCenter/index?page=content&id=D16727&token=ca1f979367db4209a586df45f31eda99>

# Collaborative Process Overview

The Version Manager integration supports a flexible range of collaborative development processes. You can:

- Use Version Manager workspaces to precisely define which files developers will work on and to automate the assignment of version labels in order to enforce maintenance of the workspaces. Workspaces greatly simplify the process of sharing files in a development project by ensuring that all developers working within the project see and update the same files. See ["Using Workspaces" on page 154](#).
- Use a pragmatic (optimistic) locking model to allow all users to work on common files without locking them. See ["Working on Files Without Locking Them" on page 155](#).
- Prevent multiple users from modifying the same files by requiring locks in order to edit files. See ["Checking Out \(Locking\) Files" on page 156](#).

## Using Workspaces

The rich integration to Visual Studio makes extensive use of Version Manager workspaces to simplify the collaborative process. Version Manager workspaces represent a collection of specific files, each of which shares a common default version label. Workspaces enable developers to get and work only on files associated with specific development efforts or projects, as defined by the default version label.

Version Manager workspaces define a default workfile location. However, this workfile location does not apply to IDE clients, such as the rich integration to Visual Studio. Instead, the workfile location specified in Visual Studio is used. Other Version Manager clients, such as the CLI or Web client, use the workfile location defined in the Version Manager workspace.

The following steps provide an overview of workspace setup and usage, with the rich integration to Visual Studio:

- 1 The Version Manager administrator sets up the workspace in Version Manager.** Unique workspaces can be set up for each project or subproject, and even for each developer. The simplest workflow is to define a common workspace that all members on a project team can share, and that defines the correct default version label for the project. If developers will use different workspaces but will work on a common project, the default version label should be set to the same value for each workspace. Because it is the version label that determines what each developer can see and modify, it is important that the default is common for all developers who will work on the same branch of development. See the *Version Manager Administrator's Guide* for details.

Any changes to the project structure in one workspace will affect all other workspaces that include the Visual Studio project. If a developer renames, moves, deletes, or adds files, those changes will appear in the other workspaces. Carefully evaluate such dependencies when defining workspaces.

- 2 The administrator defines the default version label for the workspace.** The default version label will in turn determine precisely which files should belong to the workspace.

For example, if the floating label "Latest" is assigned to all files in the project, and if the workspace should include the most recent versions of all files in the project, define the default version label as "Latest."

Or, if the workspace should include all files in a branch for which the branch version label is "branch\_01," then the default version label for the workspace should be defined as "branch\_01."

See ["Working with Branches" on page 161](#) and the *Version Manager Administrator's Guide* for details.

If Visual Studio is open when a default label is defined or changed, Visual Studio must be restarted before the new default label takes effect.

- 3 A developer selects the appropriate workspace when adding the Visual Studio solution or project to source control.** By selecting the appropriate workspace, the default version label for that workspace is assigned to all files that are added to Version Manager. For example, if the developer chooses a workspace for which "Latest" is the default version label, then the "Latest" label is assigned to all files.

- 4 Other developers select the appropriate workspace when opening the solution or project from source control.** This choice then determines what files are copied to the local workspaces. For example, if developers choose a workspace for which "Latest" is the default version label, then only those files to which "Latest" is assigned are copied to the local workspaces.
- 5 Developers synchronize workspaces, automatically checking in / getting files with the default version label.** When a developer synchronizes a local workspace with the Version Manager project:
  - Any new local files are added to the Version Manager project, and the default version label of the current workspace is assigned to them.
  - Any new files in Version Manager to which the default version label has been assigned are copied to the local workspace.

## Working on Files Without Locking Them

In this workflow, files are locked only momentarily at the point when you actually check-in changes, rather than before you begin working on the files. In Version Manager, we call this *pragmatic locking*. This model enables multiple users to get and modify the same files at the same time. Each user's modifications will be auto-merged with other users' changes when the files are checked in. If any conflicts result from attempting to auto-merge multiple users' changes, the check-in of the conflicting files will fail. You can resolve any such conflicts by launching the Merge tool from the Compare Workspaces view. Alternatively, you could simply synchronize from the Compare Workspaces view rather than invoke the Check In dialog box in the first place.

By default, pragmatic (optimistic) locking is enabled. However, the administrator can turn it off for any given project database, thus requiring users to checkout files before editing. See the *Administrator's Guide*.

The following steps illustrate this process:

- 1** At the beginning of each work day, make sure that your local workspace has all of the latest updates from the corresponding Version Manager project. To do this, you can either update your local workspace with changes in the Version Manager project (see ["Getting All Updates from Version Manager" on page 195](#)) or synchronize your local workspace with the corresponding project (see ["Comparing and Synchronizing Workspaces" on page 193](#)). If the local files already exist and are different from the latest revisions in Version Manager, the different revisions are merged. If any conflicts result from the attempt to merge the different revisions, identify the files with the Compare Workspaces view and resolve them with the Merge tool.
- 2** Edit the files.
- 3** When you have finished your work on the files, commit all local changes to Version Manager (["Committing Local Changes to Version Manager" on page 197](#)), or synchronize your workspace with the corresponding Version Manager workspace.

By default, when you get files from source control, they are set to read-only in your local workspace. In order to make changes to them, you must either make them writable from Windows Explorer before you start work on them, or you must choose to overwrite the file when attempting to save your changes in Visual Studio.

### **Example: Collaborative Development**

A team of developers work together on files stored in a project called "Patch2," in a project database called "Source." Their project allows multiple developers to work on common files at one time. Their development workflow is:

- 1 At the beginning of every day, each developer can update the local workspace with the latest changes under the "Patch2" project in Version Manager. Because the development team works from sites in multiple nations, this synchronization ensures that all developers have all of the latest updates from all development sites. See ["Synchronizing Workspaces" on page 198](#).
- 2 At the end of each day, each developer synchronizes their workspace with the corresponding Version Manager project, in order to check in all changes to files in the local workspace. If multiple developers have modified the same files, the merge functionality will attempt to merge each user's changes into the new revisions. All merged and changed files are copied to the local workspace. If conflicts result from the attempt to merge, the developer performing the check-in must resolve the conflicts, and then synchronize again. See ["Synchronizing Workspaces" on page 198](#).

## **Checking Out (Locking) Files**

In this workflow, check out any file that you intend to modify. This locks the file in Version Manager, which prevents other users from checking in changes until you unlock the files. Once you have completed your changes, check in the files. This ensures that changes from multiple users will never result in conflicts.

**IMPORTANT!** Checking in local changes does not synchronize such changes as renamed and moved files. To do this, commit all local changes to Version Manager (["Committing Local Changes to Version Manager" on page 197](#)), or synchronize your workspace with the corresponding Version Manager workspace

By default, pragmatic (optimistic) locking is enabled. However, the administrator can turn it off for any given project database, thus requiring users to checkout files before editing.

### **Example: Locking Files**

As part of a larger development team working on a complex and interdependent code base, Joe and Carol frequently work on the same files. A typical day might include the following:

- 1 To begin work on a set of files, Joe checks them out from Version Manager (see ["Checking Out Files" on page 178](#)).
- 2 When Carol opens the project, she displays the Compare Workspace view to check whether the files that she needs are currently checked out (see ["Comparing Workspaces" on page 194](#)). She sees that they are. Using the History view, she sees that Joe has the files checked out (see ["Reviewing File History" on page 175](#)).
- 3 Joe completes his work and checks in the files, associating them with the SBM issues that spawned the work (see ["Checking In Files" on page 182](#)).
- 4 Carol comes back from lunch and checks the Compare Workspace view to see what the current file status is. She sees that the files are checked in, so she checks them out and begins her work.

# Migrating and Converting Visual Studio Solutions

Consider the following before migrating your Visual Studio solutions:

- **Do not mix integrations.** All users who work on a common project must use the same integration (SCC or rich). Once you migrate a project to the rich integration, all users must open the project via the rich integration.
- All projects must be located under the solution folder. Projects outside the solution tree cannot be migrated.
- Do **not** migrate your SCC projects to the rich integration if you use TrackerLink, and will continue to use it. The rich integration supports SBM. Do not migrate to the rich integration until you have successfully migrated your Tracker projects to SBM. You can then use the rich integration to SBM from Visual Studio.

**NOTE** If your Visual Studio solutions are under Visual SourceSafe source control, see the "Using the Version Manager Conversion Utility for SourceSafe" chapter in the *Administrator's Guide*.

[Migrating from Visual Studio 2003 to Visual Studio RIDE](#)


[Migrating from Visual Studio SCC to Visual Studio RIDE](#)


[Migrating from Visual Studio 2005 to Visual Studio RIDE](#)

## Migrating from Visual Studio 2003 to Visual Studio RIDE

This procedure describes how to convert a solution from Visual Studio .NET 2003 SCC and update the source control integration to Visual Studio RIDE.

Before migrating a solution, all users should check in their work and unlock all files.

- 1 Check out the solution and all of its projects and sub projects using the Version Manager desktop client.
- 2 Launch Visual Studio.
- 3 Open the solution. The Visual Studio Conversion Wizard appears.
- 4 Complete the Visual Studio Conversion Wizard. See the Microsoft documentation for more information.
- 5 Select File | Save All.
- 6 Select File | Source Control | Migrate. Page-1 of the PVCS Version Manager Migration Wizard appears.
- 7 To specify the Version Manager project database that contains the Visual Studio solution, do any of the following:
  - Enter the path to the root project database directory, or select a recent project database from the drop-down list.
  - Click the **Browse for PDB** button  to browse for a project database. In this case, you must select the .ser file located directly under the root PDB directory.

- Click the **Browse File Servers** button  to choose from all project databases on the Version Manager file server. File servers must be defined from the Version Manager desktop client. *If no Version Manager file server is defined, ignore this button.*
  - 8 Enter your Version Manager user name and password, and click **Next**. Page-2 of the wizard appears.
  - 9 Select the Version Manager workspace to use and click **Next**. Consider the following:
    - Your choice of workspace determines the default version label and promotion group for the files. Only those files to which the default version label is assigned will be opened to your local workspace. Later on, when synchronizing your local workspace with Version Manager, the default version label will be assigned to any new files that you add. See ["Using Workspaces" on page 154](#).
    - The choice of workspace also determines which project files you can open. You can open projects to which the default version label is assigned. For example, if the default version label is "branch," you will only be able to open a project file to which the "branch" version label is assigned.
- Page-3 of the wizard appears.
- 10 Select the Version Manager project that you want to migrate.
- 11 In the **Solution file** field, enter or browse to the workfile location of the Visual Studio solution (.sln) file that you want to migrate. Be sure to point to the same location that you checked the file out to in Step 1.
- 12 Click **Next**.

Page-4 of the wizard appears.

- 13 Review the choices you have made. Click the **Back** button to change any of the settings. When you are finished, click the **Finish** button.
- 14 Select File | Save All.
- 15 Close the solution (File | Close Solution).
- 16 Check in the solution and all of its projects and subprojects using the Version Manager desktop client.
- 17 Each user must now open the updated solution from source control (File | Source Control | Open Project from Source Control). See ["Opening Solutions and Projects from Source Control" on page 170](#).

[Migrating and Converting Visual Studio Solutions](#)



[Migrating from Visual Studio SCC to Visual Studio RIDE](#)

[Migrating from Visual Studio 2005 to Visual Studio RIDE](#)

## Migrating from Visual Studio SCC to Visual Studio RIDE

This procedure describes how to convert a solution, and migrate the source control integration, from Visual Studio .NET 2003 to the Visual Studio Rich Integration Development Environment (RIDE).

Before migrating a solution, all users should check in their work and unlock all files.

- 1 Check out the solution and all of its projects and sub projects using the Version Manager desktop client.
- 2 Launch Visual Studio.
- 3 Close any open solutions (File | Close Solution).
- 4 Select File | Source Control | Migrate. Page-1 of the PVCS Version Manager Migration Wizard appears.
- 5 To specify the Version Manager project database that contains the Visual Studio solution, do any of the following:
  - Enter the path to the root project database directory, or select a recent project database from the drop-down list.
  - Click the **Browse for PDB** button  to browse for a project database. In this case, you must select the .ser file located directly under the root PDB directory.
  - Click the **Browse File Servers** button  to choose from all project databases on the Version Manager file server. File servers must be defined from the Version Manager desktop client. *If no Version Manager file server is defined, ignore this button.*
- 6 Enter your Version Manager user name and password, and click **Next**. Page-2 of the wizard appears.
- 7 Select the Version Manager workspace to use and click **Next**. Consider the following:
  - Your choice of workspace determines the default version label and promotion group for the files. Only those files to which the default version label is assigned will be opened to your local workspace. Later on, when synchronizing your local workspace with Version Manager, the default version label will be assigned to any new files that you add. See ["Using Workspaces" on page 154](#).
  - The choice of workspace also determines which project files you can open. You can open projects to which the default version label is assigned. For example, if the default version label is "branch," you will only be able to open a project file to which the "branch" version label is assigned.

Page-3 of the wizard appears.

- 8 Select the Version Manager project that you want to migrate.
- 9 In the **Solution file** field, enter or browse to the workfile location of the Visual Studio solution (.sln) file that you want to migrate. Be sure to point to the same location that you checked the file out to in Step 1.

- 10 Click **Next**.

Page-4 of the wizard appears.

- 11 Review the choices you have made. Click the **Back** button to change any of the settings. When you are finished, click the **Finish** button. The Visual Studio Conversion Wizard appears.
- 12 Complete the Visual Studio Conversion Wizard. See the Microsoft documentation for more information.
- 13 Select File | Save All.



- 14 Check in the solution and all of its projects and subprojects using the Version Manager desktop client.
- 15 Each user must now open the updated solution from source control (File | Source Control | Open Project from Source Control). See ["Opening Solutions and Projects from Source Control" on page 170](#).



[Migrating and Converting Visual Studio Solutions](#)

[Migrating from Visual Studio 2003 to Visual Studio RIDE](#)

[Migrating from Visual Studio 2005 to Visual Studio RIDE](#)

## Migrating from Visual Studio 2005 to Visual Studio RIDE

This procedure describes how to convert a Visual Studio 2005 SCC solution to Visual Studio RIDE. Before migrating a solution, all users should check in their work and unlock all files.

- 1 Check out the solution and all of its projects and sub projects using the Version Manager desktop client.
- 2 Launch Visual Studio.
- 3 Close any open solutions (File | Close Solution).
- 4 Select File | Source Control | Migrate. Page-1 of the PVCS Version Manager Migration Wizard appears.
- 5 To specify the Version Manager project database that contains the Visual Studio solution, do any of the following:
  - Enter the path to the root project database directory, or select a recent project database from the drop-down list.
  - Click the **Browse for PDB** button  to browse for a project database. In this case, you must select the .ser file located directly under the root PDB directory.
  - Click the **Browse File Servers** button  to choose from all project databases on the Version Manager file server. File servers must be defined from the Version Manager desktop client. *If no Version Manager file server is defined, ignore this button.*
- 6 Enter your Version Manager user name and password, and click **Next**. Page-2 of the wizard appears.
- 7 Select the Version Manager workspace to use and click **Next**. Consider the following:
  - Your choice of workspace determines the default version label and promotion group for the files. Only those files to which the default version label is assigned will be opened to your local workspace. Later on, when synchronizing your local workspace with Version Manager, the default version label will be assigned to any new files that you add. See ["Using Workspaces" on page 154](#).
  - The choice of workspace also determines which project files you can open. You can open projects to which the default version label is assigned. For example, if the default version label is "branch," you will only be able to open a project file to which the "branch" version label is assigned.



Page-3 of the wizard appears.

- 8 Select the Version Manager project that you want to migrate.
- 9 In the **Solution file** field, enter or browse to the workfile location of the Visual Studio solution (.sln) file that you want to migrate. Be sure to point to the same location that you checked the file out to in Step 1.
- 10 Click **Next**.

Page-4 of the wizard appears.

- 11 Review the choices you have made. Click the **Back** button to change any of the settings. When you are finished, click the **Finish** button.
- 12 Check in the solution and all of its projects and subprojects using the Version Manager desktop client.
- 13 Each user must now open the updated solution from source control (File | Source Control | Open Project from Source Control). See ["Opening Solutions and Projects from Source Control" on page 170](#).

[Migrating and Converting Visual Studio Solutions](#)

[Migrating from Visual Studio 2003 to Visual Studio RIDE](#)

[Migrating from Visual Studio SCC to Visual Studio RIDE](#)

## Working with Web Projects

The rich integration supports Web projects, such as ASP Application projects. When working with Web projects, remember the following:

- When you add a Web project to a solution, you define a Web URL for the project. All files in the project are stored in this Web location, rather than under the root solution folder in your Visual Studio workspace. This URL does correspond to a physical directory. This physical directory mapping is defined by your Web server. Once you have successfully added the projects to your solution, you can add them to Version Manager like any other type of project. See ["Adding Solutions and Projects to Version Manager" on page 168](#).
- When you open a single web project from source control, it opens in a new, blank solution. The steps to open Web projects are a bit different than those for a non-web project. See ["Opening Web Projects from Source Control" on page 173](#).

## Working with Branches

A branch is a separate line of development consisting of one or more revisions that diverge from a revision on the trunk (mainline of development) or from another branch. Branching allows you to develop alternative versions of a file in parallel with other developers who are working on the trunk or on another branch.

Some reasons for creating a branch include:

- Developing platform specific versions of a file.
- Fixing a bug or developing a new feature without interrupting the mainline of development.
- Customizing your code to meet the needs of a specific customer without affecting the mainline of development.

When you create a branch, a new revision is created. This new revision has the same revision number as the revision you are branching from, but with two more digits added to the end, "1.0". So if revision 1.5 is the revision you are branching from, revision 1.5.1.0 will be the first revision in the new branch.

## Viewing Branched Files

You can see all revisions on all branches of a particular file from the History view.

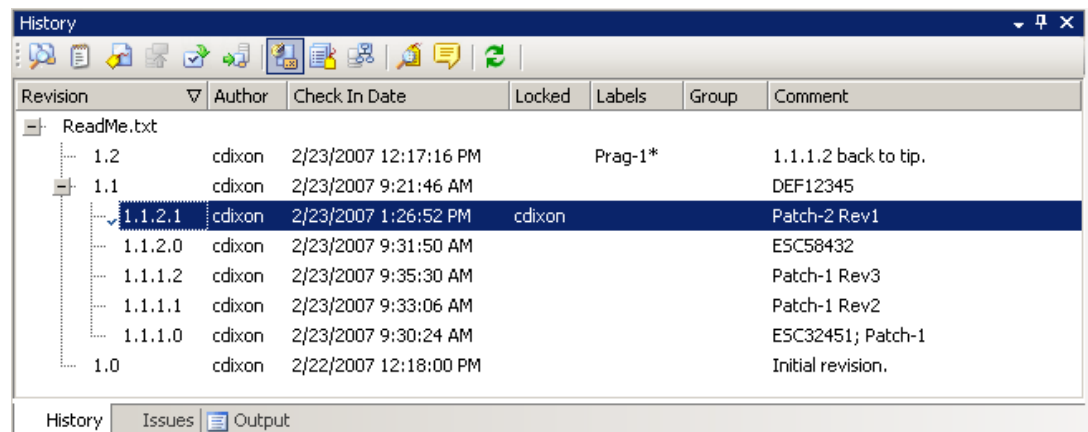
### To view revisions on a branch:

- 1 Right-Click on the desired file in the Solution Explorer and select **History** from the resulting menu. The History view displays a list of all revisions of the file.
- 2 Click the nodes to expand branches in the tree, as needed, in order to locate the desired revision.

### How Do I Tell a Branch from the Trunk and One Branch from Another?

In the following image:

- 1.0 is the initial revision of the file ReadMe.txt.
- 1.2 is the latest (tip) revision of the mainline of development (trunk) as defined by the default version label Prag-1.
- 1.1.1.0 is the initial revision of the first branch off of revision 1.1.
- 1.1.1.2 is the tip revision of the 1.1.1.x branch.
- 1.1.2.0 is the initial revision of the second branch off of revision 1.1.
- 1.1.2.1 is the tip revision of the 1.1.2.x branch.



## How Should I Branch My Files?

The integration provides two ways to create and work with branched files:

- **Automatic Label-Based Branching:** Which allows you to:
  - Branch an entire project or group of files.
  - Check out an entire branched project or group of files.
  - Manage each branch and trunk with separate Version Manager workspaces.
- **Manual Branching:** Which allows you to:
  - Create one branched file at a time.
  - Check out one branched file at a time.

To branch just a file or two, you can use the manual method. To branch a number of files or an entire project, use automatic label-based branching.


## Automatic Label-Based Branching

A user with appropriate privileges must use the Version Manager desktop client to configure the workspace for automatic label-based branching.

### Creating a branch:

- 1 From the Version Manager desktop client, create a workspace for the branch. For example, you could have a workspace named TRUNK for the mainline of development and a workspace named PATCH-1 for the new branch. See ["Using Workspaces" on page 154](#) and the *Administrator's Guide*.

Any changes to the project structure in one workspace will affect all other workspaces that include the Visual Studio project. If a developer renames, moves, deletes, or adds files, those changes will appear in the other workspaces. Carefully evaluate such dependencies when defining workspaces.

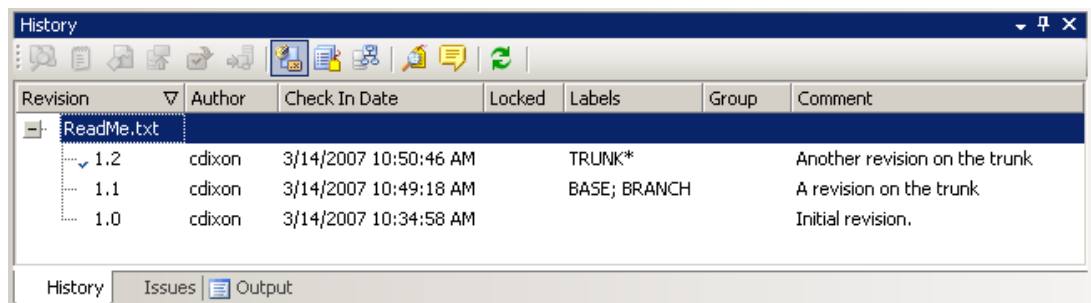
- 2 Switch to the workspace that you created for the branch. (Click on the workspace icon  RootWorkspace at the bottom of the desktop client and select the desired workspace from the resulting dialog box.)
- 3 From the desktop client, select the project/folder that is at the top of the hierarchy that you wish to branch.
- 4 Right-click and select **Properties** from the resulting menu. The Properties dialog box appears.
- 5 Select the **Workspace Settings** tab.
- 6 Do the following:
  - a **Workfile Location:** This field does not apply to the Visual Studio integration. However, if you access the files with other Version Manager clients, the files will be written to this location during get and check out operations.
  - b **Default Version:** The version label entered here determines which revision is acted upon by source control operations. Enter the version label that you want to associate with the latest revision of each file in the branch.

You may find it easier to apply the labels associated with these fields before you define the Default Version. Once the Default Version is defined, every action (including applying labels) will require that you specify a specific revision to act upon, unless the label defined for the Default Version already exists.

- c **Branch Version:** Enter the same version label that you entered in the **Default Version** field. This allows you to operate on the tip of the branch rather than the tip of the trunk.
  - d **Base Version:** The version label entered here determines which existing revision the branch will branch off from.
- 7 Assign the version labels that you defined for the Default Version and the Base Version to the revision that you want to branch from. Apply regular fixed labels, not floating labels.

If you defined the Default Version before applying the label associated with it, you must specify a specific revision in order to apply a label. Specifying 1.\*, for example, will apply the label to the tip of the trunk.

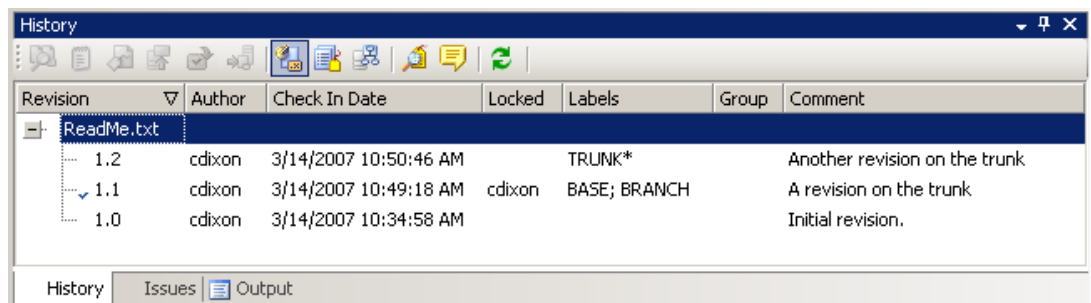
**Results:** From the History view in Visual Studio, the results might look something like the following if TRUNK is a floating label that marks the tip of the trunk, and BASE and BRANCH are on the revision targeted for branching:



Revision	Author	Check In Date	Locked	Labels	Group	Comment
1.2	cdixon	3/14/2007 10:50:46 AM		TRUNK*		Another revision on the trunk
1.1	cdixon	3/14/2007 10:49:18 AM		BASE; BRANCH		A revision on the trunk
1.0	cdixon	3/14/2007 10:34:58 AM				Initial revision.

- 8 From Visual Studio, select the solution, project, folder, or files to be branched, and check them out.

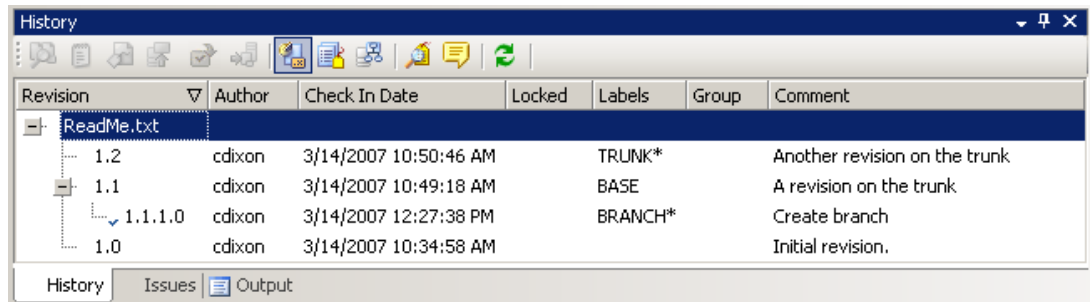
**Results:** The revision with the label corresponding to the Default Version is checked out rather than the tip of the trunk.



Revision	Author	Check In Date	Locked	Labels	Group	Comment
1.2	cdixon	3/14/2007 10:50:46 AM		BASE; BRANCH		Another revision on the trunk
1.1	cdixon	3/14/2007 10:49:18 AM	cdixon	TRUNK*		A revision on the trunk
1.0	cdixon	3/14/2007 10:34:58 AM				Initial revision.

- 9 From Visual Studio, check in the items to be branched. On the Options tab, set the **If file is unchanged** option to **Check in**. A new revision must be checked in to create a branch.

**Results:** The branch is created, and the label you defined as the Branch Version is now a floating label on the tip of the branch.



For more information on configuring branching and workspaces, see the *Administrator's Guide*.

## Manual Branching

There are two ways to manually create a branch:

- Select the **Force branch** option when you check in a file.
- Check out a non-tip revision (any revision other than the latest revision on a given trunk or branch) and check it back in.

### Using the Force Branch Option

You can create a branch by using the **Force Branch** option during check in.

#### To create a branch:

- 1 Select the file or files to branch and then invoke the Check In dialog box (Right-Click | Check In). If your organization enforces pessimistic locking, you must first check out the files (Right-Click | Check Out).
- 2 On the Options tab, select the **Force branch** option. On the Options tab, set the **If file is unchanged** option to **Check in**. A new revision must be checked in to create a branch.
- 3 Enter a comment and set any other options as desired.
- 4 Click the **Check In** button.

For complete information on the Check In dialog box, see ["Checking In Files" on page 182](#).

### Branching by Checking In a Non-Tip Revision

You can create a branch by checking out a non-tip revision of a file and checking it back in.

#### To create a branch:

- 1 Select the desired file in the Solution Explorer.
- 2 Right-Click and select **History** from the resulting menu. The History view displays a list of all revisions of the file.

The blue checkmark in the History view indicates which revision is currently in your local workspace.

- 3 Click the nodes to expand branches in the tree, as needed, in order to locate the desired revision.
- 4 Right-Click on the desired revision and select **Check Out Revision** from the resulting menu.

A red checkmark appears next to the file in the Solution Explorer and a blue checkmark appears next to the revision in the History view. The file is checked out.

- 5 Right-Click the file in the Solution Explorer and select Check In from the resulting menu. The Check In dialog box appears. On the Options tab, set the **If file is unchanged** option to **Check in**. A new revision must be checked in to create a branch.
- 6 Enter a comment and set any other options as desired.
- 7 Click the **Check In** button.

For complete information on the Check In dialog box, see ["Checking In Files" on page 182](#).

## Editing Revisions on a Branch

You can view and edit any revision of any branch from the **History** view.

### To edit a revision on a branch:

- 1 Right-Click on the desired file in the Solution Explorer and select **History** from the resulting menu. The History view displays a list of all revisions of the file.
- 2 Click the nodes to expand branches in the tree as needed in order to locate the desired revision.
- 3 Right-Click on the desired revision and select **Check Out Revision** or **Get Revision** from the resulting menu, depending on whether your organization requires a lock in order to edit files.

The blue checkmark in the History view indicates which revision is in your local workspace. The **Locked** column indicates which revisions are locked and by whom.

## Checking In Branched Files

When you check in a file (whether it is branched or not), a new revision is created in one of the following places, depending on how your Version Manager workspace is configured and whether a non-tip revision is checked out:

If	Then
No revision is locked, AND A Default Version has been defined for the workspace and applied to the file.	The new revision is checked in as the tip on the branch or trunk that is defined by the Default Version.
No revision is locked, AND No Default Version is in effect for the file.	The new revision is checked in as the tip on the trunk.
A single tip revision is locked	The new revision is checked in as the tip on the branch or trunk that contains the locked revision.
Multiple revisions are locked	You are prompted to select which revision to unlock. The new revision is created as the tip on the branch or trunk that you unlock.
A non-tip revision is locked	The new revision is checked in as the tip of a new branch off of the locked revision.

### To check in branched files:

- 1 Save all of the files that you are checking in, including any project or solution files.
- 2 Right-Click on the desired files or folders in the Solution Explorer and select **Check In** from the resulting menu. The Check In dialog box appears.
- 3 Enter a comment and set any other options as desired.
- 4 Click **Check In**. If multiple revisions of the file are locked, you will be prompted to select which revision to unlock.

For complete information on the Check In dialog box, see ["Checking In Files" on page 182](#).

## Setting Up Source Control Projects

In order to use the Version Manager integration to Visual Studio, each developer on the project must associate their local Visual Studio projects with a corresponding Version Manager project. Once they have done this, all developers working on the project can get all recent updates to the files, and check in new changes to the files. They can synchronize their local workspace with the Version Manager project, which acts as a central repository for every developer's changes. You can do this in one of two ways:

- If the solution or project does not yet exist in Version Manager, add the solution or project to source control. This creates a corresponding project, and adds all of the folders and files in the project to Version Manager. See ["Adding Solutions and Projects to Version Manager" on page 168](#).

- If the solution or project already exists in Version Manager, add it to your local workspace. This copies all files from the Version Manager project to your Visual Studio workspace, and establishes a relationship between your local solution and the corresponding Version Manager project. See ["Opening Solutions and Projects from Source Control" on page 170](#).

## Adding Solutions and Projects to Version Manager

If your solution or project has not yet been added to Version Manager, you must do so in order to use the integration to Version Manager. You can choose whether to:

- **Add your entire solution to Version Manager.** By adding the entire solution, you can store all projects and files in the solution within a single project database in Version Manager. Depending on your development scenario, this may simplify the collaborative process for members of your organization, as developers will only need to get and work within the single project database.
- **Add some projects in a solution to Version Manager.** You can add just some of the projects within a solution to source control, if you don't want to add the entire solution. This allows you the option of working with projects that are under source control within solutions that are not, themselves, under source control. This may be necessary or desirable in certain cases.

### IMPORTANT!

- Visual Studio projects do not need to be located under the root directory of a solution -BUT, if the projects ARE located under the root of the solution, they MUST be added to the same Version Manager project database as the solution.
- All files of a given CAB or C++ project must reside under the root directory of the Visual Studio project. Cutting and pasting files from one C++ project to another inside of Visual Studio violates this requirement.
- You can store just one solution or project file within each Version Manager project or subproject. For example, do not attempt to store more than one solution file within the same project in Version Manager. When adding solutions and projects to source control, select destination projects in Version Manager that are distinct from each other.

When you add a solution or project to source control:

- All files in the solution or project are added to the target Version Manager project database, with the exception of file types that are ignored by source control operations in Visual Studio.
- For each project or folder within the solution or project that you add, a corresponding Version Manager subproject is created. For example, if you add a solution that contains four projects, four subprojects will be created in the Version Manager project database.
- The default version label for the workspace that you choose is assigned to all files as they are added to source control. The default label is assigned as a *floating* label, so that it is always associated with the latest revision. This ensures that other users who will use this workspace see and work on the same set of files. See ["Using Workspaces" on page 154](#).





**To add a solution or project to Version Manager:**

- 1 If necessary, from the Version Manager desktop client, create the Version Manager project database that you will add the solution or project to. See the *Version Manager Administrator's Guide* for more information.
- 2 In Visual Studio, open the Solution that you want to add to Version Manager, or that contains the projects you want to add to source control.
- 3 Save the solution and all projects within before adding it to Version Manager.
- 4 Do one of the following:

If	Then
You will add the solution to source control	<ol style="list-style-type: none"> <li>a Select the solution root in the Solution Explorer.</li> <li>b Right-click, and select <b>Add Solution to Source Control</b> from the resulting menu.</li> </ol>
You will add a specific project to source control	<ol style="list-style-type: none"> <li>a Select the project in the Solution Explorer.</li> <li>b Right-click, and select <b>Add Project to Source Control</b> from the resulting menu.</li> </ol>

The Add to Source Control wizard appears.

- 5 Specify the project database to which you will add the solution or project. Do any of the following to specify a project database:
  - Enter the path to the root project database directory, or select a recent project database from the drop-down list.
  - Click the **Browse for PDB** button  to browse for a project database. In this case, you must select any of the .ser files located directly under the root PDB directory.
  - Click the **Browse File Servers** button  to choose from all project databases on the Version Manager file server, if a file server is defined in Version Manager. File servers must be defined from the Version Manager desktop client. *If no Version Manager file server is defined, ignore this button.*
- 6 Enter your Version Manager user name and password and click **Next**. The Workspace page of the wizard appears.
- 7 Select the Version Manager workspace that you will use for the solution or project. All public workspaces in the project, and any private workspaces to which you have access, are displayed. Note the following:
  - Your workspace choice determines the default version label for the files. This label is assigned to all files as they are added to Version Manager. This ensures that any other user that will use this workspace (for example, by opening the solution or project from source control) will work on the same set of files.
  - The choice of workspace does not affect your workfile location setting; working copies of all of your files will be saved to the Visual Studio workspace.
- 8 Click **Next**. The Create Project page of the wizard appears.
- 9 Select the project or subproject, within the project database, to which you will add your solution or project.

You can store just one solution file within each Version Manager project or subproject. Do not attempt to store more than one solution file within the same project in Version Manager. When adding solutions and projects to source control, select destination projects in Version Manager that are distinct from each other.

- 10** Click **Next**. The SBM page of the wizard appears.
- 11** Enter the SBM server name in the **SBM Server** field. If the SBM server uses a non-default port number (any port except 80), append the port number to the server name. For example, if the port number is 89:  
  
`tt_server:89`
- 12** Enter your SBM user name and password.
- 13** Click **Next**. The Review page of the wizard appears.
- 14** Review the choices you made. Click the **Back** button to change any of the settings. When you are finished, click the **Finish** button.

## Opening Solutions and Projects from Source Control



If the Visual Studio project that you want to work on was added to source control from a different computer, you must open it from source control before you can start work on it. Once you open a solution or project from source control, the Visual Studio solution or project files will be available in your local workspace.

You can choose to open an entire solution and all projects within it at one time ("[Opening Solutions from Source Control](#)" on page 170), add a project to an open solution ("[Opening Non-Web Projects from Source Control](#)" on page 171), or open a single Web project into a new solution ("[Opening Web Projects from Source Control](#)" on page 173).

### ***Opening Solutions from Source Control***

Complete the following steps to open an entire solution from Version Manager, including all projects in that solution.

#### **To open a solution from Version Manager:**

- 1** Select File | Open Solution From Source Control. The Open from Source Control wizard appears.
- 2** Specify the project database that contains the solution you are opening. Do any of the following to specify a project database:
  - Enter the path to the root project database directory, or select a recent project database from the drop-down list.
  - Click the **Browse for PDB** button  to browse for a project database. In this case, you must select any of the .ser files located directly under the root PDB directory.
  - Click the **Browse File Servers** button  to choose from all project databases on the Version Manager file server. File servers must be defined from the Version Manager desktop client. *If no Version Manager file server is defined, ignore this button.*
- 3** Enter your Version Manager user name and password.

- 4 Click **Next**. The Workspace page of the wizard appears.
- 5 Select the workspace that you will use for all projects within the solution. Note:
  - Your choice of workspace will determine the default version label and promotion group for the files. Only those files to which the default version label is assigned will be opened to your local workspace. Later on, when synchronizing your local workspace with Version Manager, the default version label will be assigned to any new files that you add. See ["Using Workspaces" on page 154](#).
  - The choice of workspace also determines which solution files you can open. You can open a solution to which the default version label is assigned. For example, if the default version label is "branch," you will only be able to open a solution file to which the "branch" version label is assigned.
- 6 Click **Next**. The Create Project page of the wizard appears.
- 7 Browse the project database folders to find and select the specific solution (.sln) file that you want to open. All solution files to which your default version label is assigned are displayed. If your default version label is not assigned to the latest revision of any solution files, then no solution files will be visible to you.
- 8 In the **Workfile location** field, enter or browse to select the local workspace location. This is the local work directory, where your working copies of the files in the solution will be stored.
- 9 Click **Next**. The SBM page of the wizard appears.
- 10 Enter the SBM server name in the **SBM Server** field. If the SBM server uses a non-default port number (any port except 80), append the port number to the server name. For example, if the port number is 89:
 



```
tt_server:89
```
- 11 Enter your SBM user name and password.
- 12 Click **Next**. The Review page of the wizard appears.
- 13 Review the choices you made. Click the **Back** button to change any of the settings. When you are finished, click the **Finish** button.
- 14 If you are opening a solution that contains Web projects, you are prompted to enter the full path to the physical directory that corresponds to your Web server's root URL. The workfiles for your Web projects will be stored here. For example, if your root Web URL (such as http://localhost) maps to the following directory: c:\inetpub\wwwroot, then choose the c:\inetpub\wwwroot directory for your Web projects.

### **Opening Non-Web Projects from Source Control**

Complete the following steps to add non-Web projects from Version Manager to an open solution. You can also open single web projects from Version Manager into new solutions. For instructions on this, see ["Opening Web Projects from Source Control" on page 173](#).

**IMPORTANT!** If you add a project from Version Manager to a solution that is already under source control, **you must save your solution and then check your solution in to Version Manager after you have added the project to it**. This ensures that the version of the solution that includes the new project is stored in Version Manager, and that all users can get this update. One fast way to do this is to commit all local changes to Version Manager. See ["Committing Local Changes to Version Manager" on page 197](#).

**To open a non-Web project from Version Manager:**

- 1 Open or create the Solution to which you will add the project.
- 2 Select File | Source Control | Open Project from Source Control. The Open from Source Control wizard appears.
- 3 Specify the project database that contains the project that you want to open. Do any of the following to specify a project database:
  - Enter the path to the root project database directory, or select a recent project database from the drop-down list.
  - Click the **Browse for PDB** button  to browse for a project database. In this case, you must select any of the .ser files located directly under the root PDB directory.
  - Click the **Browse File Servers** button  to choose from all project databases on the Version Manager file server. File servers must be defined from the Version Manager desktop client. *If no Version Manager file server is defined, ignore this button.*
- 4 Enter your Version Manager user name and password.
- 5 Click **Next**. The Workspace page of the wizard appears.
- 6 Select the desired workspace. Consider the following:
  - Your choice of workspace determines the default version label and promotion group for the files. Only those files to which the default version label is assigned will be opened to your local workspace. Later on, when synchronizing your local workspace with Version Manager, the default version label will be assigned to any new files that you add. See ["Using Workspaces" on page 154](#).
  - The choice of workspace also determines which project files you can open. You can open projects to which the default version label is assigned. For example, if the default version label is "branch," you will only be able to open a project file to which the "branch" version label is assigned.
- 7 Click **Next**. The Create Project page of the wizard appears.
- 8 Browse the project database folders to find and select the specific project file that you want to open locally. All project files to which your default version label is assigned are displayed. If your default version label, as determined by your workspace selection, is not assigned to the latest revision of any project files, then no project files will be visible to you.
- 9 In the **Workfile location** field, specify the local workspace location. This is the local work directory, where your working copies of the files in the solution will be stored.
- 10 Select the **Add to Solution** option to add the project to the solution currently open in Visual Studio. Else, the **Close Solution** option will close the current solution and create a new solution for the project.
- 11 Click **Next**. The SBM page of the wizard appears.
- 12 Enter the SBM server name in the **SBM Server** field. If the SBM server uses a non-default port number (any port except 80), append the port number to the server name. For example, if the port number is 89:

tt\_server:89

- 13 Enter your SBM user name and password.
- 14 Click **Next**. The Review page of the wizard appears.
- 15 Review the choices you made. Click the **Back** button to change any of the settings. When you are finished, click the **Finish** button.
- 16 Before you start working on any files:
  - a Save the solution.
  - b Close the solution, then reopen the solution.
  - c If the solution is under source control, you should also check in the solution to make sure that other users of the same solution will get the version of the solution with the project in it. One fast way to do this is to commit all local changes to Version Manager. See ["Committing Local Changes to Version Manager" on page 197](#).

### ***Opening Web Projects from Source Control***

Follow these steps to open a single web project into a new, blank solution. It is simpler to open an entire solution at a time. These steps, however, allow you to open and work with just a specific Web project, if you don't want to work with all projects in a solution.

#### **To open a Web project from Source Control:**

- 1 Select File | Open | Web Site. The Open Web Site dialog box appears.
- 2 Select **Source Control** in the left pane.
- 3 Click the **Select Source Control Project** button. The Open Web Project from Source Control wizard appears.
- 4 Complete the steps under ["Opening Solutions from Source Control" on page 170](#), but with the following changes on the Create Project page of the wizard:
  - Select the specific web project from the folder tree. **Do not select a solution.**
  - In the **Workfile location** field, enter a path to a new folder under your root IIS directory. For example, if your IIS server stores all Web files under the C:\Inetpub\wwwroot directory and your project is called MyWebApp, browse to select the following complete path:  
  
C:\Inetpub\wwwroot\MyWebApp  
  
**It is strongly recommended that you name this folder after the project name.**
- 5 Complete the wizard. Once the wizard is complete, the Open Web Site dialog box reappears.
- 6 To run the web project as an IIS web site, select the **Run as IIS web site** checkbox. Else, by default, it will run as a file-based web site.
- 7 Click the **Open** button.

## Opening Solutions not Added using RIDE

You can open a Visual Studio solution from Source Control if it was added to the project database using a different client to the Visual Studio RIDE, for example, the VM GUI or VM I-Net clients.

**NOTE** If you do not follow the procedure below this message is displayed.

Unable to detect relative root directory.

"Root directory for selected item is not detected. Possible it is added not through RIDE."

- 1 Select **File | Open | Project/Solution**.
- 2 Select the Visual Studio solution and open it *without* any Source Control integration.
- 3 Do one of the following:
  - Select **File | Source Control | Change Binding**.
  - Right-click the solution and select **Change Binding**.
- 4 In the Change Source Control wizard, to select the project database containing the solution that you want to open, do one of the following:
  - Enter the path to the root project database directory or select a project database from the list.
  - Click **Browse for PDB** and browse for a project database. Select any of the .ser files located directly under the root PDB directory.
  - (Only applicable if a Version Manager file server is defined) Click **Browse File Servers** and select a project database on the Version Manager file server.
- 5 Click **Next**. Enter your Version Manager user name and password.
- 6 Click **Next**. Select the workspace that you will use for all projects in the solution.

**NOTE:** The workspace determines the default version label and promotion group for the files. Only deleted files to which the default version label is assigned will open in your local workspace. When you synchronize your local workspace with Version Manager, the default version label is assigned to any new files that you add. See ["Using Workspaces" on page 154](#). The workspace also determines which solution files you can open. You can open a solution to which the default version label is assigned. For example, if the default version label is "branch," you can only open a solution file to which the "branch" version label is assigned.
- 7 Click **Next**. Browse the project database folders and select the solution file (.sln) that corresponds to the Visual Studio solution that is open.
- 8 Click **Next**, review your choices, and click **Finish**.

## Editing Files

Complete the procedures in this section to:

- Review file history. See ["Reviewing File History" on page 175](#).
- Get files and folders. See ["Getting Specific Files or Folders" on page 176](#).
- Check specific files and folders out. See ["Checking Out Files" on page 178](#).

- Undo checkout. See ["Undoing Checkout" on page 180](#).
- Edit files. See ["Editing Files" on page 181](#).
- Review local changes. See ["Reviewing Local Changes" on page 181](#).
- Check specific files in. See ["Checking In Files" on page 182](#).
- Label files. See ["Labeling Revisions" on page 186](#).
- Promote files. See ["Promoting Revisions" on page 188](#).
- Work offline. See ["Working While Offline" on page 188](#).

## Reviewing File History

Review file history to display information about a file's revisions, including check-in date and comment, author, and assigned version labels and promotion groups.

### ***Reviewing File History***

#### **To review file history:**

- 1** Select the file in the Solution Explorer.
- 2** Right-click, and select **View History** from the resulting menu. The History view appears.

3 From the History view toolbar, you can do any of the following:

#### Action Buttons

- 1 **Compare Revisions:** Compares two selected revisions, or compares your local file to the latest revision in Version Manager. To compare two revisions, CTRL-click the two revisions, then click the button. See ["Comparing Files" on page 199](#).
- 2 **View Selected Revision:** Copies the revision to a temporary location and opens it in Visual Studio. You cannot edit the file, only view it.
- 3 **Assign Label:** Assigns a new version label to the selected revision. See ["Assigning Labels to the Latest Revision" on page 186](#).
- 4 **Promote to Next:** Promote the selected revision to the next promotion group. See ["Promoting Revisions" on page 188](#).
- 5 **Check Out Selected Revision:** Check out the selected revision. See ["Checking Out Files" on page 178](#).  
Note, if you check out a non-tip revision, a branch will be created when you check it in. See ["Working with Branches" on page 161](#).
- 6 **Get Selected Revision:** Get a copy of the selected revision, which you can then work on and, if you use a pragmatic locking model, check in. See ["Getting Specific Files or Folders" on page 176](#).

#### Display Buttons

- 7 **Revision History:** Lists all revisions and branches of the file.
- 8 **Labels:** Lists each version label assigned to a revision of the file. You can click on specific label names in order to delete or rename them.
- 9 **Promotion Groups:** Lists each promotion group assigned to a revision of the file.
- 10 **Show Labels:** Shows/hides multiple version labels for the selected revision.
- 11 **Show Comments:** Shows/hides full-length check-in comments for the selected revision.
- 12 **Refresh:** Refreshes the History view.

## Getting Specific Files or Folders

Get files to copy any files that you need to work on to your local workspace. You can get the latest revisions of files, or specific revisions from specific branches.

Once you have gotten the files, you can edit them and check them in (unless your organization requires that you check out files before you can edit them; see ["Collaborative Process Overview" on page 153](#)). The local copies of the files are set to read-only; to edit them, you must either set them to be writable, or accept the prompts in Visual Studio to overwrite the existing files when you save.

From the Compare Workspaces view, you can also update your local workspaces with all new revisions of files, as well as new files and other changes to the Version Manager project. See ["Getting All Updates from Version Manager" on page 195](#).

[Getting Latest Revisions](#)

[Getting Previous Revisions or Getting from Specific Branches](#)



## Getting Latest Revisions

If you already have local copies of the files, you can optionally merge the latest revisions you are getting with your existing local copies. This ensures that any changes you have made locally are preserved. If any conflicts result from the merge attempt, the get will fail. Resolve any conflicts, and then get the files. See ["Reviewing and Resolving Conflicts" on page 201](#).


### To get the latest revisions of files:

- 1 In the Solution Explorer, select the files or folders that you want to get.
- 2 Right-click, and select **Get Latest Revision** from the resulting menu. The Get Latest dialog box appears.
- 3 Verify that the file selection is correct, and modify it if necessary.
- 4 To compare a local workfile to the latest revision in Version Manager, select a file and click the **Compare** button. The file opens in the Compare Revisions window. See ["Comparing Files" on page 199](#).
- 5 To override the default get options, select the **Options** tab and do any of the following:
  - Specify what to do if the local workfile has changed since the latest revision was checked in to Version Manager:
    - **Prompt:** Asks you what to do on a file-by-file basis.
    - **Merge:** Automatically merges the latest revision from Version Manager into your local workfile. If there are conflicting changes, the get will fail. Resolve any conflicts, and then get the files. See ["Reviewing and Resolving Conflicts" on page 201](#).
    - **Replace:** Overwrites the local workfile with the latest revision from Version Manager.
    - **Leave:** Leaves the local workfile as it is.
  - **Make writable:** Select this check box to make the workfile writable instead of read-only.  
The write attribute will be changed only if a revision is actually gotten from the repository. No revision is gotten if the local file matches the latest revision or if the local file is different but you selected the **Leave** option above.
  - **Save settings:** Select this check box to make these settings the new default.  
You can also define default options for all dialog boxes from the Tools | Options dialog box. See ["Setting Default Options for Dialog Boxes" on page 189](#)
- 6 Click the **Get** button.  
Depending on the options you chose, you may be prompted as to what to do with modified workfiles or to resolve auto-merge conflicts.

[Getting Previous Revisions or Getting from Specific Branches](#)

## Getting Previous Revisions or Getting from Specific Branches

To get a specific revision of a file:

- 1 Select the file in the Solution Explorer.
- 2 Right-click and select **View History** from the resulting menu. The History view appears.
- 3 Click the **Revision History** button  to display all revisions and branches. A checkmark in the left column indicates which revision is currently in the local workspace.
- 4 Right-click the revision that you want to get and select **Get Revision** from the resulting menu.
- 5 If the local workfile is newer than the revision you are getting, you are prompted to select one of the following options and click **OK**:
  - **Merge the workfile with the revision requested:** Automatically merges the selected revision from Version Manager into your local workfile. If there are conflicting changes, the get will fail. Resolve any conflicts, and then get the revision. See ["Reviewing and Resolving Conflicts" on page 201](#).
  - **Leave the workfile as is:** Leaves the local workfile as it is.
  - **Overwrite the workfile with the requested revision:** Overwrites the local workfile with the selected revision from Version Manager.

[Getting Latest Revisions](#)

## Checking Out Files

Check out files to lock the latest revision (or specific revision) of specific files in Version Manager, and copy the files to your local workspace. When you check out files:

- The working copies of the files are set to be writable, so that you can edit and save changes to them.
- No other user can modify that file until you check it in with your changes, or undo the checkout. This prevents other users from making changes to the file that might conflict with your changes.
- If you already have local copies of the files, you can optionally merge the repository revisions with your local copies. This ensures that any changes you have made locally are preserved. If any conflicts result from the merge attempt, the checkout will fail. Resolve any conflicts, and then check out the files. See ["Reviewing and Resolving Conflicts" on page 201](#).
- If you lock a non-tip revision (any revision other than the latest revision on a given trunk or branch), Version Manager will create a new branch when you check it in. See ["Working with Branches" on page 161](#).

**To check out a file:**

- 1 Do one of the following:

To check out	Do this
The default revision  (as defined by the default version [label] or promotion group for the workspace; else the tip of the trunk)	<ol style="list-style-type: none"> <li>a Select a project, folder, or individual files in the Solution Explorer.</li> <li>b Right-Click and select <b>Check Out</b> from the resulting menu. The Check Out dialog box appears.</li> <li>c Continue to Step 2.</li> </ol>
A specific revision	<ol style="list-style-type: none"> <li>a Select the desired file in the Solution Explorer.</li> <li>b Right-Click and select <b>History</b> from the resulting menu. The History view displays a list of all revisions of the file.  The blue checkmark in the History view indicates which revision is currently in your local workspace.</li> <li>c Click the nodes to expand branches in the tree, as needed, in order to locate the desired revision.</li> <li>d Right-Click on the desired revision and select <b>Check Out Revision</b> from the resulting menu.  A red checkmark appears next to the file in the Solution Explorer and a blue checkmark appears next to the revision in the History view. The file is checked out.</li> <li>e You are done. Skip the remainder of this procedure.</li> </ol>

- 2 Verify that the file selection is correct, and modify it if necessary.
- 3 To compare a local workfile to the latest revision in Version Manager, select a file and click the **Compare** button. The file opens in the Compare Revisions window. See ["Comparing Files" on page 199](#).
- 4 To override the default check out options, select the **Options** tab and do any of the following:
  - Specify what to do if the local workfile has changed since the latest revision was checked into Version Manager:
    - **Prompt:** Asks you what to do on a file-by-file basis.
    - **Merge:** Automatically merges the latest revision from Version Manager into your local workfile. If there are conflicting changes, the check out will fail. Resolve any conflicts, and then check out the files. See ["Reviewing and Resolving Conflicts" on page 201](#).
    - **Replace:** Overwrites the local workfile with the latest revision from Version Manager.
    - **Leave:** Leaves the local workfile as it is.

- **Save settings:** Select this check box to make these settings the new default. You can also define default options for all dialog boxes from the Tools | Options dialog box. See ["Setting Default Options for Dialog Boxes" on page 189](#)

5 Click the **Check Out** button.

Depending on the options you chose, you may be prompted as to what to do with modified workfiles or to resolve auto-merge conflicts.

## Undoing Checkout

Undo checkout in order to make them available to other users to check out. You can optionally restore your local copy of the file to the state it was in before you checked it out, replace it with the latest revision from Version Manager, or leave it as it is.

**To undo checkout:**

1 From the Solution Explorer, select one of the following:

Select	To
A project or folder	Undo checkout of all files stored in the project or folder
Individual files	Undo checkout of specific files

- 2 Right-click, and select Undo Checkout from the resulting menu. The Undo Checkout dialog box appears. You can also undo checkout by right-clicking files in the File Status view (View | File Status).
- 3 Verify that the file selection is correct, and modify it if necessary.
- 4 To compare a local workfile to the latest revision in Version Manager, select a file and click the **Compare** button. The file opens in the Compare Revisions window. See ["Comparing Files" on page 199](#).
- 5 To override the default undo checkout options, select the **Options** tab and do any of the following:
- Specify what to do with the local workfile after the file is unlocked:
    - **Leave:** Leaves the local workfile as it is.
    - **Replace with latest:** Overwrites the local workfile with a read-only copy of the latest revision from Version Manager.
  - **Save settings:** Select this check box to make these settings the new default. You can also define default options for all dialog boxes from the Tools | Options dialog box. See ["Setting Default Options for Dialog Boxes" on page 189](#)
- 6 Click the **Undo Checkout** button.

## Editing Files

Once you have gotten or checked out the files you want to work on, you can edit them locally. Depending on the workflow in your organization, you must do one of the following to edit the files:

- If your organizational workflow is to check out (lock) files before editing them, then any files you have checked out are already writable and you can start work on them. If the files are not yet checked out, you must check them out.
- If your organization supports an optimistic locking model, allowing multiple developers to work on the same files simultaneously without requiring the files to be locked, then you do not need to check out the files before editing them. You just need to get them. However, getting files does not by default make the local copies of the files writable. In order to work on the files, you must do one of the following:
  - Select the **Make writable** option when you get the files from Version Manager. The write attribute will be changed only if a revision is actually gotten from the repository. No revision is gotten if the local file matches the latest revision or if the local file is different but you selected the **Leave** option.
  - Use Windows Explorer to make the files writable.
  - Choose to overwrite the existing files when prompted by Visual Studio. This occurs when you save your changes.

## Refreshing File Status

Refresh file status in the Solution Explorer to display the current file lock status on the file icons. This ensures that you can see which files are currently locked in Version Manager.

### To refresh file status:

- 1 In the Solution explorer, click the Refresh button.

## Reviewing Local Changes

Review local changes to:

- Display a list of all changes that you have made to local copies of files, and to list any files that you have added or removed. This is an ideal way to quickly review the status of the files in your workspace.
- Quickly check in all files that you have changed since you last checked in. See ["Checking In All Local Changes" on page 185](#).

### To display local file status:

- 1 Select View | File Status. The File Status view appears with a list of all of the files that you have modified or added since your last check in.
- 2 To change which files are listed, click the **Filter** button and select or deselect any or all of the following options:
  - **Checked Out:** List the files that are currently checked out.
  - **Locally Modified:** List the files that you have modified since your last check in.

- **New Files:** List the files that you have added since your last check in. You can also check in changes and compare revisions from the File Status view. See ["Checking In All Local Changes" on page 185](#) and ["Comparing Files" on page 199](#)

## Checking In Files

Check in files to store any changes you have made in new revisions. If your workflow is to check out (lock) all files in order to edit them, then you must do so before checking them in. Otherwise, you can check in any local files, regardless of whether you first checked them out.

When you check in a file, a new revision is created in one of the following places, depending on how your Version Manager workspace is configured and whether a non-tip revision is checked out:

If	Then
No revision is locked, AND A Default Version has been defined for the workspace and applied to the file.	The new revision is checked in as the tip on the branch or trunk that is defined by the Default Version.
No revision is locked, AND No Default Version is in effect for the file.	The new revision is checked in as the tip on the trunk.
A single revision is locked	The new revision is checked in as the tip on the branch or trunk that contains the locked revision.
Multiple revisions are locked	You are prompted to select which revision to unlock. The new revision is created as the tip on the branch or trunk that you unlock.
A non-tip revision is locked	The new revision is checked in as the tip of a new branch off of the locked revision.

See ["Working with Branches" on page 161](#) for information on branching.

To check in, you can either:

- Select specific files and folders, and complete the Check In dialog box. This provides you with a full range of check-in options, including the ability to customize SBM issue associations. See ["Checking In with the Check In Dialog Box" on page 183](#).
- Check in locally modified files from the File Status view, and bypass the Check In dialog box. In this case, all default settings for check-in will apply. This method is faster, but less flexible. See ["Checking In All Local Changes" on page 185](#)

**IMPORTANT!** Checking in does not synchronize such changes as renamed and moved files. To do this, commit all local changes to Version Manager (["Committing Local Changes to Version Manager" on page 197](#)), or synchronize your workspace with the corresponding Version Manager workspace.

### About Checking In and Merging

If other users have checked in changes to the files since you last updated your local workspace with the latest revisions, by default your changes will be automatically merged with the latest revisions. This ensures that no changes are lost. If your changes conflict with changes that other users have checked in, you must resolve the conflicts before

completing check-in. The Compare Workspaces view simplifies this process by listing all of the files that conflict with files in Version Manager. See ["Reviewing and Resolving Conflicts" on page 201](#).

### **Checking In with the Check In Dialog Box**

**To check in specific files or folders using the Check In dialog box:**

- 1** Save all of the files that you are checking in, including any project or solution files.
- 2** Select the files or folders that you want to check in from the Solution Explorer. If you locked a non-tip revision (any revision other than the latest revision on a given trunk or branch), Version Manager will create a new branch when you check it in. See ["Working with Branches" on page 161](#).
- 3** Right-click, and select **Check In** from the resulting menu. The Check In dialog box appears.
- 4** Verify that the file selection is correct, and modify it if necessary.
- 5** Enter a description of the changes in the **Description** field. To enter a separate description for each file in turn, deselect the **Use description for all** checkbox on the **Options** tab.
- 6** Your currently activated SBM issues will be listed. By default, they will be associated with the files you check in. Deselect any issues that you do not want to associate with the files you are checking in.

By default, any issues that you associate with files at check-in will not be deactivated. If you want to remove associated issues from your activated issues list, select the **Deactivate selected issues after checkin** check box.

For information on SBM issue association and setup, see ["Associating and Working on SBM Issues" on page 204](#).

- 7 To modify your check-in options, click the **Options** tab and update any of the following:

Option	Description
<b>Newer file is checked in</b>	<p>Choose what to do if a new revision of the file has been checked in to Version Manager since you last got the file. Select from the following:</p> <ul style="list-style-type: none"> <li>■ <b>Merge:</b> Merges your changes with the latest revision in Version Manager. If conflicts result from attempting to merge files, you must resolve the conflicts before completing the check-in. See <a href="#">"Reviewing and Resolving Conflicts" on page 201</a>.</li> <li>■ <b>Leave:</b> Does not check in the file. The file is unlocked.</li> <li>■ <b>Force check in:</b> Checks in the file and creates a new revision based solely on your workfile.</li> </ul>
<b>If file is unchanged</b>	<p>Choose what to do if the local workfile is no different than the latest revision in Version Manager:</p> <ul style="list-style-type: none"> <li>■ <b>Prompt:</b> Displays a prompt allowing you to decide what to do with each unchanged workfile in turn.</li> <li>■ <b>Check in:</b> Checks in any unchanged workfiles as new revisions.</li> <li>■ <b>Leave:</b> Does not check in any unchanged workfiles; no new revisions are created. The file is unlocked.</li> </ul>
<b>Use description for all</b>	Select to apply the description from the <b>Description</b> field to every file. Deselect to enter unique descriptions for each file in turn.
<b>Keep checked out</b>	Select to keep any currently locked files locked after the check-in operation completes.
<b>Update after keyword expansion</b>	<p>This option applies only if the file you are checking in includes Version Manager keywords that will be expanded during check-in.</p> <p>Select to copy the latest revision, after its keywords have been expanded, to your local workspace.</p>
<b>Promote</b>	<p>Select to promote the new revision to the next group in the promotion hierarchy.</p> <p><b>IMPORTANT!</b> You cannot simultaneously keep the files locked and promote the new revisions during check-in. This is because you cannot promote a locked revision. If you want to promote the new revisions, then do not choose to keep the files locked.</p>
<b>Force branch</b>	<p>Select to force a new branch in the file's archive. The new branch will be created on whatever branch is defined by the default label. If there is no default label, the default branch is usually the tip of the trunk or branch from which the revision was checked out.</p> <p>For more information on branches, see <a href="#">"Working with Branches" on page 161</a>.</p>



Option	Description
<b>Label: Name</b>	Enter a version label to assign to the new revisions. Labels are limited to 254 characters. Do not use a colon (:), double quotes ("), a plus sign (+), or a minus sign (-).
<b>Label: If label exists</b>	Choose what to do if the label you are assigning to the new revision is already assigned to a different revision of the same files. Select one of the following: <ul style="list-style-type: none"> <li>■ <b>Prompt:</b> Displays a prompt allowing you to decide what to do with each file in turn.</li> <li>■ <b>Reassign:</b> Moves the label to the new revision.</li> <li>■ <b>Leave:</b> Leaves the label assignment as it is; does not move it to the new revision.</li> </ul>
<b>Float with tip</b>	Select if you want the label to always be associated with the latest (tip) revision of a file. A floating label will automatically reassign itself to the latest revision during any future check-ins.
<b>Save settings</b>	Select to make these settings the new default. <b>TIP</b> You can also define default options for all dialog boxes from the Tools   Options dialog. See <a href="#">"Setting Default Options for Dialog Boxes" on page 189</a>
<b>Deactivate selected issues after check in</b>	Select to deactivate any issues that you associate with files at check-in; they will no longer appear in your activated issues list. If you want to leave the issues in your activated issues list, deselect this check box. <b>NOTE</b> This setting is remembered <b>independently</b> of the <b>Save settings</b> option.

- 8 Click the **Check In** button.

Depending on the options you chose, you may be prompted as to what to do in the case of an existing label, an unchanged workfile, a newer revision in the repository, or to resolve auto-merge conflicts. If multiple revisions of the file are locked, you will be prompted to select which revision to unlock.

### Checking In All Local Changes

#### To quickly check in all local changes:

- 1 Select View | File Status. The File Status view appears with a list of all of the files that you have modified or added since your last check in.
- 2 Click the **Comments** button to expand the **Comments** field so that you can enter a description of the changes you made to the files. To enter a separate description for each file, leave the **Comments** field blank. A Comments dialog will appear for each file in turn.
- 3 Verify that the file selection is correct, and modify it if necessary.
- 4 Click the **Check In** button.

If SBM issue associations are required for all check-ins, and there are currently active issues, then the files are associated with the issues. If no issues are currently active, you must activate some issues before checking in. See ["Associating and Working on SBM Issues" on page 204](#)

## Labeling Revisions

You can use version labels to mark revisions that have been used in specific releases, or that include specific changes. For example, you can assign a floating label (a label that always moves to the newest revision of a file) that build scripts can search for, in order to be sure that only the latest revisions of files are retrieved. You can:

- Assign labels: See ["Assigning Labels to the Latest Revision" on page 186](#) and ["Assigning Labels to a Previous Revision" on page 187](#).
- Rename labels: See ["Renaming Labels" on page 187](#)
- Delete labels: See ["Deleting Labels" on page 188](#)

### **Assigning Labels to the Latest Revision**

**To label the latest revision of a file, or a selection of files:**

- 1 Select the files that you want to label. Select a folder to label all of the files within that folder.
- 2 Right-click, and select **Label** from the resulting menu. The Assign Label dialog box appears.
- 3 Verify that the file selection is correct, and modify it if necessary.
- 4 To compare a local workfile to the latest revision in Version Manager, select a file and click the **Compare** button. The file opens in the Compare Revisions window. See ["Comparing Files" on page 199](#).
- 5 Enter the label in the **Label** field. Labels are limited to 254 characters. Do not use a colon (:), double quotes ("), a plus sign (+), or a minus sign (-).
- 6 To override the default label options, select the **Options** tab and do any of the following:
  - Choose what to do if the label you are assigning is already assigned to different revisions of the same files. Select one of the following:
    - **Prompt:** Displays a prompt allowing you to decide what to do with each file in turn.
    - **Reassign:** Moves the label to the latest revision.
    - **Leave:** Leaves the label assignment as it is; does not move it to the latest revision.
  - **Float label:** Select this check box if you want the label to always be associated with the latest (tip) revision of a file. A floating label will automatically reassign itself to the latest revision during any future check-ins.
  - **Save settings:** Select this check box to make these settings the new default. You can also define default options for all dialog boxes from the Tools | Options dialog box. See ["Setting Default Options for Dialog Boxes" on page 189](#)
- 7 Click the **Assign Label** button.

[Assigning Labels to a Previous Revision](#)

## Assigning Labels to a Previous Revision

### To label a previous revision of a specific file:


- 1 Select the file that you want to label.
- 2 Right-click, and select **View History** from the resulting menu. The History view appears.
- 3 Right-click the revision that you want to label and select **Assign Label** from the resulting menu. The Assign Label dialog box appears.
- 4 To compare a local workfile to the latest revision in Version Manager, select a file and click the **Compare** button. The file opens in the Compare Revisions window. See ["Comparing Files" on page 199](#).
- 5 Enter the label in the **Label** field. Labels are limited to 254 characters. Do not use a colon (:), double quotes ("), a plus sign (+), or a minus sign (-).
- 6 To override the default label options, select the **Options** tab and do any of the following:
  - Choose what to do if the label you are assigning is already assigned to a different revision of the same file. Select one of the following:
    - **Prompt:** Displays a prompt allowing you to decide what to do.
    - **Reassign:** Moves the label to the selected revision.
    - **Leave:** Leaves the label assignment as it is; does not move it to the selected revision.
  - **Float label:** Select this check box if you want the label to always be associated with the latest (tip) revision of a file. A floating label will automatically reassign itself to the latest revision during any future check-ins.
  - **Save settings:** Select this check box to make these settings the new default. You can also define default options for all dialog boxes from the Tools | Options dialog box. See ["Setting Default Options for Dialog Boxes" on page 189](#)
- 7 Click the **Assign Label** button.

### [Assigning Labels to the Latest Revision](#)

## Renaming Labels

From Visual Studio, you can rename a label assigned to a specific revision of a specific file. To rename a label across multiple files, use the Version Manager desktop client.


### To rename a label:

- 1 Select the file that contains the label you want to rename.
- 2 Right-click, and select **View History** from the resulting menu. The History view appears.
- 3 Click the **Label** () button to display all version labels assigned to revisions of the file.
- 4 Click on the label that you want to rename, enter the new text, and press ENTER.

## Deleting Labels

From Visual Studio, you can delete a label assigned to a specific revision of a specific file. To delete a label across multiple files, use the Version Manager desktop client.

### To delete a label:

- 1 Select the file that contains the label you want to delete.
- 2 Right-click, and select **View History** from the resulting menu. The History view appears.
- 3 Click the **Label** () button to display all version labels assigned to revisions of the file.
- 4 Right-click the label that you want to delete, and select **Delete Label** from the resulting menu.

## Promoting Revisions

Promote a revision to associate it with the next promotion group in the promotion model. For example, if the next group in your promotion model is called *test* and is used to identify files that are ready to be tested, and you check in a revision that makes a component ready to be tested, then promote the revision to assign the *test* group to it.

### Promoting a Revision

#### To promote a specific revision of a specific file:

- 1 Select the file that you want to promote.
- 2 Right-click, and select **View History** from the resulting menu. The History view appears.
- 3 Right-click on the revision that you want to promote, and select **Promote to Next** from the resulting menu.

You can promote files during check in. See ["Checking In Files" on page 182](#).

## Working While Offline

You can continue to work on projects while offline, if:

- The files are already on your local drive, and
- Pragmatic locking is enabled, or the files are already checked out to you.

If you attempt to login into a project database while the system hosting the project database is unavailable, a **Work Offline** button appears at the bottom of the Log In dialog. Click the button to initiate an offline session.

While you are working offline:

- The following message is displayed in the console: The session for PDB *ProjectDatabaseName* is currently offline.
- All Version Manager commands that require a connection to the project database are disabled.

Version Manager will automatically reconnect to the project database when it detects that the system is accessible again.

## Setting Default Options for Dialog Boxes

Set default options to define the default settings and behavior for dialog boxes. Silent operations, such as checking in from the File Status view, use these defaults as well.

### To set default options for dialog boxes:

- 1 Select Tools | Options. The Options dialog box appears.
- 2 From the tree on the left, select Source Control | Commands.
- 3 Set any of the following default options to your choice:

Category	Options
<b>Label</b> (Click the <b>Label</b> button in the <b>Check in</b> group to display these options)  Defaults apply whenever you assign version labels. See <a href="#">"Labeling Revisions" on page 186</a> .	<b>If label exists</b> Choose what to do if the label you are assigning is already assigned to a different revision of the same file. Select one of the following: <ul style="list-style-type: none"> <li>■ <b>Prompt:</b> Displays a prompt allowing you to decide what to do with each file in turn.</li> <li>■ <b>Reassign:</b> Moves the label to the new revision.</li> <li>■ <b>Leave:</b> Leaves the label assignment as it is; does not move it to the new revision.</li> </ul>
	<b>Float with tip</b> Select if you want the label to always be associated with the latest (tip) revision of a file. A floating label will automatically reassign itself to the latest revision during any future check-ins.
	<b>Name</b> Enter a version label to assign by default. Labels are limited to 254 characters. Do not use a colon (:), double quotes ("), a plus sign (+), or a minus sign (-).

Category	Options
<b>Check in</b> Defaults apply whenever you check in files. See <a href="#">"Checking In Files"</a> on page 182.	<b>If newer is checked in</b> Choose what to do if a new revision of the file has been checked in to Version Manager since you last got the file. Select from the following: <ul style="list-style-type: none"> <li>■ <b>Merge:</b> Merges your changes with the latest revision in Version Manager. If conflicts results from attempting to merge files, you must resolve the conflicts before completing the check-in. See <a href="#">"Reviewing and Resolving Conflicts"</a> on page 201.</li> <li>■ <b>Leave:</b> Does not check in the file. The file is unlocked.</li> <li>■ <b>Force check in:</b> Checks in the file and creates a new revision based solely on your workfile.</li> </ul>
	<b>If file is unchanged</b> Choose what to do if the local workfile is no different than the latest revision in Version Manager: <ul style="list-style-type: none"> <li>■ <b>Prompt:</b> Displays a prompt allowing you to decide what to do with each unchanged workfile in turn.</li> <li>■ <b>Check in:</b> Checks in any unchanged workfiles as new revisions.</li> <li>■ <b>Leave:</b> Does not check in any unchanged workfiles; no new revisions are created. The file is unlocked.</li> </ul>
	<b>Use description for all</b> Select to apply the description from the <b>Description</b> field to every file. Deselect to enter unique descriptions for each file in turn.
	<b>Keep checked out</b> Select to keep any currently locked files locked after the check-in operation completes.
	<b>Update after keyword expansion</b> This option applies only if the file you are checking in includes Version Manager keywords that will be expanded during check-in. Select to copy the latest revision, after its keywords have been expanded, to your local workspace.
	<b>Promote</b> Select to promote the new revision to the next group in the promotion hierarchy. <b>IMPORTANT!</b> You cannot simultaneously keep the files locked and promote the new revisions during check-in. This is because you cannot promote a locked revision. If you want to promote the new revisions, then do not choose to keep the files locked.

Category	Options
<b>Get latest</b> Defaults apply whenever you get files. See <a href="#">"Getting Specific Files or Folders"</a> on page 176.	<b>If file has changed</b> Specify what to do if the local workfile has changed since the latest revision was checked into Version Manager: <ul style="list-style-type: none"> <li>■ <b>Prompt:</b> Asks you what to do on a file-by-file basis.</li> <li>■ <b>Merge:</b> Automatically merges the latest revision from Version Manager into your local workfile. If there are conflicting changes, the get will fail. Resolve any conflicts, and then get the files. See <a href="#">"Reviewing and Resolving Conflicts"</a> on page 201.</li> <li>■ <b>Replace:</b> Overwrites the local workfile with the latest revision from Version Manager.</li> <li>■ <b>Leave:</b> Leaves the local workfile as it is.</li> </ul>
	<b>Make writable</b> Select this check box to make the workfile writable instead of read-only.
<b>Check out</b> Defaults apply whenever you check out files. See <a href="#">"Checking Out Files"</a> on page 178.	<b>If file has changed</b> Specify what to do if the local workfile has changed since the latest revision was checked into Version Manager: <ul style="list-style-type: none"> <li>■ <b>Prompt:</b> Asks you what to do on a file-by-file basis.</li> <li>■ <b>Merge:</b> Automatically merges the latest revision from Version Manager into your local workfile. If there are conflicting changes, the get will fail. Resolve any conflicts, and then get the files. See <a href="#">"Reviewing and Resolving Conflicts"</a> on page 201.</li> <li>■ <b>Replace:</b> Overwrites the local workfile with the latest revision from Version Manager.</li> <li>■ <b>Leave:</b> Leaves the local workfile as it is.</li> </ul>
	<b>After file is unlocked</b> Specify what to do with the local workfile after the file is unlocked: <ul style="list-style-type: none"> <li>■ <b>Leave:</b> Leaves the local workfile as it is.</li> <li>■ <b>Replace with latest:</b> Overwrites the local workfile with a read-only copy of the latest revision from Version Manager.</li> </ul>

#### 4 Click **OK**.

You can save the current configuration of a dialog box as the default by selecting the **Save Settings** check box on the dialog box's Options tab.

## Configuring Client/Server-Side Processing

By default, server-side processing is enabled. In most cases, this allows faster operation since less data needs to be transferred back and forth between the Version Manager File Server and your IDE client. However, you can revert to the traditional client-side

processing if your use case requires it. For instance, this feature does not currently support client-side event triggers.

**To configure Client/Server-Side processing:**

- 1 Select Tools | Options. The Options dialog box appears.
- 2 From the tree on the left, select Source Control | Client/Server.
- 3 Select or deselect the **Enable server-side processing** checkbox.
- 4 Click **OK**.

## Setting Encoding and Display Options

Set encoding and display options for the Compare and Merge tools to define:

- How text in file comparisons will appear, including markup for each type of difference (such as changes, deletions, and conflicts)
- How text in file comparisons will be encoded
- Whether line numbers should be displayed
- How many spaces tabs should take up
- Whether, when resolving conflicts, to display the original file (ancestor), as well as the derivatives that are in conflict

**To set encoding and display options for file comparisons:**

- 1 Select Tools | Options. The Options dialog box appears.
- 2 From the tree on the left, select Source Control | Compare and Merge.
- 3 Complete the general settings as follows:

Option	Description
<b>Show ancestor</b>	Select to always show the ancestor (or <i>original</i> ) revision pane when resolving conflicts.
<b>Syntax highlighting</b>	Select to highlight keywords when comparing files. You can highlight syntax in the following types of files:  CSharp, CPP, Java, Visual Basic, HTML, XML, Java Script, VB Script, CSS, SQL, Python, .ini
<b>Tab size</b>	Enter the number of spaces that comprise a tab.
<b>Line numbers</b>	Select to display line numbers when comparing files.

- 4 Under the **Encoding** options, choose any of the following:
  - **UTF-8**: 8 bit Unicode.
  - **UTF-16**: 16 bit Unicode (*Unicode* typically refers to this encoding).
  - **UTF-16BE**: Big-endian Unicode.
  - **ASCII**: 7 bit characters.



- **UTF-7 or high-ASCII:** Variable length encoding, commonly used in e-mail.
- 5 Under the **Display** options, click the **Font**, **Text Color**, or **Background** buttons to modify the way that the text will display when you compare files. You can preview the text style in the **Text** box.
- 6 Also under the **Display** options, select a type of difference and click the **Custom** button in order to customize the markup of specific types of differences. You can customize the text style and highlighting for changes, deletions, insertions, and moves. You can customize text color for conflicts and filler.

For example, by default, deletions are marked up with a red strike-out. To change this, select **Delete**, click **Custom**, and then select new text properties from the resulting dialog box.
- 7 Click **OK**.

## Comparing and Synchronizing Workspaces

Synchronize workspaces to check in any changes to files in your local workspace, and to update your local workspace with changes that other users have checked into Version Manager. Synchronizing workspaces includes the following procedures:

- Compare your local workspace to the corresponding Version Manager project to determine what has changed since your last synchronization. See ["Comparing Workspaces" on page 194](#).
- Commit all changes to local files, to check in the changes to Version Manager. See ["Committing Local Changes to Version Manager" on page 197](#).
- Update your local workspace with all updates to files in Version Manager. See ["Getting All Updates from Version Manager" on page 195](#).
- Synchronize workspaces to commit all local changes to Version Manager, and update your local workspace with changes from Version Manager, all in one step. See ["Synchronizing Workspaces" on page 198](#).

## About the Merge Process

Files may be merged when checking in files, and when synchronizing workspaces. If you and others have made changes to a shared file without checking that file out, then your respective changes will be merged when you synchronize workspaces. For example, Developer 1 and Developer 2 both get Queries.cs in their local workspaces, without checking it out first. Developer 1 checks in a new revision with changes. Developer 2 then checks in with changes, as well. At this time, the changes that Developer 2 made are merged into the revision of the file that Developer 1 checked in. If there are any conflicts, Developer 2 must resolve them before completing his check-in.

## Important Refactoring Considerations

When you synchronize, your local workspace is automatically updated with any files that have been deleted, moved, renamed, or added within another Visual Studio environment. **However, if any files have been added, moved, renamed, or deleted from any of the other Version Manager clients (such as the desktop, Web, or command line**

**clients), your Visual Studio solution or project will not be automatically updated with all of the changes.** We therefore recommend that any additions, moves, deletions, or renames of files in any Visual Studio projects are completed within Visual Studio itself. This updates the Visual Studio solution and project information so that the changes will be available from within Visual Studio.

### IMPORTANT!

- All files of a given CAB or C++ project must reside under the root directory of the Visual Studio project. Cutting and pasting files from one C++ project to another inside of Visual Studio violates this requirement.
- A New Visual Studio project added to a solution that is already under source control will **not** be listed as having been locally added when you compare workspaces. To add such projects to source control, select the new project, right-click, and select **Add Project to Source Control**.

If any files are added, renamed, deleted, or moved from any other Version Manager interface, you must complete the following steps to reconcile your local workspace to the changes:

- 1 Synchronize your workspace. This will update your working directories with all of the changes. For example, if a file was renamed in Version Manager, the file with the old name will be replaced by the file with the new name. Any files that were deleted in Version Manager will be deleted from your working directories.
- 2 From the Solution explorer, delete any references to files that are now missing. This includes any file that has been renamed or moved. Because the Visual Studio solution or project are not aware of the changes, the files simply appear to be missing, rather than renamed or moved.
- 3 Add any files that have been renamed, moved, or added. These files were all placed in your physical working directories when you synchronized; you must add them to your Visual Studio solution or project to make sure that the solution or project are aware of them. Make sure that you add them to the location in your project that corresponds to their location in your working directories.
- 4 Once again synchronize your workspace with Version Manager. This will ensure that your solution or project is checked in to Version Manager, with all of the changes. Then, any other developer will get the updated solution or project with all of moved, renamed, added, and deleted files.

## Comparing Workspaces

Compare your local workspace to the corresponding Version Manager project to determine what has changed since you last synchronized. Once this is done, you can determine whether you need to commit local changes to Version Manager ("[Committing Local Changes to Version Manager](#)" on page 197), update your local workspace ("[Getting All Updates from Version Manager](#)" on page 195), or synchronize workspaces ("[Synchronizing Workspaces](#)" on page 198) to update and commit all changes. You can also determine whether you need to resolve any conflicting changes between different revisions of files.

Only projects that are under source control can be compared. However, the solution does not have to be under source control. This allows for the placement of projects outside of the solution structure.


Potential changes include:

- New files have been added, either to your local workspace or to Version Manager.
- Files have been changed in your local workspace.
- New revisions of files have been checked in to Version Manager.
- Files have been deleted, moved, or renamed, either in your workspace or in Version Manager.

**To compare workspaces:**

- 1 Make sure to save all files that you have modified locally, including solution and project files.
- 2 Select File | Source Control | Compare Workspaces. The Compare Workspace view appears.

By default, the contents of the local workspace appear in the left pane, under Local Workspace. The contents of the Version Manager project appear in the right pane, under Repository.

- 3 To limit the display to specific types of changes, click the **Filter** button and select any of the following filtering options:
  - Select / de-select **Outgoing** to show / hide changes to local files
  - Select / de-select **Incoming** to show / hide changes to files in Version Manager
  - Select / de-select **Conflicts** to show / hide conflicts between files in the local workspace and files in Version Manager
- 4 To open a conflict in the Merge tool, select the conflict and click the **Resolve conflicts** () button. See ["Reviewing and Resolving Conflicts" on page 201](#).

## Getting All Updates from Version Manager

Get all updates from Version Manager to update your local workspace with all changes in the corresponding Version Manager project. When you get updates from Version Manager, your workspace is updated with the following types of changes:


- All new revisions of files are copied to your workspace. By default, all changes in the new revisions are merged with changes that you have made to your local copies of the files since you last checked in. If your changes conflict with other users' changes, you must either:
  - Resolve the conflicts and update again.
  - Or force conflicting files to update, despite conflicts. In this case, your local changes will be overwritten with the latest revisions. The revisions are not merged, and your local updates are lost. See ["Forcing Updates" on page 196](#).
- Any new files, that have been added to Version Manager since you last got all updates, and to which the default version label for your workspace is assigned, are added to your local workspace.
- If any files have been deleted from Version Manager, the corresponding files are deleted from your local workspace. This includes any files from which the default version label has been removed.

- If any files in Version Manager have been moved or renamed, the corresponding files on your workspace are moved and renamed.

Only additions, deletions, moves, and renames that are made via the rich integration to Visual Studio are automatically synchronized. See ["Important Refactoring Considerations" on page 193](#) for information on refactoring changes made via other interfaces, such as the desktop.

### **Updating and Merging Local Workspaces**

#### **To update your local workspace:**

- 1 Select File | Source Control | Compare Workspaces. The Compare Workspaces view appears.
- 2 Compare your local workspace to the corresponding Version Manager project (see ["Comparing Workspaces" on page 194](#)). Of particular interest are the *Incoming* changes; these are all changes to the Version Manager project that will be updated in your local workspace.
- 3 Determine if any of the local files conflict with files in Version Manager. If so, you must resolve these conflicts before you can update your workspace with the new files. To open a conflict in the Merge tool, select the conflict and click the **Resolve conflicts** () button. See ["Reviewing and Resolving Conflicts" on page 201](#).

If there are conflicts, but you do not want to resolve them and would prefer to start over with new copies of the latest revisions from Version Manager, you can do so by forcing updates. See ["Forcing Updates" on page 196](#).

- 4 Once you have reviewed the changes, click the **Update** button to get all new and updated files from Version Manager.
- 5 Once the operation is complete, you can verify that you have successfully gotten all updates by again displaying all incoming changes, as well as conflicts.

### **Forcing Updates**

If you want to overwrite any local files with the latest revisions from Version Manager, you can do so by forcing an update. A forced update does not attempt to merge files. Any local modifications are over-written. This is very useful if there are conflicts that you do not need to resolve, and you would instead prefer to start over with the latest revisions.

#### **To force updates:**

From the Compare Workspace view, right-click the file (or files) and select Force Update from the menu that displays. Your local files are over-written with the latest revisions from Version Manager.

## Committing Local Changes to Version Manager

Commit local changes to Version Manager to update the Version Manager project with all changes to your local workspace. When you commit local changes, the Version Manager project is updated with the following types of change:

- Changes to files are checked in to new revisions. By default, all of your changes are merged with changes that other users have checked in since you last checked in. If your changes conflict with other users' changes, you must either:
  - Resolve the conflicts and commit again.
  - Force the conflicting files to commit, despite conflicts. In this case new revisions are created from your local files. The revisions are not merged. See ["Forcing Commits" on page 197](#).
- Any new files are added to the Version Manager project, and the default version label for your workspace is assigned to them.
- If you have deleted any files from your workspace, the corresponding files are deleted from Version Manager.
- If you have moved or renamed any files in your local workspace, the corresponding files are moved and renamed in Version Manager.

### ***Committing and Merging Local Changes***

#### **To commit local changes to Version Manager:**

- 1 Make sure to save all files that you have modified locally, including solution and project files.
- 2 Select File | Source Control | Compare Workspaces. The Compare Workspaces view appears.
- 3 Compare your local workspace to the corresponding Version Manager project (see ["Comparing Workspaces" on page 194](#)). Of particular interest are the *Outgoing* changes; these are all changes to your local workspace that will be committed to the Version Manager project.
- 4 Determine if any of the local files conflict with files in Version Manager. If so, you must resolve these conflicts before you can update your workspace with the new files. See ["Reviewing and Resolving Conflicts" on page 201](#).

If there are conflicts, but you do not want to resolve them and would prefer to check in new revisions from your files without merging them, you can do so by forcing a commit. See ["Forcing Commits" on page 197](#).

- 5 Once you have reviewed the changes, click the **Commit** button to update the Version Manager project.

If any of your local changes conflict with the latest revisions of files in Version Manager, you must resolve those conflicts before you can merge the files with the latest revisions. See ["Reviewing and Resolving Conflicts" on page 201](#).

### ***Forcing Commits***

If you want to create new revisions in Version Manager from your local copies of files, without merging them into the latest revisions, you can do so by forcing a commit. A forced commit does not attempt to merge files. This is very useful if there are conflicts

that you do not need to resolve, and would instead prefer to check in new revisions using your local files.

**To force commits:**

From the Compare Workspace view, right-click the file (or files) and select **Force Commit** from the menu that displays. New revisions are created in Version Manager.

## Synchronizing Workspaces

Synchronize workspaces to automatically complete all of the following actions:

- Check in any changes to files that you have modified locally. By default, all changes to your local copies of the files are merged with any changes that other users have checked in to Version Manager since you last synchronized (or committed changes). If your changes conflict with other users' changes, you must resolve them. See ["Reviewing and Resolving Conflicts" on page 201](#).
- Add any new files from your local workspace to the corresponding Version Manager project, and assign the default version label for your workspace to them.
- Get the latest revisions of all files in Version Manager. By default, all changes in the new revisions are merged with changes that you have made to your local copies of the files since you last checked in. If your changes conflict with other users' changes, you must resolve them. See ["Reviewing and Resolving Conflicts" on page 201](#).
- Get any files that have been added to Version Manager since you last synchronized, and to which the default version label has been assigned.
- Delete any files from Version Manager that have been deleted in your local workspace, and delete any files from your local workspace that have been deleted in Version Manager. This includes any files that haven't actually been deleted, but from which the default version label has been removed.
- Move or rename any files in Version Manager that you have moved or renamed locally, and move or rename any files locally that have been moved or renamed in Version Manager.

Only additions, deletions, moves, and renames that are made via the rich integration to Visual Studio are automatically synchronized. See ["Important Refactoring Considerations" on page 193](#) for information on refactoring changes made via other interfaces.

**To synchronize workspaces:**

- 1 Make sure to save all files that you have modified locally, including solution and project files.
- 2 Compare your local workspace to the Version Manager project to determine what changes you need to synchronize. See ["Comparing Workspaces" on page 194](#).
- 3 Optionally select / de-select changes to choose specific changes to synchronize.
- 4 Click the **Synchronize** button.

# Comparing Files and Resolving Conflicts

Using the rich integration to Visual Studio, you can compare and resolve conflicts between your local workfiles and the latest revisions in Version Manager. You can also compare two revisions of the same file. See the following sections for detailed information:

- See ["About File Comparison" on page 199](#) for information on the types of differences you can evaluate by comparing files.
- See ["Setting Encoding and Display Options" on page 192](#) to learn how to set display options for the file comparison tool, and how to set encoding options.
- See ["Comparing Files" on page 199](#) to learn how to compare files.
- See ["Reviewing and Resolving Conflicts" on page 201](#) to learn how to resolve conflicts between different revisions of files.

## About File Comparison

File comparison allows you to carefully evaluate the differences between a local copy of a file and the latest revision in Version Manager, or between two revisions of a file in Version Manager. You can compare any text based file, such as code source files.

### *Types of Differences*

File comparison reveals the following types of differences between files:

- General changes: Changes that are not clearly insertions or deletions.
- Additions: Additions to one revision of a file that are not present in another revision of the file.
- Deletions: Content that has been deleted from one revision of a file but not from another.
- Moves: Content that has been moved in one revision of a file but not in another.

## Comparing Files

To compare files:

- You must first select the file that you will compare, and display the file comparison view. You can compare a local file to the latest revision in Version Manager, or you can compare two revisions of the same file. See ["Displaying File Comparisons" on page 200](#).
- You can then review the differences. See ["File Comparison Usage Overview" on page 200](#) and ["Navigating Differences" on page 201](#).

## Displaying File Comparisons



### To display a file comparison:

Do one of the following:

To compare	Do this
The local copy of a file to the latest revision in Version Manager	<ol style="list-style-type: none"> <li>Select the file in the Solution Explorer.</li> <li>Right-click, and select <b>Compare Revisions</b> from the resulting menu.</li> </ol>
Two different revisions of the same file in Version Manager	<ol style="list-style-type: none"> <li>Select the file from the Solutions Explorer.</li> <li>Right-click, and select <b>View History</b> from the resulting menu.</li> <li>CTRL-click to select two revisions, then right-click and select <b>Compare Revisions</b> from the resulting menu.</li> </ol>

The Compare Revisions view appears. By default, both of the revisions appear, side-by-side. Path and revision information appears above each file pane.

You can click either of the following buttons to show / hide just one of the files:


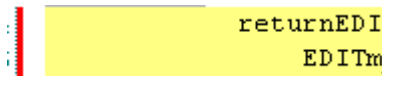
-  To only display file one.
-  To only display file two.

## File Comparison Usage Overview






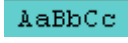
Each difference and conflict in a file comparison is represented by unique graphical elements. Use these graphical elements to help you navigate difference reports. Each difference is indicated by the following:

- **Marker:** Color markers in the bar to the right of each file pane summarize all of the differences. Different colors mark each type of difference. A separate marker appears for each difference.
- **Text markup:** Differences are highlighted within the text itself. Different colors and styles are applied to each type of difference. You can customize the text markup styles for each type of difference. See ["Setting Encoding and Display Options" on page 192](#) for details.

The following table describes each of the markers, icons, and default text markup styles you will see for each type of difference.



Type	Marker	Default Text Markup
<b>Modification</b> Any difference that is neither an addition nor a conflict is marked as a change.		Yellow: <pre> :      returnEDI :      EDITm           </pre>
<b>Deletion</b> Any text that has been deleted from one of the revisions of the file that you are comparing.		Strike-out: <pre> 127  -m-CountOfTypeOccura           </pre>



Type	Marker	Default Text Markup
<b>Addition</b> Text that has been added to one of the revisions of the file that you are comparing.		Green: 
<b>Conflict</b> Text that has been changed in one of the two revisions, and that somehow conflicts with the same text in the other revision. You must resolve any conflicts if you want to merge two revisions of a file during check-in. See <a href="#">"Reviewing and Resolving Conflicts" on page 201</a> .		Gray: 
<b>Move</b> Any text that has been moved in one of the revisions of the file you are comparing.		Blue: 

## Navigating Differences

### To review changes in a difference report:

- 1 Click the difference markers to navigate to a specific difference. When you click the marker in one file pane, all file panes automatically jump to the same marker, allowing you to compare differences side-by-side.
- 2 To jump from the current change to the next change, click the **Next Difference** button .
- 3 To jump from the current change to the previous change, click the **Previous Difference** button .

## Reviewing and Resolving Conflicts

Review the following procedures to learn how to review conflicts between your local copies of files and the latest revisions in Version Manager, and how to resolve those conflicts before checking in new revisions of the files:







- ["Reviewing Differences and Conflicts" on page 201](#)
- ["Resolving Conflicts" on page 202](#)
- ["Completing Merges" on page 203](#)

You must resolve conflicts before checking in, if you intend to merge your local updates with the latest revision in Version Manager.

### Reviewing Differences and Conflicts

Note the following about reviewing differences and conflicts from the Merge tool:


- For an explanation of how to review and navigate differences and conflicts in a file comparison, see ["Comparing Files" on page 199](#).

- Each type of difference is marked by a specific icon to the left of the Latest, Local, and Merged file panes:
  - Additions: 
  - Deletions: 
  - Edits: 
- Conflicts are marked on the left by the **Copy edits to solution** button: . See ["Resolving Conflicts" on page 202](#) for more information.
- You can skip ahead to the next conflict, or back to the previous conflict using the **Next Conflict**  and **Previous Conflict**  buttons.

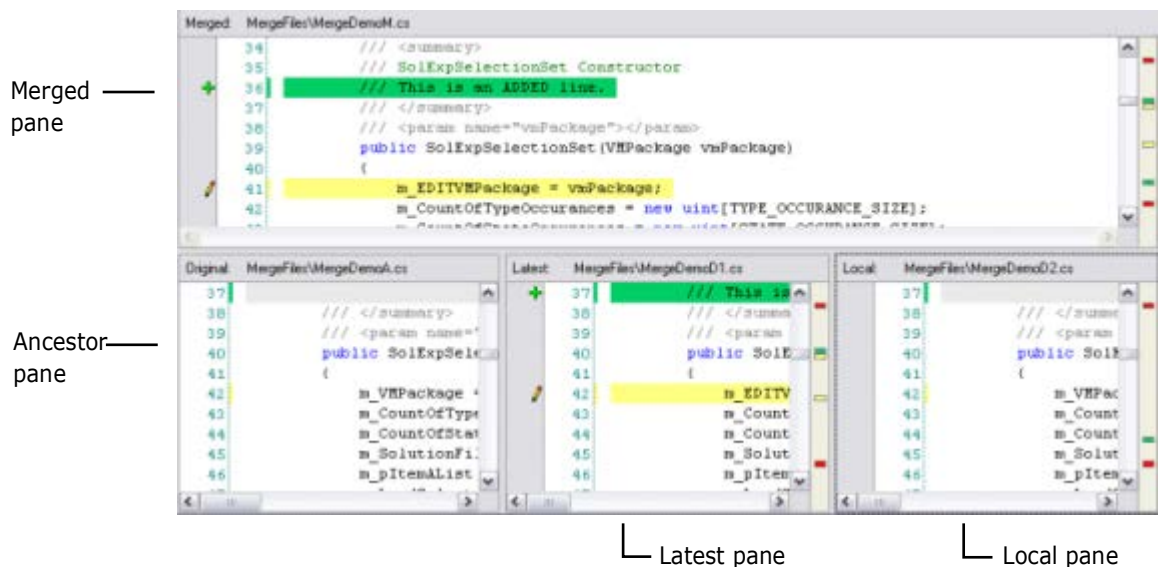
## Resolving Conflicts

You must resolve all conflicts in a file before you can get, check out, or check in the file. The file is ready to check in or get once there are no more conflicts visible in the Merged file pane.




### To resolve conflicts:

- Select a conflicting file in the Compare Workspaces view and click the **Resolve conflicts**  button. The file opens in the Merge tool. (See ["Comparing Workspaces" on page 194](#) for information on the Compare Workspaces view.)

By default, the Merge tool displays four panes: **Ancestor**, **Latest**, **Local**, and **Merged**:





- The **Ancestor** pane displays the original file, without markup.
- The **Latest** pane displays the latest revision in Version Manager, and highlights differences from the original file.
- The **Local** pane displays your local, edited copy of the file, and highlights differences from the original file.
- The **Merged** pane displays the new revision that will be created as a result of the merge operation.

- 2 Locate the conflict you want to resolve. You can jump to it by clicking the **Next Conflict**  or **Previous Conflict**  button, or by clicking a red marker in the right margin . Within the Merged file pane, the area where the conflict occurs is blank, and indicated by a gray placeholder:

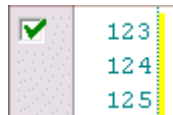


As you navigate the conflicts in one file pane, all of the file panes jump to the corresponding point. For example, if you navigate to a conflict placeholder in the Merged file pane, the Latest and Local file panes display the corresponding lines that are in conflict with each other.

- 3 Review the conflicting lines in the Latest and Local panes, and choose how to resolve it. You can resolve it in any of the following ways:

Solution	Procedure
Edit the text directly	Click on the conflict placeholder in the Merged file pane, and enter the text that will resolve the conflict. You can also paste or drag text from another window or application into the Merged file pane.
Use the lines from either the Latest or the Local file pane	To the left of the conflicting lines in either the Latest or the Local file pane, click the <b>Copy edits to solution</b> button:  . The lines are copied into the corresponding placeholder in the Merged file pane. You can also click in the lines you want to copy to the Merged file pane, and click the <b>Copy edits to solution</b> button  on the toolbar.

- 4 Once the conflict is resolved, a checkbox appears to the left of the updated text in the Merged file pane:



## Completing Merges

Once you have resolved all conflicts, click the **Done** button and do one of the following:

- If you are performing a bulk synchronization of your local workspace with the corresponding Version Manager project, synchronize workspaces from the Compare Workspaces view. See ["Synchronizing Workspaces" on page 198](#).
- If you are checking files in, check them in from the Solution Explorer, or perform a bulk commit of all local changes. See ["Checking In Files" on page 182](#) or ["Committing Local Changes to Version Manager" on page 197](#).
- If you are getting files, get them from the Solution Explorer, or perform a bulk get of all changes from Version Manager. See ["Checking In Files" on page 182](#) or ["Getting All Updates from Version Manager" on page 195](#).
- If you are checking files out, check them out from the Solutions Explorer. See ["Checking Out Files" on page 178](#).

## Associating and Working on SBM Issues

If your organization uses SBM to track development issues, such as defects and tasks, you can access your issues from within the Version Manager integration to Visual Studio. You can submit and modify SBM issues from within Visual Studio, and then associate issues with specific files. When you associate issues with files, the versioned file history is added to the issue.

**NOTE** The SBM user privilege **Run System Reports** is required in order to use the integration to SBM, else the following error message will appear:

Error reading associations (No permission)

See the SBM Administrator Guide.

See the following for detailed information on the SBM integration to Visual Studio:

- ["Issue Management Workflow" on page 204](#)
- ["Setting Up Your IDE Folder" on page 205](#)
- ["Defining Association Options" on page 206](#)
- ["Logging into SBM" on page 207](#)
- ["Displaying Reports and Issues" on page 208](#)
- ["Submitting and Modifying Issues" on page 208](#)
- ["Associating Issues" on page 209](#)

### Issue Management Workflow

The following table describes the issue management workflow in Visual Studio. You must follow this workflow to successfully display issues, and associate them with files.

Step	Description
1	<b>Set up IDE Folder</b> Before you can access issues from within Visual Studio, you must set up your IDE folder in SBM. The IDE folder is a special system folder that enables you to display specific issues and listing reports from the rich integrations to Visual Studio and Eclipse. See <a href="#">"Setting Up Your IDE Folder" on page 205</a> .
2	<b>Define Integration Settings</b> In the Version Manager desktop client, you can optionally update settings that affect issue association in the rich integration to Visual Studio, including: <ul style="list-style-type: none"> <li>■ Whether to apply a version label to all revisions associated with a SBM issue that includes the issue number</li> <li>■ Whether to require issue associations on check-in</li> <li>■ Whether to automatically add notes about associated issues to the check-in comments for new revisions</li> </ul> See <a href="#">"Defining Association Options" on page 206</a> .

Step	Description
3	<p><b>Connect to the SBM Server</b></p> <p>In Visual Studio, when adding or getting a project to source control, you have the option to specify the SBM server that contains the solution you will use to manage your issues. If you did not specify the SBM server at that time, or if you need to change the SBM server connection, complete this procedure. When you connect to a SBM server, you also login as a specific user. See <a href="#">"Logging into SBM" on page 207</a>.</p>
4	<p><b>Review, modify, and submit issues</b></p> <p>From the Issues view, you can display all issues that are available via listing reports in your IDE folder, or that have been added directly to your IDE folder. For example, this may include specific reports that list only issues that are assigned to you.</p> <p>You can then modify these issues, and even submit new issues. See <a href="#">"Displaying Reports and Issues" on page 208</a> and <a href="#">"Submitting and Modifying Issues" on page 208</a>.</p>
5	<p><b>Associate issues with revisions of files</b></p> <p>The SBM integration to Version Manager also allows you to <i>associate</i> issues with specific revisions of files. When you associate an issue with a file, a <b>Version Control History</b> section is added to the issues, that tracks information about the revisions of the files. A version label with the issue ID can also be assigned to the associated revision. See <a href="#">"Associating Issues" on page 209</a>. for details.</p> <p><b>To associate issues:</b></p> <ol style="list-style-type: none"> <li>1 <i>Activate the issue.</i> This places the issue in a queue of issues that you can optionally choose to associate with files when you check them in.</li> <li>2 <i>Work on the files that are affected by the issue.</i> For example, you may need to edit specific source code files to resolve a problem described in a specific issue.</li> <li>3 <i>Check in the files.</i> When you check in, you have the option to <i>associate</i> the files with any (or all) of the currently activated issues. At this time, you can choose specifically which issues the files will be associated with. If the files you are checking in effectively end your portion of the work to address the issues, you can also choose to deactivate the issues.</li> </ol>

## Setting Up Your IDE Folder

If your user folder is not visible in the Issues tab of Visual Studio (or the IDE folder is not visible in the **Favorites** list in SBM), then you must enable it.

### To enable your IDE folder:

- 1 Launch SBM.
- 2 Click the **User Profile** link (in older versions it is your user name) in the upper-right corner of the Web client. The Edit User Profile page appears.
- 3 Select the **Display** tab.
- 4 Select the **Auto Folder Items** option.
- 5 Click the **Save Profile** button.

The IDE folder (and other auto folders) will now appear in the **Favorites** list in the Web client, and the contents of the folder will appear in the Issues tab of Visual Studio. You can now specify which issues you want to access from Visual Studio by doing any of the following from the Web client:

- Add specific issues directly to your IDE folder.
- Add listing reports to your IDE folder.

You can access only specific issues and listing reports from within Visual Studio. You cannot access other types of reports, or other types of items, such as URLs.

See the *SBM User's Guide* for details on setting up personal or favorites folders.

## Defining Association Options

From the Version Manager desktop client, the administrator can define settings for rich IDE integrations to SBM, including:

- Whether to apply a version label with the issue number to all associated revisions
- Whether to require issue associations on check-in
- Whether to automatically add notes about associated issues to the check-in comments for new revisions

### To define association options:

- 1 From the Version Manager desktop client, select the project database to which you will apply the settings.
- 2 Select Admin | SourceBridge settings. The SourceBridge Settings dialog box appears.

3 Set the following options:


Field	Description
<b>Show Issue association dialog on checkin / Association required</b>	Select to require that issues be associated with files at check-in. If you select this option, users will be unable to complete checkins if no issues are currently active. The <b>Show Issue association dialog on checkin</b> option has no effect within the rich integration to Visual Studio, but you must select it in order to enable you to then select the <b>Association required</b> option.
<b>Tag workfile comment with association</b>	<p>Select to add information about the associated issue(s) to the check-in comments for files as they are checked in. Select <b>Before existing comment</b> or <b>After existing comment</b> to determine the placement of this information within the comment.</p> <p>In the <b>Tag</b> field, enter the text that you want to add to the check-in comments. This can include any of a number of keywords that will automatically enter information about the associated issues. These include:</p> <ul style="list-style-type: none"> <li>■ \$id -- Expands to the issue ID number</li> <li>■ \$ownid -- Expands to the user ID of the issue owner</li> <li>■ \$owner -- Expands to the name of the issue owner</li> <li>■ \$project -- Expands to the name of the current project</li> <li>■ \$title -- Expands to the title of the issue</li> </ul>
<b>Use Version Labels on checkin</b>	Select to apply a version label consisting of the issue number when checking in a file.

4 Click **OK**.

## Logging into SBM

When adding or opening a project from Version Manager, you have the option to specify the SBM server that contains the solution you will use to manage your issues. If you did not specify the SBM server at that time, or if you need to change the SBM server connection, complete this procedure. When you connect to a SBM server, you also login as a specific user. All issues that are visible from the IDE folder in SBM are then visible from Visual Studio.

### To connect to an SBM server:

- 1 Select View| Issues. The Issues view appears.
- 2 Click the **SBM Login** button . The Connect to SBM dialog box appears.
- 3 Enter the SBM server name in the **SBM Host** field. If the SBM server uses a non-default port number (any port except 80), append the port number to the server name. For example, if the port number is 89:


```
tt_server:89
```

- 4 Enter your SBM user name and password and click **Next**.
- 5 Click **Finish**.

## Displaying Reports and Issues

From the Issues view, you can display all issues that are available via listing reports in your IDE folder, or that have been added directly to your IDE folder. For example, this may include specific reports that only list issues that are assigned to you.


### To display reports and issues:

- 1 Select View | Issues. The Issues view appears.
- 2 To review your issues:
  - Select your user name node to list any issues that have been added to your IDE folder.
  - Expand your user name node to display all reports that are available to you. Any listing reports that have been added to your IDE folder appear here. You can then click any of the reports to display issues.
  - Select the **Activated Issues** node to display any currently activated issues (issues that you are currently working on).  
  
The **Related Issues** node displays all issues that are associated with a particular file. See ["Associating Issues" on page 209](#) for information on using this list.
- 3 To view the contents of an issue, select the issue and click the **View Issue** button .


## Submitting and Modifying Issues

Submit and modify SBM issues to track the status and details of the tasks that you are completing in Visual Studio. You can submit new issues for tasks, defects, or other work that needs to be completed, or modify issues to provide input into your work assignments. Depending on the workflow for your organization, you may modify issues in order to move them to another state, for example if you have completed your portion of the task and need to mark it as ready to test.

### To submit an issue:

- 1 Select File | Source Control | Issues. The Issues view appears.
- 2 Click the **Submit Issue** button . See the *SBM User Guide* for more information on submitting issues.

### To modify an issue:

- 1 Locate the issue you want to update. See ["Displaying Reports and Issues" on page 208](#).
- 2 Select the issue and click the **View Issue** button .
- 3 Update the issue as needed. See the *SBM User Guide* for more information on updating issues.



## Associating Issues

In addition to providing access to specific issues and reports, The SBM integration to Version Manager also allows you to *associate* issues with specific revisions of files. When you associate an issue with a file:

- A Version Control History section is added to the issues, that tracks:
  - The name of the associated file
  - The revision number
  - The check-in date
  - The user who checked in the file
  - The description of the change that the user entered when checking in

For example, if Joe associated an issue with a file called test.cs, something like the following might appear in the issue after check-in:




```

▢ Version Control History
  /Application/app 1/test.cs
    Revision 1.3 Checked In by Joe Manager 2/4/2005 3:54:49 AM
    Revision 1.2 Checked Out 2/4/2005 3:54:49 AM
    minor change
  
```

- Optionally, a version label is assigned to the revision of the file that is associated with the issue. The version label includes the issue number.
- Optionally, information about the associated issue(s) is added to the check-in comment for the new revision.

See ["Defining Association Options" on page 206](#) for information on setting the association options.

### To associate issues:

- 1 Locate the issues that you will work on and eventually associate to file revisions. See ["Displaying Reports and Issues" on page 208](#).
- 2 Select the issue and click the **Activate Issue** button . The issue is added to your **Activated Issues** list.  
To remove an issue from the Activated Issues list, select the issue from the list and click the **Deactivate Issue** button .
- 3 At any point, you can review the details of an issue by selecting it and clicking the **View Issue** button .
- 4 Complete the work required to resolve the issue, or your portion of it.
- 5 Check in the file or files that resolve the issue. On the Check In dialog box, under SBM Associations, select the issue that you want to associate with the file or files. Only issues that are currently activated can be associated during check-in. See ["Checking In Files" on page 182](#).

**To display all issues associated with a particular file:**

In the Solution Explorer, right-click the file and select **Related Issues** from the resulting menu. The Issues view appears with the **Related Issues** node selected. All issues related to the currently selected file are listed in the right pane.

# Appendix A: Naming Conventions and Restrictions

---

General Naming Conventions and Restrictions	212
Specific Naming Conventions and Restrictions	213

---

## 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



**IMPORTANT!** On Windows systems, files and directories (and thus Version Manager entities and paths) cannot end with a period (.).

### 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 by 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.

Item Type	Restrictions	
	Characters	Length
Archives	As listed in <a href="#">"Prohibited Characters for Files and Directories" on page 212</a> , plus cannot use:  Ampersand: & Brackets: [ ] Parenthesis: ( ) Plus sign: + Semicolon: ;	254 (full path including the file name)  <b>NOTE</b> 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 <a href="#">"Prohibited Characters for Files and Directories" on page 212</a> .	254 (full path including the file name)
pvcs_bindir	As listed in <a href="#">"Prohibited Characters for Files and Directories" on page 212</a> .	254 (full path including the file name)  <b>NOTE</b> 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 <a href="#">"Prohibited Characters for Files and Directories" on page 212</a> , plus cannot be: <ul style="list-style-type: none"><li>■ The two-character name of: ..</li><li>■ The one-character name of: .</li><li>■ The one-character name of: @</li></ul>	

Item Type	Restrictions	
	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: /  <b>NOTE</b> The backslash (\) serves as an escape character. To create or delete a label that includes a backslash, the backslash must be preceded by another backslash (\\Label would result in \Label; \\\\Label would result in \\Label).	254

# Index

---

## A

- access list, definition of 12
- adding files to source control 24
- administrative workflow
  - non-web projects 18
- archives
  - creating 12, 24
  - definition of 12
  - directory structure 24
  - history 14
  - properties, reviewing 46
  - sharing 25
  - sharing, about 14
- assigning labels to revisions 34

## B

- branches
  - creating 34
  - definition of 14

## C

- changes to files, storing 33
- check in
  - defaults 33
  - overriding defaults 33
  - procedure 33
- check out
  - by date 31
  - by revision 31
  - default options 30
  - overriding defaults 31
  - undoing 32
- ColdFusion Studio
  - adding
    - files to source control 61
    - projects to source control 60
  - checking in files 63
  - checking out files 63
  - getting files 63
  - removing files from source control 62
  - setting up multiple-users 59
  - undoing checkout 63
- comparing files 15, 52
- configuring
  - project databases 20
  - source control environment 22

- Version Manager defaults 22
- creating projects in
  - SCC IDEs 23

## D

- default
  - check-in options 33
  - checkout options 30
  - get options 28
  - revision, definition of 13
- deleting version labels 37
- design part, top level. See project databases
- difference reports
  - about 15
  - generating 52
- displaying properties 46

## E

- Eclipse 3
  - adding projects to source control 80
  - checking in files 87
  - checking out files 86
  - compare with local history 94
  - connecting additional workstations to source control 82
  - disconnecting projects from source control 84
  - excluding files from source control 80
  - getting files 85
  - icon glyphs enabling 85
  - locking files 86
  - offline mode 90
  - removing files from source control 84
  - replace with local history 94
  - source control status 84
  - undoing checkout 87
- Eclipse rich integration
  - adding projects to source control 104
  - associating TeamTrack issues 130
  - checking in files 115
  - checking out files 113
  - compare with local history 123
  - connecting additional workstations to source control 106
  - disconnecting projects from source control 108
  - excluding files from source control 103
  - getting files 112
  - reconnecting projects to source control 108
  - replace with local history 124

---

- setting default options 131
- source control status 109
- submitting TeamTrack issues 129
- synchronizing with source control 118
- undoing checkout 114
- working offline 99
- working with TeamTrack issues 125

editing files 30

environment, configuring 22

## F

files

- adding to source control
  - non-web projects 24
- checking in 33
- checking out 30
- getting 28

floating labels 24

## G

generating

- difference reports 52
- history reports 51

getting files

- by date 30
- by promotion group 29
- by revision 29
- default options 28
- overriding defaults 29

global workset. See root workspaces 21

## H

history reports

- about 14
- generating 51
- types of 51

## I

information, source control 46

initial revision, definition of 13

invoking Pulse 48

## L

labels

- assigning 34, 186
- deleting 37, 188
- floating 35
- reassigning 36
- renaming 36, 187

launching

- Pulse 48
- Version Manager 21

locking files, definition of 13

logging in to a project 28

## M

migrating SCC projects to rich Visual Studio 2005

- integration 157

monitoring source control activity

- about 47
- in multiple environments 48

## N

nested project structure 12

numbering revisions 13

## O

options, setting up environment 22

organizing projects 12

## P

parallel development. See branches 14

PowerBuilder

- checking in objects 74
- checking out objects 73
- getting objects 73
- objects
  - removing from source control 72
- taking objects 73
- undoing checkout 74
- version 8
  - disconnecting from source control 72

preferences 22

product item. See versioned files

project activity, monitoring 47

project databases

- creating 20
- definition of 12

projects

- creating
  - for SCC IDEs 23
- definition of 12
- setting up 18

promotion

- groups
  - assigning 24
  - checking out by 31
  - definition of 13
  - reviewing 46



---

- models, definition of 13
- promotion groups 38
  - about 38
  - assigning to revisions 39
  - changing 41
  - checking out revisions 38
  - permissions 38
  - promoting to next group 40
  - removing 42
  - setting up a promotion model 38
- promotion models
  - about 38
  - setting up 38
- properties, reviewing 14, 46
- public workset. See public workspaces 21
- public workspaces 21
- Pulse
  - about 47
  - configuring 47
  - starting 48
  - suspending 50
- PVCS Merge 52
- PVCSCLIServ service 20

## R

- Rational Application Developer
  - adding projects to source control 104
  - associating TeamTrack issues 130
  - checking in files 115
  - checking out files 113
  - compare with local history 123
  - connecting additional workstations to source control 106
  - disconnecting projects from source control 108
  - excluding files from source control 103
  - getting files 112
  - reconnecting projects to source control 108
  - replace with local history 124
  - setting default options 131
  - source control status 109
  - submitting TeamTrack issues 129
  - synchronizing with source control 118
  - undoing checkout 114
  - working offline 99
  - working with TeamTrack issues 125
- reassigning version labels 36
- refresh rate, Pulse 47
- relating TeamTrack issues 209
- removing files from source control
  - SCC IDEs 26
- renaming version labels 36, 187
- reports
  - difference
    - about 15
    - generating 52
  - history
    - about 14

- generating 51
- results messages, displaying 47
- reviewing TeamTrack issues 208
- revisions
  - assigning to a promotion group 39
  - assigning version labels to 13
  - checking out
    - assigned to promotion group 38
  - comparing 52
  - definition of 13
  - numbering 13
  - promoting 40
  - properties, reviewing 46
- rich integration
  - TeamTrack and Visual Studio 2005 204
  - Visual Studio .NET 2003 152
- root workspaces 21

## S

- saving changes to files 33
- SCC projects, migrating to Visual Studio 2005
  - rich integration 157
- SCC provider
  - selecting 19
  - testing 20
- selecting an SCC provider 19
- settings, source control 22
- sharing files
  - definition of 14
  - how to 25
- source control
  - concepts 12
  - environment, configuring 22
  - information, viewing 14, 46
  - removing SCC projects from 26
- subprojects, definition of 12

## T

- team environment
  - personal workspaces 21
  - sharing files 25
  - updating your workfiles 30
- TeamTrack
  - association options 206
  - IDE folder 205
  - issues
    - associating 209
    - displaying 208
    - submitting and modifying 208
  - logging in to 207
  - workflow 204
- testing SCC initiation 20
- timestamps, setting 30, 32
- tip revision, definition of 13

---

trunk, definition of 14

## U

undoing checkout 32

unlocking files 32, 33

user

    ID, entering 28

using

    reports 46

    version labels 34

## V

version labels

    assigning 24, 34, 186

    assigning to a previous revision 187

    definition of 13

    deleting 37, 188

    floating 24, 35

    reassigning 36

    renaming 36, 187

    viewing 46, 175

Version Manager

    about project databases 12

    launching 21

    workspaces 21

    workspaces, using in Visual Studio 154

versioned files

    comparing 52

    editing 30

viewing

    file status in Visual Studio 181

    properties 14, 46

    source control activity 47

Visual Studio Rich Integration

    adding solutions and projects to Version Manager 168

    branching, about 161

    default version 150

    editing files 174

    forcing a branch at check in 161, 184

    migrating SCC projects to 157

    offline, working 188

    opening solutions and projects from Version Manager 170, 174

    overview 152

    resolving conflicts 199

    Source Control toolbar 152

    synchronizing workspaces 193

    toolbar 152

    using TeamTrack 204

    using workspaces 154

    working offline 188

Visual Studio SCC integration

    adding projects to source control 143

    checking in files 148

    checking out files 146

    configuring SCC behavior 142

    getting files 146

    removing files from source control 143

    supported features 140

    undoing checkout 147

## W

workfiles

    adding to projects 24

    comparing 52

    definition of 13

workflow

    non-web projects 18

    TeamTrack rich integration to Visual Studio 204

workset. See workspaces 21

workspaces

    definition of 14

    Version Manager 21

    Visual Studio

        comparing 194

        overview 154

        synchronizing 198