Perforce 2003.2
Using IDE Plug-ins

February 2004

Table of Contents

Preface About This Manual

This guide describes the installation, configuration, and operation of Perforce plug-ins for a variety of integrated development environments (IDEs).

This manual assumes some familiarity with Perforce. For more information on Perforce from a user's perspective, consult the *Perforce User's Guide* or *Perforce Command Reference*.

All our documentation is available from our web site at `http://www.perforce.com`.

Please Give Us Feedback

We are interested in receiving opinions on it from our users. In particular, we'd like to hear from users who have never used Perforce before. Does this guide teach the topic well? Please let us know what you think; we can be reached at `manual@perforce.com`.

Chapter 1 Basic Concepts

About IDEs and plug-ins

Perforce plug-ins enable you to perform basic Perforce software configuration management (SCM) tasks from within integrated development environments (IDEs). Perforce offers two IDE plug-ins:

- Perforce SCC Plug-in: based on the Microsoft Source Code Control (SCC) API and has been tested with the following IDEs:
  - Microsoft Visual C++, Visual Basic, and Visual Studio .NET
  - TogetherSoft Together
- Perforce CodeWarrior Plug-in: supports Metrowerks CodeWarrior for Macintosh and Windows, and is based on their proprietary API.

Perforce also offers plug-ins for other development tools. See our Web site for details.

The Perforce SCC Plug-in is installed when you install Perforce on Windows. To obtain the Perforce CodeWarrior Plug-in, go to the Perforce downloads web page.

The SCC and Perforce CodeWarrior Plug-ins offer similar functionality but have different user interfaces. For details about supported versions of each IDE, refer to the release notes for the plug-ins.

Note that the terminology for SCM tasks varies depending on which development environment you are using. The following table compares some common terms.

IDE with SCM plug-in	Other commonly used terms	Corresponding Perforce command
Add to source control/Perforce	Add	`p4 add`
Check out	Edit	`p4 edit`
Check in	Submit	`p4 submit`
Show differences	Diff	`p4 diff`
Get latest version	Sync, Refresh	`p4 sync`
Show history	Filelog	`p4 filelog`
Undo checkout	Revert	`p4 revert`
Remove from source control	Delete	`p4 delete`

About Perforce

Perforce is an SCM system that is based on a client/server architecture. The main repository (the *depot*) resides on a central server while the files you work on reside in a workspace on your local machine. You can place some or all of the files in your workspace under source control. When you perform SCM tasks, the files remain in your workspace, and Perforce reads or writes them as required. For example, when you submit a change, Perforce reads the edited files in your workspace and updates the information in the database accordingly. When you issue a **Get latest version** or **refresh** command, Perforce transfers files from the depot to your workspace.

The most current revision of the file in the depot is called the *head revision*. Perforce allows you to check out the head revision or any previous revision of a file. To enforce the IDEs' check-in and check-out procedure, Perforce controls the read-write permissions of files. When files are checked out for edit, their permissions are set to read-write. When files are not checked out, Perforce sets them to read-only.

Perforce submits changed files in groups called *changelists*. Perforce keeps track of a project's revision history as a sequence of changelists. This approach allows you to reconstruct a project in your workspace as it appeared at any point in its history. Perforce makes changes *atomically*, meaning that when you submit changes to a group of files, either all of the changed files are accepted simultaneously or none of them are. If conflicts result from multiple users working on the same files, these conflicts must be resolved before Perforce will accept the changes.

Perforce offers a command-line client (`P4`), a native Windows GUI (P4Win), a cross-platform GUI (P4V) and a Web-based interface (P4Web). For detailed information about Perforce, refer to the user documentation available on the web at:

`http://www.perforce.com/perforce/technical.html`

Most IDEs are project-based: they manage a group of files according to the project to which the files belong. However, Perforce manages files with no provisions for specific IDEs' project structure. You must determine which files you need to place into your Perforce depot, depending on the conventions of your IDE and your group development practices.

Tracking changes

When files are added to source control, deleted from source control, or checked out for edit, Perforce adds them to a changelist. The changelist contains the file names, revision numbers, and operations to be performed. Any edits you make to checked out files are kept in your local client workspace until you send the changelist to the depot with a check in or submit command.

The Perforce server tracks changelists by numbering them sequentially. Changelist numbers are displayed in a results window after a change has been submitted.

Handling conflicts

The ability to detect and resolve conflicts is important in team development, when multiple developers are working on the same files. For example, suppose two programmers copy the same file from the depot into their workspaces and each programmer edits the file differently. When the first programmer submits his version of the file to the depot, the file becomes the head revision. When the second programmer tries to submit her changes to the depot, Perforce determines that her changes are based on a previous revision and does not allow the file to be checked in. If the second file were accepted without question, the first programmer's changes would be overwritten.

When Perforce detects a conflict, it requires you to choose the changes to be checked in. When resolving file conflicts, you can use a merge utility to display the differences between two text files, to help you determine how to resolve the conflict. For more details, see the *Perforce Command Reference*.

File types

Perforce automatically detects whether files placed under source control are text or binary files and stores them accordingly on the server machine. By default, text files are stored in reverse delta format, and binary files are stored in their entirety. The following table lists recommended Perforce file types and attributes for common file types. For details about Perforce file types, refer to the *Perforce User's Guide* For a list of common file types, see `http://www.icdatamaster.com`.

File Type	Perforce file type	Description
`.asp`	`text`	Active server page file
`.avi`	`binary+F`	Video for Windows file
`.bmp`	`binary`	Windows bitmap file
`.btr`	`binary`	Btrieve database file
`.cnf`	`text`	Conference link file
`.css`	`text`	Cascading style sheet file
`.doc`	`binary`	Microsoft Word document
`.dot`	`binary`	Microsoft Word template
`.exp`	`binary+w`	Export file (Microsoft Visual C++)
`.gif`	`binary+F`	GIF graphic file
`.htm`	`text`	HTML file

File Type	Perforce file type	Description
`.html`	`text`	HTML file
`.ico`	`binary`	Icon file
`.inc`	`text`	Active Server include file
`.ini`	`text+w`	Initial application settings file
`.jpg`	`binary`	JPEG graphic file
`.js`	`text`	JavaScript language source code file
`.lib`	`binary+w`	Library file (several programming languages)
`.log`	`text+w`	Log file
`.mpg`	`binary+F`	MPEG video file
`.pdf`	`binary`	Adobe PDF file
`.pdm`	`text+w`	Sybase Power Designer file
`.ppt`	`binary`	Microsoft Powerpoint file
`.xls`	`binary+w`	Microsoft Excel file
`.zip`	`binary+F`	ZIP compressed archive file

Note The Perforce apple file type stores Macintosh files in the depot as a single file containing both the data and resource forks. Pre-2000.1 versions of Perforce stored Macintosh file forks as separate files in the depot. If your depots contain Macintosh files stored in this manner, upgrade the existing files to use the apple format. If you do not convert existing files after you upgrade, Perforce continues to store them using its two-file approach. For details, see “Working with CodeWarrior files” on page 52

Configuring IDEs with plug-ins

Regardless of the IDE you use, you must define a client view, add files to the Perforce depot, and configure your source control settings and preferences. The following sections provide details about these tasks.

Defining the client workspace and view

To set up the Perforce SCM environment, you define a *client workspace* (a directory on the client machine that contains the project files) and a *client view* (a mapping of the depot to your client computer). The files in your IDE projects must reside under the client workspace root (the highest-level directory of your client workspace). Your files can be put under source control only if they are located in the client workspace. You can specify a directory that already contains files, or an empty directory for a project you intend to create. You can also populate your client workspace with files from the depot which are currently under version control.

To create a client workspace and view, you must use the `p4` command line client program. Note that P4SCC does not observe settings in Perforce config files. For details about client workspaces and views, refer to the *Perforce User's Guide*.

Add a project to source control

To add a project to source control, you specify the Perforce settings required to connect to the server where you want to store the project. You must specify the following settings:

- Server host and port

This setting specifies the name of the host on which the server is running and the port on which the server is listening, using the following format:

IDE and Perforce host location	Required Settings	Examples

Same machine	Only port number is required	`1666`
		`1818`

Different machines	Host name (or IP address) and port are required	`myhost:1818`
		`mydomain.com:2550`
		`10.0.0.5:1666`

- User name: your Perforce user name.
- Password: (optional) your Perforce user password.
- Client workspace: the name of the client specification describing the workspace where you work on local copies of project files.

Adding files to the depot

After you specify the settings for the server where you want to store project files, you can add the files. The exact commands required to add files to the depot depend on the IDE in which you are working. Typically you perform two steps:

1. Add the files to a Perforce changelist by issuing a command such as **Add to Source Control**.
2. Submit the changelist by issuing a command such as **Check In** or **Submit**.

Files are added when the **Check In** or **Submit** is completed.

Basic SCM tasks

The plug-ins enable you to perform the following basic SCM tasks from within your IDE:

Task	Description	Perforce activity
Add files to the depot	After you add files to your project, you add them to the depot.	Files are opened for add in a pending changelist.
Retrieve files	Get a copy of a file from the depot.	Files are synced to your computer.
Check files out	Get the latest version from the depot for editing.	Files are opened for edit in a pending changelist.
Check files in	Put your edited files in the depot as the most recent revisions.	A pending changelist is submitted.
Diff files	Compare a file with a previous revision to see what's changed.	The Perforce diff utility is launched to display differences between two versions of a file.
Revert files	Discard changes you've made to your local copy of a depot file.	The head revision is synced to your client workspace and removed from its pending changelist.
Delete files	Remove a file from a project and from the depot.	Files are opened for delete in a pending changelist.

When to go outside the IDE

Perforce plug-ins support a basic set of SCM tasks. To perform administrative tasks, you need to use the `p4` command line client. For details about performing these tasks, see the *Perforce User's Guide*.

Tasks that require you to work outside the IDE include:

- *Modifying a client view or user password*: you can create a client specification or user through the **Connection** dialog when adding a project to source control, but to modify it subsequently, you must use another Perforce client program (the `p4` command line, P4Win, or P4V).
- *Creating branches and labels*: these and other administrative tasks must be done using the `p4` command. Refer to the *Perforce User's Guide* for details.

If you do perform Perforce SCM tasks outside the IDE, be sure to refresh the IDE display afterwards, to ensure that file status is updated and displayed correctly.

Chapter 2

Microsoft Visual C++

This chapter describes how to perform Perforce source code control tasks in the Visual C++ 6.0 environment. It assumes the Perforce SCC Plug-in is installed and that your client machine can communicate with the Perforce server.

When you configure Visual C++ with the Perforce SCC Plug-in, the settings are specific to the project you are working on, so no overall Perforce configuration is required. The Perforce SCC Plug-in retains project settings from session to session and it is not necessary to re-enter them.

Visual C++ has a **Source Code Control** toolbar: to enable it, right-click the toolbar and check **Source Code Control** on the pop-up menu.

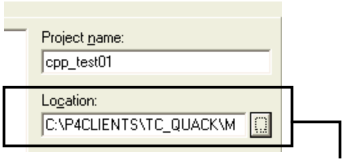
Basic SCM tasks

This section tells you how to perform the following tasks:

- “Adding a project to the depot” on page 17
- “Checking files out” on page 19
- “Retrieving files from the depot” on page 18
- “Checking files in” on page 20
- “Resolving file conflicts” on page 20
- “Diffing files” on page 21
- “Reverting files” on page 21

Adding a project to the depot

When you create a project you intend to manage with Perforce, specify a location that resides within the Perforce client root folder of the client you intend to use as illustrated in the following figure.



Project name: cpp_test01

Location: C:\P4CLIENTS\TC_QUACK\M

Must be in
client root folder

When you save your project, you are prompted “Do you want to put the newly created project under source control?”

1. Click **Yes**. The **Open Connection** dialog is displayed. On the **Open Connection** dialog, specify Perforce settings as follows:
    - *Server*: the name of the host computer running the Perforce server in which you want to store the project.
    - *Port*: the port number on which the specified Perforce Server accepts commands.
    - *User*: the Perforce user name you want associated with the SCM operations you perform on the project. To choose from a list of users defined for the specified server, click **Browse**.
    - *Password*: the user’s password, if any.
    - *Client*: the Perforce client specification you want associated with the SCM operations you perform on the project. To choose from a list of client specifications defined for the specified server, click **Browse**.
2. Click **OK** to save settings and dismiss the dialog. The **Add to Source Control** dialog is displayed.
3. Right-click the project and choose **Check In...** The **Check In Files to Perforce** dialog is displayed.
4. Check the files you want to add and click **OK**. The **Pending Changelist** form is displayed.
5. Enter a comment and click **Submit**. The files are checked into the depot.

When you open a project that you have placed under Perforce control, Visual C++ attempts to connect to the Perforce server that you configured. If Visual C++ can not connect to Perforce, it asks you whether it should try to connect in future sessions. Choose **Yes**. If you choose **No**, Visual C++ assumes the project is no longer under version control, requiring you to add it again. If you try to add the project again, Perforce returns an error indicating that the project is already under source control, and you must retrieve the project from the depot.

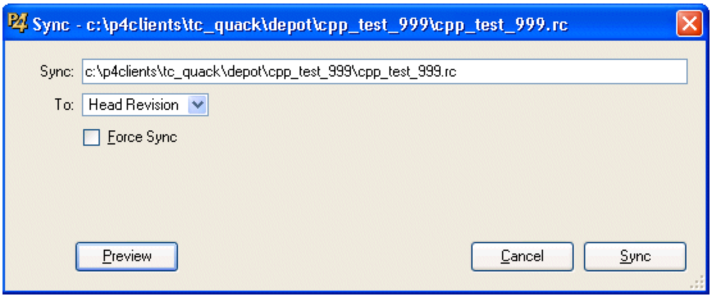
Retrieving files from the depot

To get the most recent file revisions from the depot:

1. In the file list pane, select the files you want to retrieve.
2. Choose **Project >Source Control >Get Latest Version...** The **Get Latest Version** dialog is displayed.
3. Check the files you want to retrieve and click **OK**.

To retrieve revisions prior to the head (latest) revision, you can sync from the **Revision History** dialog or perform the following steps:

1. In the file list pane, select the files you want to retrieve.
2. Choose **Project >Source Control > Get Latest Version...** The **Get Latest Version** dialog is displayed.
3. Check the files you want to retrieve and click **Advanced...** Dismiss the "Advanced sync dialog..." prompt by clicking **OK**, then click **OK** on the **Get Latest Version** dialog. The **Sync** dialog is displayed as shown in the following figure.


Sync - c:\p4clients\tc_quack\depot\cpp_test_999\cpp_test_999.rc

Sync: c:\p4clients\tc_quack\depot\cpp_test_999\cpp_test_999.rc

To: Head Revision

☐ Force Sync

4. Specify the revision you want to retrieve, using a Perforce changelist number, label, version number, or date. For details about specifying revisions, refer to the *Perforce User's Guide*.
   - To preview the results of the operation with actually retrieving files, click **Preview**.
   - To retrieve a file from the depot if you've deleted your local copy manually, check **Force Sync**. By default, Perforce assumes you still have your copy and doesn't retrieve it again.
5. Click **Sync**.

Note If you retrieve a file that you already have checked out, for example, to obtain changes checked in by another user, Perforce will require you to resolve the file, to enable you to merge your changes with changes made by other users and to ensure that you don't inadvertently overwrite other changes.

Checking files out

Before you check a file out, be sure you have retrieved the latest revision of the file. (If you check out a file for which the depot contains later revisions, you will be required to resolve it when you check it in.)

To check out a file for edit, right-click the file and choose **Check out *filename***.

Checking files in

After editing the open files, you can save them to your local directory and check the files into the depot. To save the files on your local directory, choose **File>Save**. Note that saving does not update the depot.

To make your changes available to others working on the project, you must check your edited files into the depot. If the changes are accepted, your files become the most recent (or *head*) revisions in the depot. If any of the changes conflict with changes previously submitted by another user, the check-in fails and the differences must be resolved using Perforce. To minimize the complexity of merges, check in small sets of changes frequently.

To check in a file:

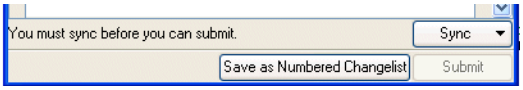
1. Right-click the file and choose **Check in...**

   The **Check in file(s)** dialog is displayed.

2. Click **OK**. The **Pending Changelist** form is displayed.
3. Enter comments and click **Submit**.

Resolving file conflicts

If you encounter conflicts when checking in files, (for example, from multiple users working on the same files,) you must resolve the conflicts before you can submit the files. For example, if you and another user have changed the same file and the other user checked in changes before you, when you attempt to check in your changes, Perforce notifies you that resolution is required by displaying the following message in the **Pending Changelist** dialog:



You must sync before you can submit.

To resolve file conflicts:

1. Click **Sync** and select the files to be resolved. The button text changes to **Resolve**.
2. Click **Resolve** and choose **All Files...** The **Resolve Files** dialog is displayed, describing the extent of the file differences and listing the following options:
   - *Accept Yours*: check your changes in, overwriting the most recently checked-in version.
   - *Accept Theirs*: discard your changes and preserve the other user’s changes.
   - *Merge*: launch the merge utility so you can view the differences between your version and the most recently checked-in version. Using the merge utility, you can select

individual passages or enter new changes to create the final version that you check in.

After you choose how the files are resolved, the **Pending Changelist** dialog is redisplayed, with the Resolve button text changed to **Submit**.

3. Enter a description of your changes and click **Submit**. Your changes are checked in.

Diffing files

To diff a file you are editing with the head revision in the depot, right-click the file and choose **Project > Source Control > Diff.**

To diff two revisions of a file, perform the following steps:

1. Select the file and choose **Project > Source Control > Show History...** The **Revision History** dialog is displayed.
2. Click and drag one of the revisions you want to diff, and drop it on the other revision.

P4Diff is launched, displaying file differences.

Reverting files

To discard any changes you have made to a file after checking it out, right-click the file and choose **Undo Check-out**. The head revision from the depot is copied to your client workspace, overwriting any changes you have made. Using Perforce, you can verify that the file is removed from the pending default changelist.

You can also revert unchanged files from the **Pending Changelist** dialog.

Working with Visual C++ files

Visual C++ displays a file status icon next to each file in the project window. In addition, the Source Control menu is context-sensitive: if you highlight a file, the menu selections are enabled or disabled depending on file status.

You can use the icon or **Source Control** menu appearance to identify file status.

File status	Icon appearance	Source Control menu appearance
Files are not under Perforce source code control	Original VC++ icon	**Add to Source Control** option is enabled

File status	Icon appearance	Source Control menu appearance
Files have been added to a changelist but not submitted to Perforce.	Icon is gray with red check mark	**Check in** and **Undo Check out** options are enabled
Files have been successfully submitted to Perforce.	Icon is gray	**Check out** option is enabled

Which files do I put in the depot?

Put the following files in your Perforce depot:

- `.dsp` files
- `.rc` files
- `.cpp` files
- `.h` files

Do not put IDE-generated files (such as `.ncb`, `.clw` and `.opt` files) in your Perforce depot. If you put the `.ncb` file under Perforce control, it is marked read-only when it's not checked out. If the `.ncb` file is read-only, class view information in VC++ is disabled.

If you intend to put the workspace file (`.dsw`) under Perforce control and multiple developers work on the same project, use relative paths to specify file locations to ensure that the entries in the workspace file work correctly on different client computers.

Location of project files

The FileView window displays source file names and their location on your local directory, but is only an approximate representation. Although the display simplifies project development in Visual C++, it differs from the actual file structure in the following ways:

- FileView groups files by type, such as Source files and Header files, whereas the local directory does not.
- FileView does not display the exact file names for the workspace file and the project file.

For some source control tasks, you need to specify the names and locations of these files. To view this information, right-click on the file and choose **Properties**. The **Source File Properties** dialog is displayed. The **General** properties option displays the complete file name and path.

Chapter 3 Microsoft Visual Basic

This chapter describes how to perform Perforce source code control tasks in the Visual Basic 6.0 environment. It assumes the Perforce SCC Plug-in is installed and that your client machine can communicate with the Perforce server.

The **Perforce** submenu lists the SCM operations you can perform. Note that the **Tools>Perforce>Create Project from Perforce...** option is not supported.

If the **Add-Ins > Add-In Manager...** menu item does not contain **Source Code Control** in the list box, you do not have the VB Source Code Add-In. For help getting it from Microsoft, contact `support@perforce.com`.

Configuring Visual Basic with Perforce

Before you can put Visual Projects under Perforce control, verify that source code control is enabled, as follows:

1. In Visual Basic, choose **Add-Ins>Add-In Manager...** The **Add-In Manager** dialog is displayed.
2. If **Source Code Control** is not listed, close Visual Basic, edit your `vbaddin.ini` file (located by default in the Program Files folder), add the following entry under `[Add-Ins32]`

   ```
   vbscc=3
   ```
3. Save the file.

Basic SCM tasks

This section tells you how to perform the following tasks.

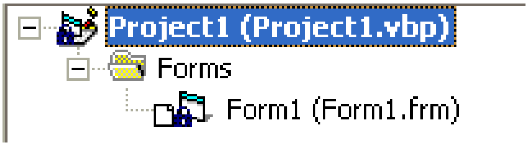
- “Adding a project to the depot” on page 24
- “Checking files out” on page 24
- “Retrieving files from the depot” on page 25
- “Checking files in” on page 26
- “Resolving file conflicts” on page 26
- “Diffing files” on page 27
- “Reverting files” on page 27

Adding a project to the depot

When you save your project, you are prompted “Add this project to Perforce?”

1. Click **Yes**. The **Open Connection** dialog is displayed. On the **Open Connection** dialog, specify Perforce settings as follows:
   - *Server*: the name of the host computer running the Perforce server in which you want to store the project.
   - *Port*: the port number on which the specified Perforce Server accepts commands.
   - *User*: the Perforce user name you want associated with the SCM operations you perform on the project. To choose from a list of users defined for the specified server, click **Browse**.
   - *Password*: the user’s password, if any.
   - *Client*: the Perforce client specification you want associated with the SCM operations you perform on the project. To choose from a list of client specifications defined for the specified server, click **Browse**.
2. Click **OK** to save settings and dismiss the dialog.
3. Right-click the project and choose **Check In...** The **Check In Files to Perforce** dialog is displayed.
4. Check the files you want to add and click **OK**. The **Pending Changelist** form is displayed.
5. Enter a comment and choose **Submit**. The files are added to the depot.

Checked-in files are displayed with a lock icon, as shown in the following figure.



Checking files out

To check out a file for edit, right-click the project file in the Project window and choose **Check Out** from the pop-up menu.

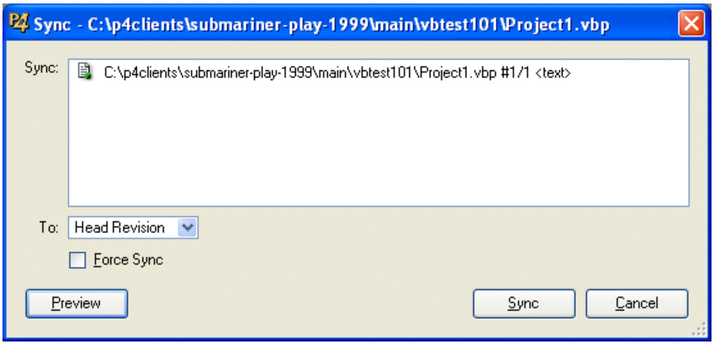
Checked out files are displayed with a red check mark, indicating that the files are writable.

Retrieving files from the depot

To get the most recent revision of a file from the depot, right-click the file and choose **Get Latest Version**.

To retrieve revisions prior to the head (latest) revision, you can sync from the **Revision History** dialog or perform the following steps:

1. Choose **Tools > Perforce >Get Latest Version...** The **Get Latest Version** dialog is displayed.
2. Check the files you want to retrieve and click **Advanced...** Dismiss the "Advanced sync dialog..." prompt by clicking **OK**, then click **OK** on the **Get Latest Version** dialog. The **Sync** dialog is displayed as shown in the following figure.



Sync - C:\p4clients\submariner-play-1999\main\vbtest101\Project1.vbp

Sync: C:\p4clients\submariner-play-1999\main\vbtest101\Project1.vbp #1/1 <text>

To: Head Revision

☐ Force Sync

3. Specify the revision you want to retrieve, using a Perforce changelist number, label, version number, or date. For details about specifying revisions, refer to the *Perforce User's Guide*.

- To preview the results of the operation with actually retrieving files, click **Preview**.
- To retrieve a file from the depot if you've deleted your local copy manually, check **Force Sync**. By default, Perforce assumes you still have your copy and doesn't retrieve it again.

4. Click **Sync**..

Note If you retrieve a file that you already have checked out, for example, to obtain changes checked in by another user, Perforce will require you to resolve the file, to enable you to merge your changes with changes made by other users and to ensure that you don't inadvertently overwrite other changes.

Note If you retrieve a file that you already have checked out, for example, to obtain changes checked in by another user, Perforce will require you to resolve the file, to enable you to merge your changes with changes made by other users and to ensure that you don't inadvertently overwrite other changes.

Checking files in

After editing checked-out files, you must check them into the depot. Your edited files become the head revision in the depot. (If any of your changes conflict with changes previously submitted by another user, the check-in fails and the differences must be resolved using Perforce.)

To check in files, perform the following steps:

1. Right-click and choose **Check in...** from the pop-up menu.

   The **Check in files to Perforce** dialog is displayed, listing the selected files.

2. Check the files you want to checked in and click **OK**.

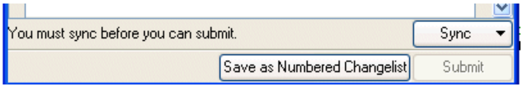
   The **P4 Submit** form is displayed.

3. Enter your comments and click OK.

Checked-in files are indicated with lock icons.

Resolving file conflicts

If you encounter conflicts when checking in files, (for example, from multiple users working on the same files,) you must resolve the conflicts before you can submit the files. For example, if you and another user have changed the same file and the other user checked in changes before you, when you attempt to check in your changes, Perforce notifies you that resolution is required by displaying the following message in the **Pending Changelist** dialog:



You must sync before you can submit.

To resolve file conflicts:

1. Click **Sync** and select the files to be resolved. The button text changes to **Resolve**.

2. Click **Resolve** and choose **All Files...** The **Resolve Files** dialog is displayed, describing the extent of the file differences and listing the following options:

   - *Accept Yours*: check your changes in, overwriting the most recently checked-in version.

- *Accept Theirs*: discard your changes and preserve the other user's changes.
- *Merge*: launch the merge utility so you can view the differences between your version and the most recently checked-in version. Using the merge utility, you can select individual passages or enter new changes to create the final version that you check in.

  After you choose how the files are resolved, the **Pending Changelist** dialog is redisplayed, with the Resolve button text changed to **Submit**.

3. Enter a description of your changes and click **Submit**. Your changes are checked in.

Diffing files

To diff a file you are editing with the head revision in the depot, right-click the file and choose **Tools > Perforce > Show Differences**.

To diff two revisions of a file, perform the following steps:

1. Select the file and choose **Tools > Perforce > Show History...** The **Revision History** dialog is displayed.
2. Click and drag one of the revisions you want to diff, and drop it on the other revision.

P4Diff is launched, displaying file differences.

Reverting files

To discard changes you've made to a checked-out file and reload the head revision from the depot, right-click the file and choose **Undo Check Out.** You can also revert unchanged files from the **Pending Changelist** dialog.

Working with Visual Basic files

Which files do I put in the depot?

In general, add the project file (such as `.vbp` files) but not the workspace file (`.vbw` files), especially if multiple developers are working on the same project. For example, if a developer has a workspace file checked out for edit and another developer adds a file to the project, the change is not reflected in the workspace file. Exclude the following Visual Basic file types from Perforce control; Visual Basic caches and recreates them as required.

- `.dca`: active designer cache
- `.oca`: control typelib cache
- `.vbg`: group file

You can exclude them using Perforce protections or client views to prevent them from being inadvertently added to the depot.

Location of project files

The FileView window displays source file names and their location on your local directory, but is only an approximate representation. Although the display simplifies project development in Visual Basic, it differs from the actual file structure in the following ways:

- The Project window groups source files by file type, such as Forms and Designers, but the local directory does not.
- The Project window does not display the workspace file (`.vbw` file) and some other files.

For some source control tasks, you need to identify the actual names and locations of these files. For detailed information about Visual Basic project files, see:

`http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon98/html/vbconformprojectfileformats.asp`

Visual Basic file pairs

The following table lists associated files that must be checked out together.

File Type	Check out in IDE	Check out using Perforce
Forms	`.frm`	`.frx`
UserDocument objects	`.dob`	`.dox`
UserControls	`.ctl`	`.ctx`
Property pages	`.pag`	`.pgx`

For example, if you check out a `.frm` file, be sure to check out its associated `.frx` file.

Recommended Perforce file types

Following are recommended Perforce file types for Visual Basic files. For details about Perforce file types and attributes, refer to the *Perforce User’s Guide*.

File type	Perforce file type	Description
`.bas`	`text`	Basic file
`.cls`	`text`	class file
`.ctl`	`text`	control file
`.ctx`	`binary`	control binary file
`.dsr`	`text+w`	designer file

File type	Perforce file type	Description
`.dsx`	`binary`	active designer binary file
`.frm`	`text`	form file
`.frx`	`binary`	form binary file
`.vbg`	`text+w`	group project
`.vbp`	`text`	project
`.vbw`	`text+w`	workspace

Checking out the project file

In Visual Basic 6.0 it is necessary to check out the project file (`.vbp` file) for edit to save *any* changes, including changes that affect only `.frm` files. If you do not want to submit an unchanged project file, right- click the project and choose **Undo checkout**. If multiple developers are working on the same project, keep the project file writable by setting its Perforce file type to `+w` when you add it to the depot.

Chapter 4 Microsoft Visual Studio .NET

This chapter describes how to perform Perforce source code control tasks in the Visual Studio .NET environment. It assumes the Perforce SCC Plug-in is installed and that your client machine can communicate with the Perforce server.

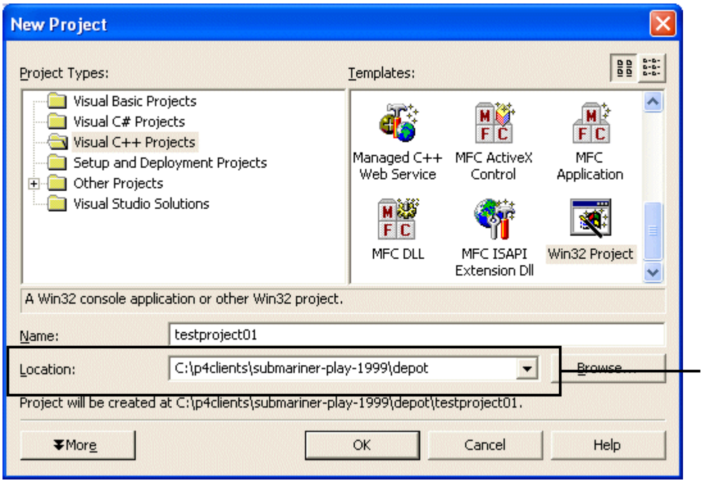
This screen illustrations in this chapter shows Visual C++, but the same dialogs are displayed regardless of which Visual Studio IDE you use.

Note The version control dialogs displayed for check in, check out, and undo checkout include comment fields, but these comment fields are not stored by the Perforce SCC Plug-in. To avoid displaying the check in dialog, choose **Check-in Now**.

Configuring Visual Studio .NET with Perforce

When you configure VS .NET with the Perforce SCC Plug-in, the settings are specific to the project you are working on, so no overall Perforce configuration is required for Visual Studio. The Perforce SCC Plug-in retains project settings from session to session and it is not necessary to re-enter them.

When you create a project you intend to manage with Perforce, specify a location that resides within the Perforce client root folder of the client you intend to use as illustrated in the following figure.



A Win32 console application or other Win32 project.

Name: testproject01

Location: C:\p4clients\submariner-play-1999\depot

Project will be created at C:\p4clients\submariner-play-1999\depot\testproject01.

Must be in client root folder

Basic SCM tasks

This section tells you how to perform the following tasks.

- "Adding a project to the depot" on page 32
- "Checking files out" on page 34
- "Retrieving files from the depot" on page 34
- "Checking files in" on page 35
- "Resolving file conflicts" on page 35
- "Diffing files" on page 36
- "Reverting files" on page 36
- "Miscellaneous tasks" on page 37

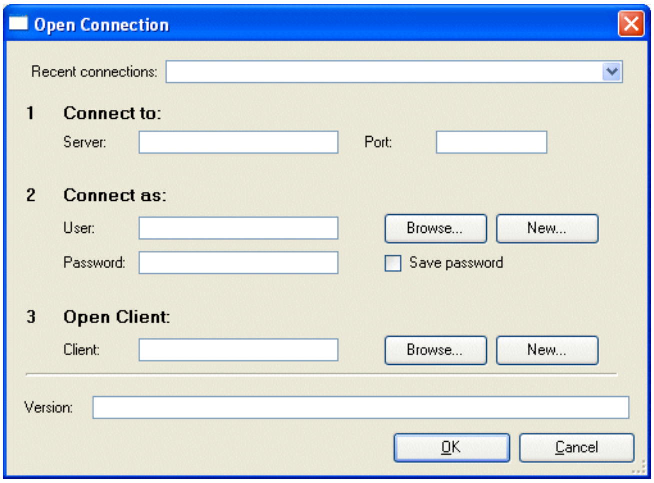
Also note that the **Exclude from source control** option is intended for files that, by nature, do not belong under source code control (such as build outputs).

Adding a project to the depot

Make sure your Visual Studio .NET projects reside within your Perforce client workspace.

To add a Visual Studio .NET project to your depot, perform the following steps:

1. In the Solution Explorer, right-click the project and choose **Source Control>Add Solution to Source Control**. The **Open Connection** dialog is displayed, as shown in the following figure.



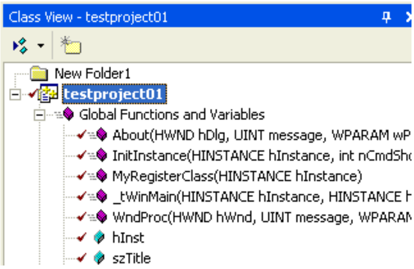
2. Enter the required settings as follows:
   - *Server*: the name of the host computer running the Perforce server in which you want to store the project.
   - *Port*: the port number on which the specified Perforce Server accepts commands.
   - *User*: the Perforce user name you want associated with the SCM operations you perform on the project. To choose from a list of users defined for the specified server, click **Browse**.
   - *Password*: the user's password, if any.
   - *Client*: the Perforce client specification want associated with the SCM operations you perform on the project. To choose from a list of client specifications defined for the specified server, click **Browse**.
3. Click **OK**. Project files are displayed in the **Pending Checkins** pane. (Using another Perforce client, you can verify that the files are open for add.)
4. In the Solution Explorer, right-click the solution and choose **Check In.**
5. The **Check In** dialog, listing files, is displayed.
6. Click **Check In...** The **Pending Changelist** dialog is displayed.
7. Enter a description in the **Description** field and click **Submit**. Your files are added to the depot. (Using Perforce, you can verify that the default changelist has been submitted and the files are now in the depot.)

Checked-in files are displayed with a lock icon. If you attempt to edit a file that is not checked out, a check-out dialog is displayed.

Checking files out

To check out files:

1. In the Solution Explorer pane, right-click the desired file (or the project, if you want to check out all its files) and choose **Check Out...** The **Check Out** dialog is displayed.
2. Click **OK**. In the Solution Explorer pane, files are displayed with a check mark, as shown in the following figure.


Class View - testproject01
New Folder1
testproject01
Global Functions and Variables
About(HWND hDlg, UINT message, WPARAM wP
InitInstance(HINSTANCE hInstance, int nCmdSh
MyRegisterClass(HINSTANCE hInstance)
_tWinMain(HINSTANCE hInstance, HINSTANCE h
WndProc(HWND hWnd, UINT message, WPARAM
hInst
szTitle

The checked-out files are listed in the **Pending Checkins** pane. Using another Perforce client, you can verify that the files are open for edit.

Retrieving files from the depot

To get the most recent revision of a file from the depot, right-click the file in the Solution Explorer pane and choose **Get Latest Version**.

To retrieve revisions prior to the head (latest) revision, you can sync from the **Revision History** dialog or perform the following steps:

1. In the Solution Explorer pane, click the file.
2. Choose **File > Source Control > History**. The **Revision History** dialog is displayed, listing the revisions in the depot.
3. Right-click the desired revision and choose **Sync Revision**.

Note If you retrieve a file that you already have checked out, for example, to obtain changes checked in by another user, Perforce will require you to resolve the file, to enable you to merge your changes with changes made by other users and to ensure that you don't inadvertently overwrite other changes.

To sync all files for a project that resides in the depot:

1. Choose File **> Source Control > Open from Source Control...** The **Open Connection** dialog is displayed.

2. Enter the settings that specify the server where the project is stored and the client workspace and user with which you want to check the project files out, and click **OK**. The **Perforce Open Project** dialog is displayed.
3. Browse to the `.sln` file for the project you want to sync and click **OK**. The project files are synced to your workspace.

Checking files in

After editing checked-out files, you must check them into the depot. Your edited files become the head revision in the depot. (If your changes conflict with changes previously submitted by another user, the check-in fails and the differences must be resolved using Perforce.).

To check in files, perform the following steps:

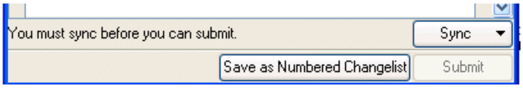
1. In the Solution Explorer pane, right-click the file and choose **Check In...** The **Check In** dialog is displayed.
2. Click **Check In**. The **Pending Changelist** dialog is displayed.
3. Enter a description of your changes and click **Submit**.

In the Solution Explorer pane, the check marks are replaced by locks, indicating that the submitted files were checked in.

Note To produce meaningful change history, check in related files together. For example, if you changed two different files to fix a bug, check them both in at the same time. The files are submitted in the same changelist.

Resolving file conflicts

If you encounter conflicts when checking in files, (for example, from multiple users working on the same files,) you must resolve the conflicts before you can submit the files. For example, if you and another user have changed the same file and the other user checked in changes before you, when you attempt to check in your changes, Perforce notifies you that resolution is required by displaying the following message in the **Pending Changelist** dialog:



You must sync before you can submit.

To resolve file conflicts:

1. Click **Sync** and select the files to be resolved. The button text changes to **Resolve**.

2. Click **Resolve** and choose **All Files...** The **Resolve Files** dialog is displayed, describing the extent of the file differences and listing the following options:
   - *Accept Yours*: check your changes in, overwriting the most recently checked-in version.
   - *Accept Theirs*: discard your changes and preserve the other user's changes.
   - *Merge*: launch the merge utility so you can view the differences between your version and the most recently checked-in version. Using the merge utility, you can select individual passages or enter new changes to create the final version that you check in.

   After you choose how the files are resolved, the **Pending Changelist** dialog is redisplayed, with the Resolve button text changed to **Submit**.
3. Enter a description of your changes and click **Submit**. Your changes are checked in.

Diffing files

To diff a file you are editing with the head revision in the depot, right-click the file and choose **Compare Versions...**.

To diff two revisions of a file, perform the following steps:

1. In the Solution Explorer pane, right-click the file and choose **File > History**. The **Revision History** dialog is displayed, listing the revisions in the depot.
2. To diff two revisions drag one revision to the other.

P4Diff is launched, displaying file differences.

Reverting files

To discard changes you've made to a checked-out file and reload the head revision from the depot:

1. In the Solution Explorer pane, right-click the file and choose **Undo Check Out**... The **Undo Check Out** dialog is displayed, listing the files to be reverted.
2. Check the files you want to revert and click **Undo Checkout**.

In the Solution Explorer pane, the reverted files are displayed with a lock icon, indicating that they are no longer checked out. The head revision is copied from the depot to your workspace, overwriting any changes you made to the workspace file.

You can also revert unchanged files from the **Pending Changelist** dialog.

Which files do I put in the depot?

Visual Studio .NET adds all appropriate files when the project is first added to source control, so, in general, don't change the contents of the changelist that is created when you choose **Add to Source Control**.

Put `.sln` files in the depot. Do not put the `.suo` files in the depot. The `.suo` files are managed by Visual Studio .NET separately for each client computer.

Miscellaneous tasks

To view the Perforce commands issued by the plug-in, choose **View>Other Windows>Output** and display the **Source Control** pane.

Chapter 5 Metrowerks CodeWarrior

This chapter describes how to perform Perforce source code control tasks in the CodeWarrior environment. Note that the Perforce CodeWarrior Plug-in supports Perforce features directly, whereas the Perforce SCC Plug-in maps Perforce features to generic functions like “check-in”. For details about managing your code using Perforce, and especially about using advanced features such as resolve and integrate, refer to the *Perforce User’s Guide*.

Note With the CodeWarrior plug-in, you cannot configure individual projects using a Perforce config file. (For details about Perforce config files, refer to the *Perforce User’s Guide*.)

Installing the CodeWarrior Plug-in

To obtain the Perforce CodeWarrior Plug-in, go to the Perforce downloads web page (`http://www.perforce.com/perforce/downloads/macosxppc.html`) and browse to the page for your Windows platform. After you download the file, copy it to the appropriate CodeWarrior directory, as follows:

- Macintosh pre-OS X:

  `Metrowerks CodeWarrior x.x:Metrowerks CodeWarrior:CodeWarrior Plugins`

- All other platforms:

  `Metrowerks CodeWarrior x.x/Metrowerks CodeWarrior/CodeWarrior Plugins`

Configuring CodeWarrior with Perforce

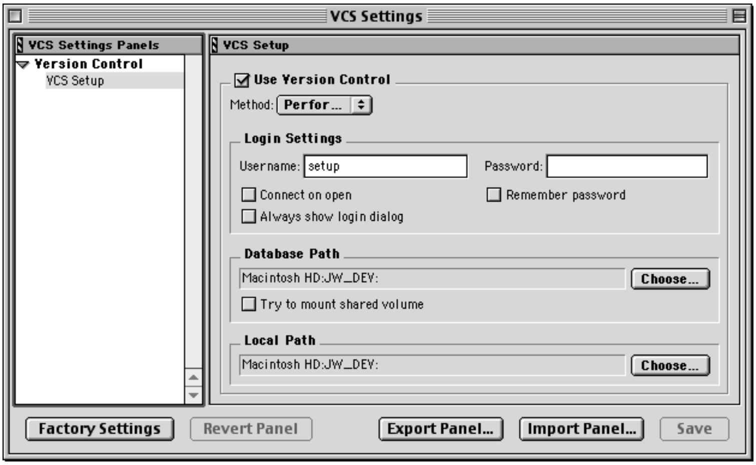
The instructions in this section apply to CodeWarrior for Macintosh and for Windows. The screen captures are for Macintosh. Although Windows screens differ in appearance from Macintosh screens, they are identical in content.

Warning! To avoid conflicts between project-level and global settings, which cause problems for CodeWarrior, always enter settings with a project open, and be sure to enter settings for each project. Do not use global version control settings (settings you make when no project is open); conflicts between project-level and global settings cause problems for CodeWarrior.

To configure the Perforce CodeWarrior Plug-in, perform the following steps.

1. Launch CodeWarrior.
2. Open the project for which you want to specify version control settings.
3. Choose **Edit > Version Control Settings...**

   CodeWarrior displays the **VCS Settings** dialog as shown in the following figure.



☑ Use Version Control

Method: Perfor...

Username: setup

Password:

☐ Connect on open

☐ Remember password

☐ Always show login dialog

Database Path: Macintosh HD:JW_DEV:

☐ Try to mount shared volume

Local Path: Macintosh HD:JW_DEV:

4. Check **Use Version Control** and choose Perforce as the method.

   If Perforce is not listed in the **Method** popup menu, the Perforce CodeWarrior Plug-in is not installed correctly. Verify that you extracted the plug-in to the Perforce CodeWarrior Plug-in directory.

5. Specify login settings. In the **Username** field, type the Perforce user name (the `P4USER` value) for the client workspace you want to link to this project.
6. If you use a password, set the **Login Settings** checkboxes as follows:

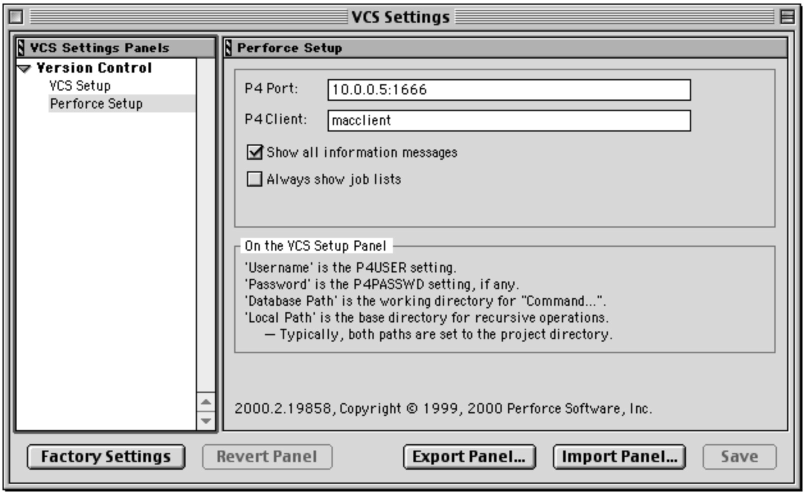
To be asked for your password...	Set “Always Show Login Dialog” to...	Set “Remember Password” to...
Never	Off	On
Once	Off	Off
Always	On	Off

Recommendation: to minimize the number of times you have to log in, turn off the checkbox **Always Show Login Dialog**, turn on **Remember Password**, and leave the **Password** field blank.

The **Connect on Open** checkbox determines whether you are asked to login when the project starts up or when you perform the first Perforce operation.

7. Click **Perforce Setup** in the left pane of the **VCS Settings** dialog.

   The **Perforce Setup** pane is displayed as shown in the following figure.



P4 Port: 10.0.0.5:1666

P4 Client: macclient

☑ Show all information messages

☐ Always show job lists

On the VCS Setup Panel

'Username' is the P4USER setting.
'Password' is the P4PASSWD setting, if any.
'Database Path' is the working directory for "Command...".
'Local Path' is the base directory for recursive operations.
— Typically, both paths are set to the project directory.

2000.2.19858, Copyright © 1999, 2000 Perforce Software, Inc.

   Set `P4PORT` and `P4CLIENT` to the Perforce server port and client workspace name, respectively.

   Set other preferences as desired. The following table describes the options.

Checkbox	Effect
Show all information messages	Specifies whether some Perforce operations display a dialog containing messages about the results of the operation.
Always show job lists	Specifies whether job lists are always displayed on the **Submit** form, or only when there are jobs already linked to the changelist.

8. Click Save to save your settings.

For each project, you must specify its working directories. In CodeWarrior, the *database path* is the current working directory for files specified using relative paths in the **VCS** menu's **Command...** menu item, and *local path* is the directory used as root for all of the **VCS** menu's **Recursive** commands. The local path must be set to the client workspace root or a directory under the client workspace root.

To specify project directories, open the project and perform the following steps.

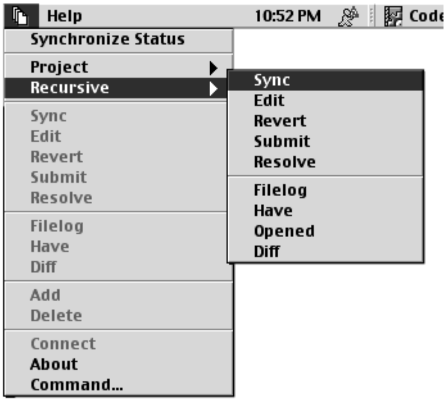
1. Specify the Database Path.
2. Choose “Project relative” and enter the project root.
3. Click Choose. CodeWarrior displays the **Change an Existing Path** dialog. Browse to the desired folder and click OK.
4. To specify the Local Path, enter the client workspace root or click Choose, browse to the client workspace root, and click OK.

The VCS menu

To perform source control operations, you choose options from the **VCS** menu. Most of the commands in this menu have the same name and functionality as the corresponding Perforce command. To issue Perforce commands other than those on the **VCS** menu, choose **VCS>Command...** or issue `P4` commands in MPW.

On Macintosh, the **VCS** menu is displayed as the following icon in the toolbar:

The **VCS** menu options appear as shown in the following figure.



The **Project** and **Recursive** menu selections enable you to specify the set of files to which you want the command to apply. The following table summarizes these options.

Menu selection	Command applies to
VCS	All files that you select in the project window.
VCS > Project	Only the current project file.
VCS > Recursive	All the files under the *{localpath}* directory that are mapped to files in the depot using the client view. Recursive commands act like `p4 command {localpath}/...` Recursive commands act on the project file if the project file is under *{localpath}*. Note: To run a recursive operation on a file subtree on Macintosh computers, hold down the Shift key while choosing the operation.

The *{localpath}* identifier refers to the value of the **Local Path** directory as set in the **VCS Settings** dialog. Most of the **Recursive** submenu items act on all the files under this directory, even when these files are not in the project.

Warning! Be careful when performing a recursive sync operation. Depending on your client mapping, you can inadvertently download the entire contents of the depot to your client computer, possibly hundreds or thousands of files. To avoid this problem, ensure that your client workspace is mapped to a limited and specific directory on the depot.

The Perforce CodeWarrior Plug-in has advanced options for many menu items. When you choose an advanced menu item, CodeWarrior displays dialogs that enable you to specify additional options before the operation is executed.

To view the advanced options on Macintosh platforms, press the **Option** key. Some of the menu selections appear with ellipses after them. For example, instead of **Sync**, the menu item becomes **Sync...** To choose an advanced option, press the **Option** key.

On Macintosh, no setup steps are required to enable the advanced options. On Windows platforms, the advanced option is enabled by specifying either **Advanced** or **Both** in the **Amount of Menu Detail** field during setup.

The following table describes the menu options.

Option	Description	Example	Advantages / Disadvantages
Simple	Only the basic menu options are displayed.	The **VCS** menu contains **Sync** but not **Sync...**	*Advantage*: shorter menus. *Disadvantage*: advanced operations such as `p4 sync -f` can't be performed directly from the **VCS** menu.
Advanced	For menu options that have both a basic and advanced version, the advanced version is displayed	The **VCS** menu contains **Sync...** but not **Sync**	*Advantage*: the plug-in displays dialogs for most of the commands, allowing more advanced options to be specified for each command, such as being able to change a file's type before you **Add** it to Perforce. *Disadvantage*: the dialogues are always displayed, adding steps to most **VCS** commands.
Both	For menu options that have both a basic and advanced version, both versions are displayed.	The **VCS** menu contains both **Sync** and **Sync...**	*Advantage*: you can choose between the simple and advanced form of the commands *Disadvantage*: the **VCS** menu size doubles.

To configure the display of advanced menu options:

1. Choose **Edit > Version Control Settings...**
2. In the **Amount of Menu Detail** field, specify either `Advanced` or `Both`.

The Perforce CodeWarrior Plug-in identifies file status by placing an icon next to each file in the project window. In addition, the **VCS** menu is context-sensitive: if you highlight a file, the menu selections are enabled or disabled depending on file status.

VCS menu commands

The **VCS** menu contains selections for **Project** and **Recursive** commands. Project commands operate on the project file, as opposed to source files. Recursive commands operate on all files in the directory set as the Local Path in the **VCS** options menu. You can display the recursive subtree of a Perforce operation on Macintosh computers by holding down the Shift key.

To display advanced menu options on Macintosh, hold down the Option key and click the menu item. To display advanced menu options in Windows, specify **Advanced** or **Both** in the **VCS setup** window.

The following table describes the version control commands that the Perforce CodeWarrior Plug-In adds to CodeWarrior.

Menu Command	Description	Option Key or Advanced Menu Action	Related P4 Command
Synchronize Status	Synchronizes the CodeWarrior file status with the Perforce file status. Refreshes the version control system icons for each file in the project. Execute this command after you perform an operation outside of CodeWarrior that affects the status of files in a CodeWarrior project.	None	None
Sync	Copies the head revisions of the selected files from the depot to the client workspace, if not already copied. To get all files from the depot for the first time, use **Recursive>Sync,** then add the files to your project using CodeWarrior's **Project>Add Files...** command. To retrieve specific revisions, enable advanced menus and use the **Sync...** command.	Enables you to enter revision specifications and other `p4 sync` options.	`sync`
Edit	Opens the selected files for edit in the client workspace.	Opens the files for edit and enables you to change the file types.	`edit`
Revert	Reverts the selected files to the revision last synced from the depot.	None	`revert`

Menu Command	Description	Option Key or Advanced Menu Action	Related P4 Command
Filelog	Displays the revision history of the selected files.	Displays the full description of each changelist.	`filelog`
Have (recursive)	Lists the revision numbers of the selected files that were last synced to the client workspace.	None	`have`
Diff	Diffs the revision as edited on the client workspace against the revision last synced from the depot, using the CodeWarrior diff utility.	Enables you to select the diff style, and to force the diff.	`diff`
Add	Add the selected files to the depot. To add files recursively, select all the files and folders in the project window and choose **Add**. To add the project file, use the **Project>Add** submenu	Enables you to set the file's Perforce file type.	`add`
Delete	Deletes the selected files from the depot. After you delete files from the depot, delete them from the CodeWarrior project using the **Project>Remove Selected Items** menu item. To delete a CodeWarrior project, use the **Project>Delete** submenu.	None	`delete`

Menu Command	Description	Option Key or Advanced Menu Action	Related P4 Command
Submit	Displays the submission dialog for the default pending changelist. If you have open files selected in the CodeWarrior project, those files are selected in the changelist. In the recursive submit dialog, all open files under {localpath} are selected in the changelist. To submit a numbered changelist, use the **Command...** menu item. If a **Submit** of the default pending changelist fails, Perforce assigns a number to changelist and you *must* use **Command...** to submit it. All jobs linked to the default pending changelist are displayed in the **Submit** dialog. To specify whether CodeWarrior displays an empty job list in the **Submit** dialog, enable the **Always Show Job List** checkbox in the **Perforce Settings** preferences panel. To add jobs to any changelist, use the resulting job list.	None	`submit`
Resolve	For files scheduled for resolve, displays a dialog enabling you to select and perform the desired type of resolve. Refer to the *Perforce User's Guide* for detailed information about resolving files using Perforce.	None	`resolve`
Opened (**Recursive** *submenu)*	Displays the list of all open files in the current client workspace.	Enables you to view a list of all open files in all client workspaces.	`opened`

Menu Command	Description	Option Key or Advanced Menu Action	Related P4 Command
Connect and Disconnect	Irrelevant for plug-in users. The VCS API requires use of this menu option, but Perforce has no corresponding feature.	None	None
About	Display information about your current Perforce configuration.	None	`info`
Command...	Enables you to run any Perforce command. File name arguments specified using relative pathnames are interpreted by Perforce using the database path specified in the **Version Control Settings** preferences panel. All Perforce special characters and wildcards are supported. To specify files or directories with spaces in their names, use quotes around the entire file argument; for example: `p4 changes "//depot/a b c"` The **Command...** menu item does not currently support: • Perforce commands that use Perforce's global options. • File or directory names containing quotes. • MPW special characters such as `>>>`. • Output redirection. If you need to use any of these features use the `P4` command line interface.	None	None

Basic SCM tasks

This section describes how to perform the following tasks:

- "Adding a project to the depot" on page 49
- "Checking files out" on page 50
- "Checking files in" on page 50
- "Diffing files" on page 51
- "Reverting files" on page 51
- "Deleting files" on page 52

Adding a project to the depot

To add all of the source files in a CodeWarrior project to version control, perform the following steps:

1. With the project open, choose **Edit >Select All**.

   All of the project files, including source files and other resource files, are highlighted.

2. Choose **VCS>Add**.

   For each source file that was added to the default pending changelist, the **VCS Messages** dialog displays a message similar to the following:

   ```
   //depot/projectdir/filename.c#1 - was added
   ```

3. Highlight the open files

4. Choose **VCS>Submit**.

   The **P4 Submit Form** dialog is displayed.

5. Enter your comments and click OK.

   The **VCS Messages** dialog displays a message similar to the following:

   ```
   //depot/projectdir/filename.c#1 - was submitted
   ```

Using Perforce, you can verify that the files are listed in the default pending changelist as opened for add.

Although the project file does not appear in the CodeWarrior project window, it does reside in the file system. Include the project file under version control, especially if multiple developers are working on the same project. Leave the project file checked out, so it can be modified automatically by the IDE if project resources change.

To add the project file to version control, perform the following steps.

1. Choose **VCS>Project>Add.**
2. Choose **VCS>Project>Submit.**
3. Choose **VCS>Project>Edit.**

Note that CodeWarrior displays a project icon in the lower left.

Tip: you can store the project file in XML format, which enables you to diff versions of the file to see how it's changed.

To verify that source files are under Perforce control, attempt to edit files that are not checked out. If the files are under Perforce control, CodeWarrior prompts you to check it out, remove the lock, or cancel the operation.

Checking files out

Files that are not checked out are designated by a crossed-out pencil icon, indicating that they are write-protected.

To check a file out of the depot so you can change it, click the file and choose **VCS>Edit**. If you want to change its file type, choose **VCS>Edit...** and specify the desired file type in the **P4 Edit Options** dialog.

Checking files in

In the project window, files marked by a pencil icon are open for edit.

After editing the open files, save them to your local directory and check the files into version control. To save the files on your local directory, choose **File>Save**; however, saving does not update the depot.

To check files into the depot, perform the following steps.

1. Select the files you want to check in.
2. Choose **VCS>Submit**.

   The **P4 Submit Form** dialog is displayed, containing a list of the selected files.
3. Check the files you want to check in.
4. Enter your comments in the **Comments** field and click OK.

In the **VCS Messages** dialog, a message similar to the following appears:.

```
Change 5059 created with 3 open file(s).
Submitting change 5059.
Locking 3 files ...
edit //depot/Demo3/DataEnvironment1.Dsr#2
edit //depot/Demo3/DataReport1.Dsr#2
edit //depot/Demo3/frmDataEnv.frm#2
Change 5059 submitted.
```

Diffing files

The Perforce diff utility P4Diff allows you to compare the contents of two text files. On Macintosh, you can diff Macintosh TEXT files and files stored in the depot using the `apple` file type with the `-t` option.

P4Diff enables you to compare a file that is open for edit with another version residing in the depot. You can use this capability to review changes before checking a file into version control, or compare your file with a previously submitted version of the file in case of a conflict.

To diff files:

1. In the Project window, click the file you want to compare.
2. Choose the menu option **VCS>Diff...**

   The **Perforce Diff Options** is displayed.
3. Specify desired diff options by clicking the corresponding radio button. By default, Perforce compares the open file with the version previously acquired from the depot. You can also compare any two previous file versions residing in the depot by entering the two version numbers.
4. Click OK.

The **Perforce Diff** window appears with the file differences highlighted.

Reverting files

To discard any changes you have made to a file after checking it out, right-click the file and choose **VCS>Revert**. The head revision from the depot is copied to your client workspace, overwriting any changes you have made. In Perforce, the file is removed from the default pending changelist.

Deleting files

To delete files from the depot:

1. Click the files and choose **VCS>Delete**.

   The **VCS Messages** dialog displays a message indicating that the files are opened for delete. Using Perforce, you can verify that the file is in the default pending changelist, opened for delete.

2. Choose **VCS>Submit**.

   The **P4 Submit Form** is displayed.

3. Check the files you want to delete, enter comments and click OK.

   The **VCS Messages** dialog is displayed, indicating that the changelist has been submitted.

Working with CodeWarrior files

How file status is displayed

The following table describes how CodeWarrior indicates file status.

File status	Icon appears as	VCS menu appears as
Files are recognized by CodeWarrior but not by Perforce	Gray lock	**Add** command is enabled, most others are inactive.
Files have been added to a changelist but not submitted to Perforce.	Pencil	Most commands, including **Submit**, are enabled.
Files have been successfully submitted to Perforce.	Pencil with slash	Check in

To display results for all Perforce commands, enable the **Display All Information Messages** option in the **VCS Settings** dialog. CodeWarrior displays the results of each command in its own window (labeled **VCS Message Window**). By default, results are displayed only when there is no visual change in the Project window.

Location of project files

The Project window has tabs for Files, Link Order and Targets. If you click the Files tab, the window displays source files and libraries, organized into collapsible lists or groups. The groups do not necessarily represent the actual file structure.

To view the location of a file, click and hold on a specific file until the pop-up menu appears, then choose **File Path**. The file's location on your local directory is displayed hierarchically.

The window does not display the project file (`.mcp` file), although this file resides in the project directory.

Choosing Perforce file types when adding files

Macintosh TEXT files are always stored as Perforce text files (without resource forks) so the files can be diffed, resolved, and edited on platforms other than Macintosh. For non-TEXT files, the file's extension, Finder type, and Creator are looked up in the Internet mappings database. If the database contains an entry mapping the file type, Perforce uses the Mac default format specified. If the database does not contain an entry, Perforce store the file using its apple file type.

When adding files to the depot, check the files' Perforce type before submitting the files to ensure that correct file types are assigned. To see the Perforce types of open files, use the **Opened** menuitem. To override the file's default Perforce type when adding the file to Perforce, hold down the Option key when choosing the **Add** menuitem, and select the file's type in the resulting dialog.

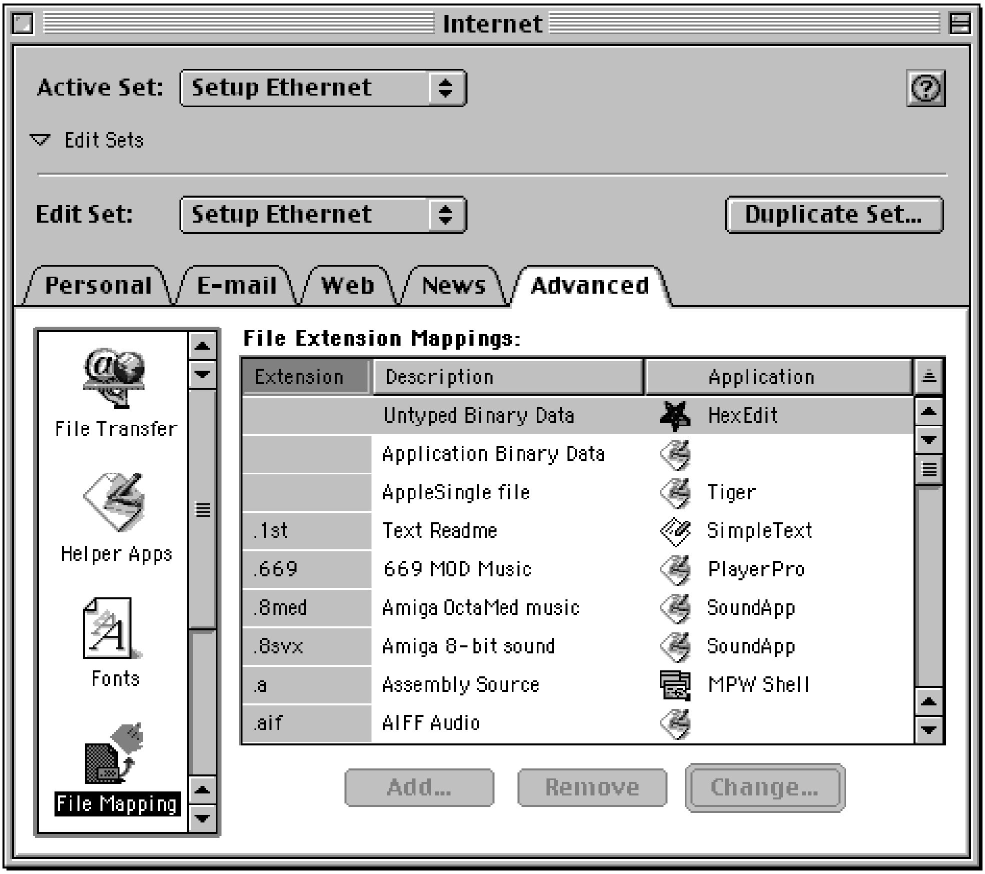
When you add portable binary files (for example, `.gif` files) to a Perforce depot, store them using the Perforce binary format, to avoid complications when the file is synced to a non-Macintosh computer.

Specifying default file types

When you add files (other than text files) from a Macintosh client to a Perforce depot, Perforce determines the file type by examining the Macintosh client's Internet mappings database. The factors that determine how a Macintosh file is stored are as follows:

- File name extension
- Finder type
- Creator

The following figure shows the **Internet** dialog with **File Mappings** selected at the left. The right pane shows file extensions mapped to Macintosh applications.



Active Set: Setup Ethernet

Edit Set: Setup Ethernet

Extension	Description	Application
	Untyped Binary Data	HexEdit
	Application Binary Data	
	AppleSingle file	Tiger
.1st	Text Readme	SimpleText
.669	669 MOD Music	PlayerPro
.8med	Amiga OctaMed music	SoundApp
.8svx	Amiga 8-bit sound	SoundApp
.a	Assembly Source	MPW Shell
.aif	AIFF Audio	

The following table lists standard mappings for Perforce.

Macintosh file type	Perforce file type
Macintosh	apple
Plain text	text
Binary data	binary

If the Internet mappings database does not contain an entry for a file type, Perforce stores it using the apple file type.

To change the default Perforce file type for all files of a particular combination of file extension, Mac file type, and Mac creator:

1. Open the Internet control panel and click the Advanced tab. (If the Advanced tab is not visible, choose **Edit>User** mode and set the mode to Advanced).
2. Click the File Mapping icon at the left of the Advanced tab dialog.
3. Scroll through the mappings list, looking for your extension, Mac file type, and Mac creator combination. If the combination you're looking for does not exist, you can add it by clicking Add... and entering the file extension, Mac type, and creator in the **Change Mapping** dialog.
4. Click Show Advanced Options.
5. On the right side of the **Advanced Options** dialog, choose the correct default format for this particular file type. The Macintosh format maps to the Perforce apple file type, Plain Text maps to text, and Binary Data maps to binary.
6. Save your changes and close the Internet control panel.

Chapter 6 TogetherSoft Together

This chapter describes how to perform Perforce source code control tasks in the Together environment. It assumes the Perforce SCC Plug-in is installed and that your client machine can communicate with the Perforce server.

When you configure Togther with the Perforce SCC Plug-in, the settings are specific to the project you are working on, so no overall Perforce configuration is required. The Perforce SCC Plug-in retains project settings from session to session and it is not necessary to re-enter them.

Basic SCM tasks

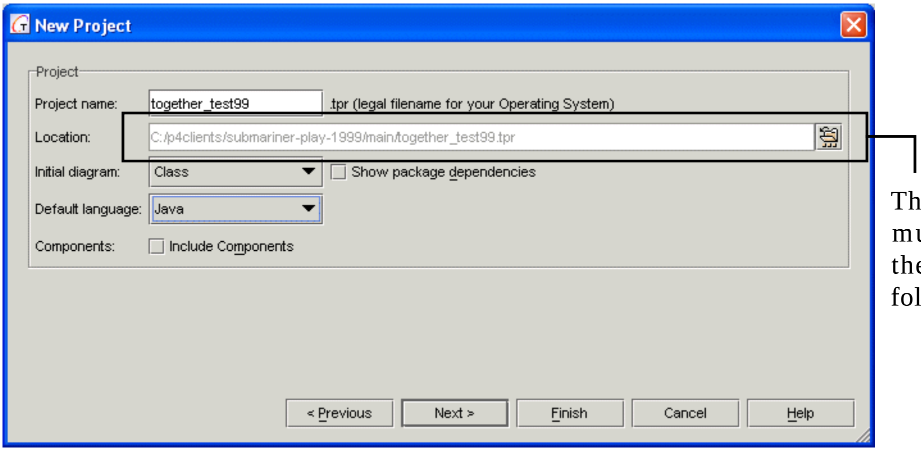
This section tells you how to perform the following tasks:

- "Adding a project to the depot" on page 58
- "Retrieving files from the depot" on page 59
- "Checking files out" on page 60
- "Checking files in" on page 60
- "Resolving file conflicts" on page 61
- "Diffing files" on page 61
- "Reverting files" on page 62

The **System** dialog, which you display by right-click and choosing **Version Control>System...**. from the pop-up menu, enables you to perform the full range of features supported by the Perforce SCC Plug-in for the Together IDE. The procedure in the following sections describe basic approaches to the most common tasks.

Adding a project to the depot

When you create a project that you intend to manage with Perforce, specify a location that resides within the Perforce client root folder of the client specification you intend to use to manage the project, as illustrated in the following figure.



Project name: together_test99 .tpr (legal filename for your Operating System)
Location: C:/p4clients/submariner-play-1999/main/together_test99.tpr
Initial diagram: Class ☐ Show package dependencies
Default language: Java
Components: ☐ Include Components

The project must reside in the client root folder.

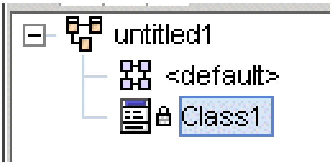
To configure Perforce as the SCM provider for a project:

1. Choose **Project > Project Properties...**
2. On the **Project Properties** dialog, check the **Version control project path** check box, then click **Options**.
3. On the **Project Options: Version Control** dialog, check **Version Control enabled** and choose “SCC” from the “Use” field drop-down list.
4. Click **OK** to dismiss the **Project Options: Version Control** dialog.
5. On the **Project Properties** dialog, click the button (lower right). The **Open Connection** dialog is displayed.
6. On the **Open Connection** dialog, specify Perforce settings as follows:
    - *Server*: the name of the host computer running the Perforce server in which you want to store the project.
    - *Port*: the port number on which the specified Perforce Server accepts commands.
    - *User*: the Perforce user name you want associated with the SCM operations you perform on the project. To choose from a list of users defined for the specified server, click **Browse**.
    - *Password*: the user’s password, if any.

- *Client*: the Perforce client specification want associated with the SCM operations you perform on the project. To choose from a list of client specifications defined for the specified server, click **Browse**.

7. Click **OK** to save settings and dismiss the dialog. The **Add Files to Perforce** dialog is displayed, listing files to be added.
8. Click OK to dismiss the dialog.
9. In the Project Explorer pane, right-click the project and choose **Check In...** The **Check in to Version Control** dialog is displayed.
10. Verify that all project files are checked and click **OK**. The **Pending Changelist** dialog is displayed.
11. Enter a comment and click **Submit**. Checked in files are listed with a lock icon, as shown in the following figure.

Checked-in files are displayed with a lock icon, as shown in the following figure.



Note If Together can not communicate with the Perforce server configured for a project, it disables the version control setting for the project. After reestablishing communication with Perforce, you must reselect this option manually. [TRUE?]

Retrieving files from the depot

To retrieve files from the depot:

1. In the Explorer pane, right-click the files you want to retrieve.
2. Choose **Version Control > Get...** The **Get from Version Control** dialog is displayed.
3. Check the files you want to retrieve.

To retrieve revisions prior to the head (latest) revision, you can sync from the **Revision History** dialog or perform the following steps:

1. Click the **Advanced** button, dismiss the “Advanced sync dialog...” prompt, and click OK. The Sync dialog is displayed. specify the revision you want to retrieve, using a Perforce changelist number, label, version number, or date, and click OK. For details about specifying revisions, refer to the *Perforce User’s Guide*.

- To preview the results of the operation without actually retrieving files, check **No file updates**.
- If you retrieve a file from the depot, then delete your local copy, you can retrieve it again by checking **Force resynchronization**. By default, Perforce assumes you still have your copy and doesn't retrieve it again.

2. Click OK

Note If you retrieve a file that you already have checked out, for example, to obtain changes checked in by another user, Perforce will require you to resolve the file, to enable you to merge your changes with changes made by other users and to ensure that you don't inadvertently overwrite other changes.

Checking files out

You must check out a project before adding, changing, or deleting packages.

To check out files from the depot for editing, perform the following steps:

1. Right-click the file you want to check out and choose **Version Control>Check Out...**

   The **Check out from Version Control** dialog is displayed, listing the files to be checked out.

2. Click **OK**.

   The **Perforce Message** dialog is displayed, indicating the files are opened for edit.

3. Click **OK** to dismiss the dialog.

In the Explorer pane, the file is displayed without a lock icon, indicating that it is checked out.

Checking files in

To check files you've edited, perform the following steps:

1. Right-click the file you want to add and choose **Version Control>Check In...** from the pop-up menu.

   The **Check in to Version Control** dialog is displayed, listing the files to be checked in.

2. Click OK.

   The **P4 Submit** form is displayed, listing the files to be submitted to the depot.

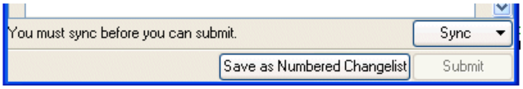
3. Enter your comments and click **OK**.

4. The **Perforce Message** dialog is displayed, listing the results of the submit operation.
5. Click **OK** to dismiss the dialog.

The file is now displayed with a lock icon, indicating it is checked in.

Resolving file conflicts

If you encounter conflicts when checking in files, (for example, from multiple users working on the same files,) you must resolve the conflicts before you can submit the files. For example, if you and another user have changed the same file and the other user checked in changes before you, when you attempt to check in your changes, Perforce notifies you that resolution is required by displaying the following message in the **Pending Changelist** dialog:


You must sync before you can submit.

To resolve file conflicts:

1. Click **Sync** and select the files to be resolved. The button text changes to **Resolve**.
2. Click **Resolve** and choose **All Files...** The **Resolve Files** dialog is displayed, describing the extent of the file differences and listing the following options:
   - *Accept Yours*: check your changes in, overwriting the most recently checked-in version.
   - *Accept Theirs*: discard your changes and preserve the other user’s changes.
   - *Merge*: launch the merge utility so you can view the differences between your version and the most recently checked-in version. Using the merge utility, you can select individual passages or enter new changes to create the final version that you check in.

   After you choose how the files are to be resolved, the **Pending Changelist** dialog is redisplayed, with the Resolve button text changed to **Submit**.
3. Enter a description of your changes and click **Submit**. Your changes are checked in.

Diffing files

To diff a file you are editing with the head revision in the depot, right-click the file and choose **Version Control > Diff**.

To diff two revisions of a file, perform the following steps:

1. Right-click the file and choose **Version Control > History.** The **Revision History** dialog is displayed, listing the revisions in the depot.

2. To diff two revisions drag one revision to the other.

P4Diff is launched, displaying file differences.

Reverting files

To discard any changes you've made to a file and restore the head revision from the depot, perform the following steps:

1. Right-click the file you want to add and choose **Version Control>UnCheck out...** from the pop-up menu.

   The **Undo check out from Version Control** dialog is displayed with the file selected.

2. Click **OK**.

You can also revert unchanged files from the **Pending Changelist** dialog.

Working with Together files

TogetherSoft suggests that all `.tpr` and `.df*` files be put under version control. The `.tpr` file is the Together project file. The `.df*` files are the UML diagram files. The `.tws` file contains the project settings. There is no reason to place this file under version control.

Index