Perforce 2004.2
Using IDE Plug-ins

September 2004

Table of Contents

Preface

About This Manual

This guide describes the configuration, and operation of Perforce plug-ins for a variety of integrated development environments (IDEs). For installation instructions, refer to the release notes for the SCC plug-in and the CodeWarrior plug-in.

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 Microsoft Visual C++, Visual Basic, and Visual Studio .NET
- 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:

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

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 54.

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.

Configuring Perforce preferences For Visual Studio IDEs

To configure Perforce preferences, display the settings control panel by choosing the corresponding menu entry:

- Visual Studio .NET: **Tools > Options > Source Control >SCC Provider** and click **Advanced...**
- Visual Basic 6.0: **Tools > Perforce > Options** and click **Advanced...**

- Visual C++ 6.0: **Tools > Options**, scroll right and click the **Source Control** tab and click **Advanced...**

The following sections describe the Perforce settings you can configure on each tab.

General tab

Option	Description
Check server for updates every *n* minutes	Frequency with which the Perforce server is queried for updated file status. Performance consideration: frequent checks enable you to view up-to-date status information but Perforce server performance can be adversely affected if numerous users are constantly querying the server.
Number of entries to fetch at a time	The maximum number of updates returned when the Perforce server is queried. Performance considerations: a high limit helps ensure that you see all the entries, but larger data transmissions incur more network overhead.
Logging Options:	
Log all	Logs all commands sent to the Perforce server. To long only user-initiated commands, disable this option.
Enable logging to file	Specifies the name, location and maximum size of the log file.
Enable diff2 on file-to-file drag and drop	Enables you to diff files in the depot using drag and drop. To prevent inadvertently launching a diff (if you never use drag-and-drop diffing), disable this feature.
Warn before reverting files	Displays a dialog enabling you to change your mind before reverting files. When you revert an open file, any changes you made are discarded.

Connection tab

Option	Description
When binding a project to source control	Specifies how you want to configure the Perforce server and workspace associated with the solution. **Bind to the workspace that matches your Perforce environment settings**: uses the settings in effect when you add the project ("global" settings). **Show the Perforce connection dialog:** enables you to specify the settings manually. If you are adding a solution that contains a large number of projects, choose **Bind to the workspace that matches your Perforce environment settings** and configure the P4PORT and P4CLIENT settings to specify the server and workspace. If you choose **Show the Perforce connection dialog**, you must manually enter settings for each project in the solution.
When connecting to a server	For 2004.2 server and above: Enables / disables display of login-related dialogs. Perforce servers that are running in a high security mode can set a session time limit. After the session expires, you are prompted to log in when you attempt another Perforce operation.
Show Remember Password dialog	For 2003.2 server and below: if enables, the plug-in displays a "Remember Password" dialog when prompting you for your password. If disabled, the plug-in prompts you every time the Perforce server requires your password.
Show Login Expiration dialog	For 2004.2 server and above: enables / disables display of a dialog that displays session information when you connect to a server running at security level 3.

Diff tab

Option	Description
Default diff application	Enables you to select the application that is used to diff your files. Optionally enables you to configure different applications for specific file types.

Merge tab

Option	Description
Default merge application	Enables you to select the application that is used to perform three-way merges when you resolve files.

Version tab

This tab displays version information about the Perforce SCC Plug-in.

Font tab

This tab enables you to configure the font used by Perforce merge, diff, and time-lapse view utilities.

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 Perforce Command-Line Interface (P4) 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.

Task	Description	Perforce activity
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. For details about configuring settings, see “Configuring IDEs with plug-ins” on page 12.

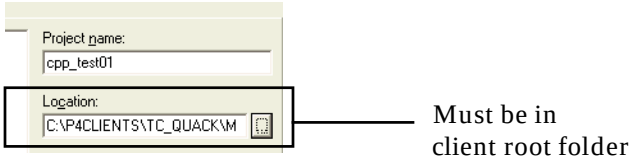
Visual C++ has a **Source Code Control** toolbar: to enable it, right-click the toolbar and check **Source Code Control** on the pop-up menu.

Basic SCM tasks

This section tells you how to perform the following tasks:

Adding a project to the depot

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.



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

1. Click **Yes**. If you enabled the **Show the Perforce Connections** option on the **Preferences > Connection** tab, 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

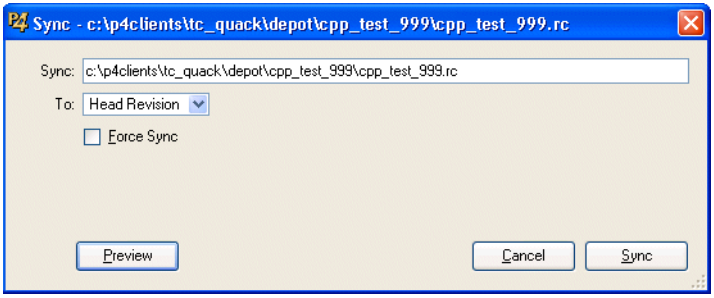
To get the most recent file revisions from the depot:

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.



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...**

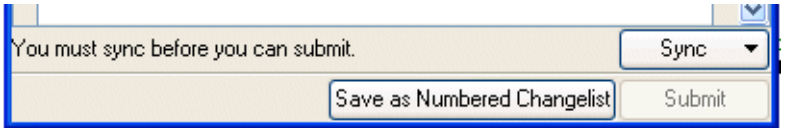
   The **Check in file(s)** dialog is displayed.

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:



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
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. For details about configuring settings, see “Configuring IDEs with plug-ins” on page 12.

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

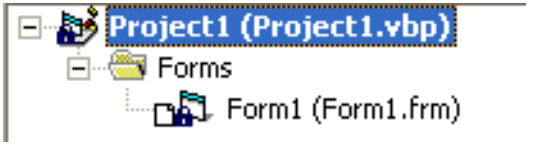
This section tells you how to perform the following tasks.

Adding a project to the depot

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

1. Click **Yes**. If you enabled the **Show the Perforce Connections** option on the **Preferences > Connection** tab, 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.

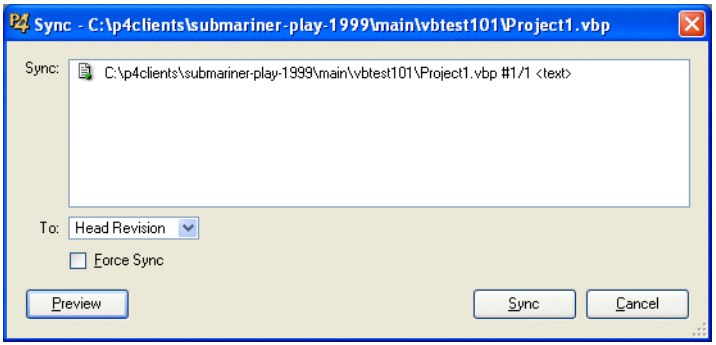
Checked out files are displayed with a red check mark, indicating that the files are writable.

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.



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**.

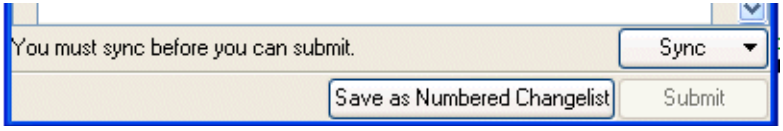
   The **P4 Submit** form is displayed.

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:



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

File type	Perforce file type	Description
.dsr	text+w	designer file
.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. For details about configuring settings, see “Configuring IDEs with plug-ins” on page 12.

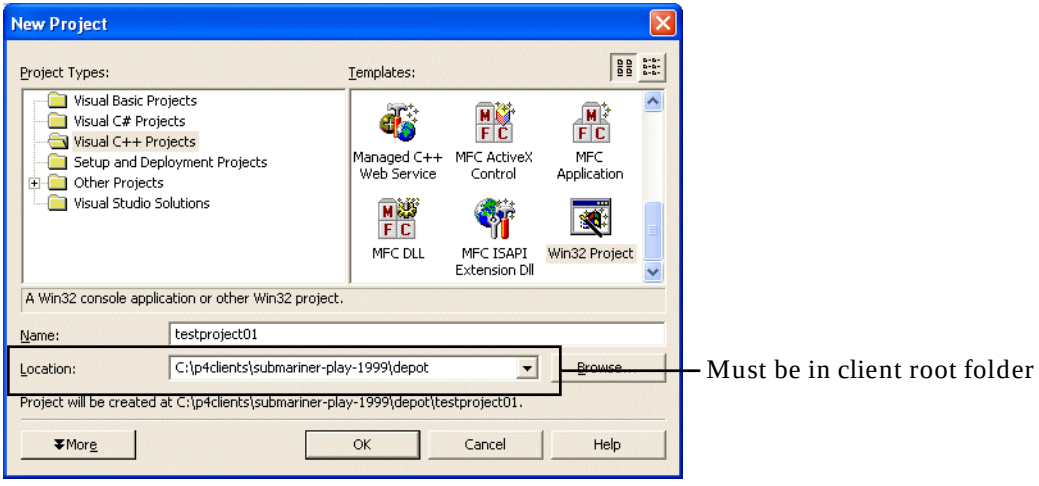
This screen illustrations in this chapter shows Visual C++, but the same dialogs are displayed regardless of which Visual Studio IDE you use.

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.



Basic SCM tasks

This section tells you how to perform the following tasks.

- "Adding a project to the depot" on page 36.

  If you are adding a solution that contains a large number of projects, you can avoid specifying settings manually for every project by choosing the **Options > Connection** tab setting **Bind to the workspace that matches your Perforce environment settings** and setting `P4CLIENT` and `P4PORT` to the desired server and workspace.

- "Checking files out" on page 38
- "Retrieving files from the depot" on page 39
- "Checking files in" on page 40
- "Resolving file conflicts" on page 40
- "Diffing files" on page 41
- "Reverting files" on page 41
- "Miscellaneous tasks" on page 42

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).

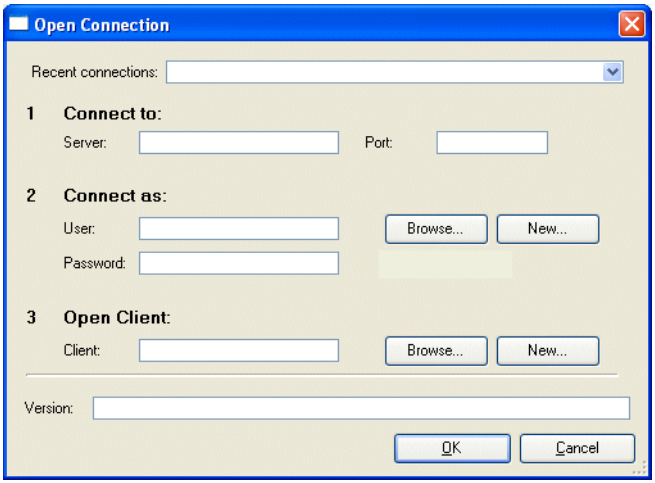
Adding a project to the depot

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**. If you enabled the **Show the Perforce Connections**

option on the **Preferences > Connection** tab, the **Open Connection** dialog is displayed. .



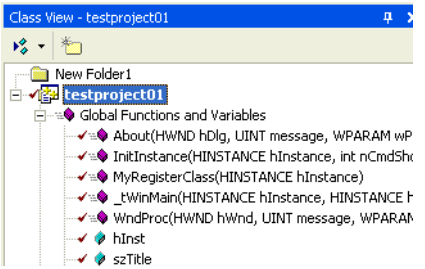
2. On the **Open Connection** dialog, enter the required settings as follows:
   - *Server*: the name of the host computer running the Perforce server in which you want to store the project.
   - *Port*: the port number on which the specified Perforce Server accepts commands.
   - *User*: the Perforce 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.

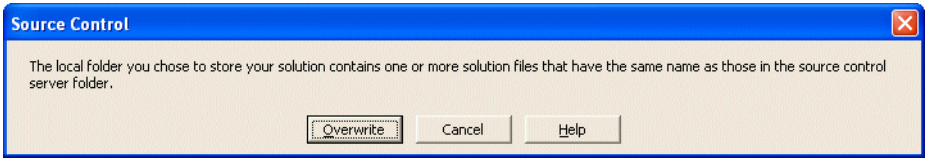


The checked-out files are listed in the **Pending Checkins** pane. Using another Perforce client, you can verify that the files are open for edit.

If you branch a project using Perforce, the Visual Studio bindings must be corrected to correspond to the project's new location. To correct bindings for a branched solution, you must perform the following steps the first time you check out the branched project.

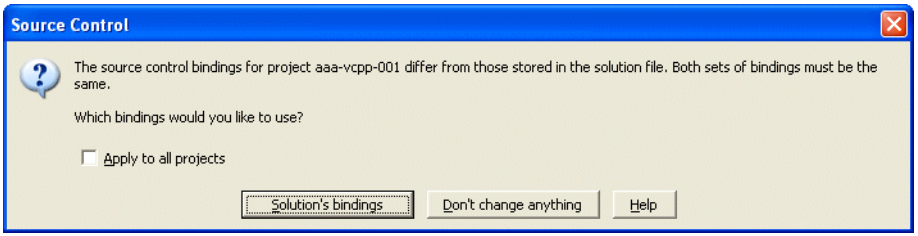
To check out a newly-branched project (first time only):

1. Choose **File> Source Control > Open from Source Control...**. The **Open Connection** dialog is displayed.
2. Specify connection settings for the server that contains the project you want to check out and click **OK**. The **Open Project** dialog is displayed.
3. Browse to the branched project you want to open and double-click its `.sln` file.
4. If the newly-branched files are present in your workspace, Visual Studio displays the following dialog:

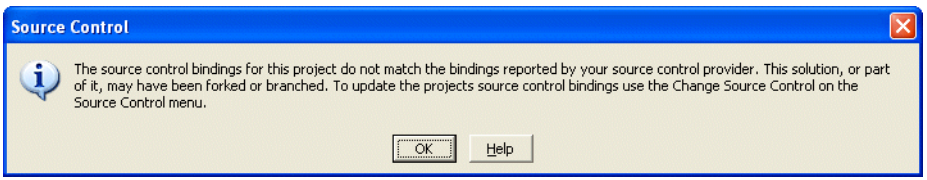


5. Click **Overwrite**. Visual Studio displays the **Open Solution** dialog.

6. Double-click the `.sln` file. Visual Studio displays the following dialog.



7. Click **Solution's bindings**. Visual Studio displays the following dialog.



8. Click **OK**.

The project now contains correct binding information.

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 requires you to resolve the file, to ensure that you don't inadvertently overwrite other users' 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:

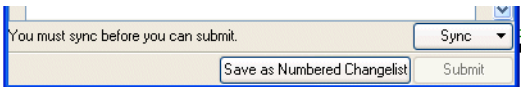
1. In the Solution Explorer pane, right-click the file and choose **Check In...** The **Check In** dialog is displayed.
2. Click **Check In**. The **Pending Changelist** dialog is displayed.
3. Enter a description of your changes and click **Submit**.

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:



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, but the Perforce SCC Plug-in maps Perforce features to generic functions (for example, "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.*)

Configuring CodeWarrior with Perforce

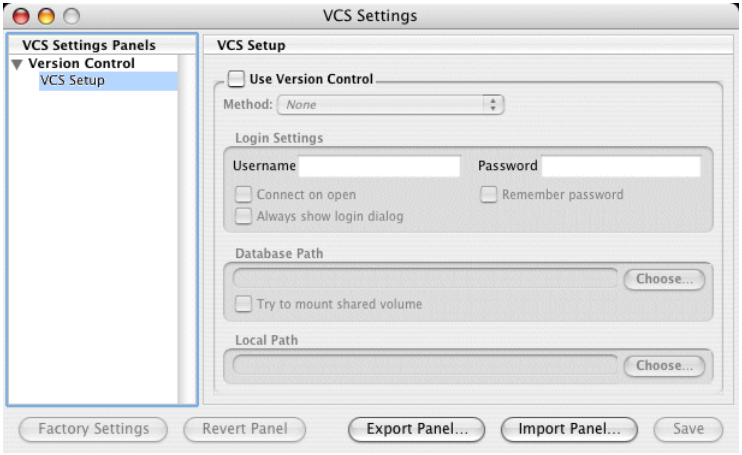
The instructions in this section apply to CodeWarrior for Macintosh and for Windows. Although Windows CodeWarrior screens differ in appearance from Macintosh screens, they are identical in content. Note that, on Macintosh the VCS menu is represented by this icon in the menu bar: 

Warnings!
- 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.
- Do not change file read / write settings manually. Perforce manages these settings and uses them to track the status of files.

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.



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 copied the plug-in files to the Perforce CodeWarrior plug-in directory as described in "Configuring CodeWarrior with Perforce" on page 43.

5. Specify login settings. In the **Username** field, type the Perforce user name (the `P4USER` value) of the owner of the client workspace where you want to store this project.

6. If the specified Perforce user has a password, set the **Login Settings** checkboxes as follows:

To be asked for your password	Always show login dialog	Remember password
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.

For each project, you must specify its working directories:

- Database path: the current working directory for files specified using relative paths in the **VCS** menu's **Command...** menu item.
- Local path: the directory used as root for all of the **VCS** menu's **Recursive** commands.

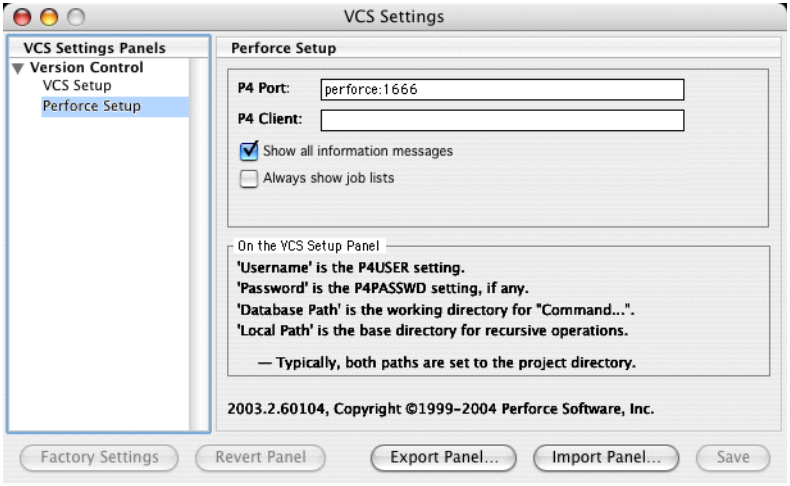
To specify working directories for the **Database Path** and **Local Path** fields on the **VCS Setup** pane:

1. Click **Choose**. CodeWarrior displays a folder browser dialog.
2. Specify the desired **Path Type** (for details, refer to CodeWarrior on-line help).
3. Browse to the desired folder (the folder in the client workspace where you want to store the project files) and click **OK**.

To configure the settings required for CodeWarrior to interact with the Perforce Server, perform the following steps.

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

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



   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.

2. Click **Save** to save your settings.

The VCS menu

Scope of menu options

To perform source control operations, you choose options from the **VCS** menu or one of its submenus, according to the set of files you want to operate on.

To operate on...	Choose commands from this menu
Files and folders selected in the IDE project window	**VCS**
The project file itself	**VCS > Project**
All files and folders in the path specified in the **Local Path** field on the **VCS Settings** pane	**VCS > Recursive**

To run a recursive operation on a file subtree on Macintosh computers, hold down the Shift key while choosing the operation.**VCS** menu commands operate on the files and folders selected in the IDE project window.

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. For details about client views, refer to the *Perforce User Guide*.

To issue Perforce commands other than those on the **VCS** menu, choose **VCS>Command...** or issue `P4` commands at the operating system command line prompt.

Simple versus advanced menu options

There are two types of menu options: *simple* and *advanced*. You can configure CodeWarrior to display simple, advanced, or both sets of options. The following table describes the menu settings.

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 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 Windows platforms, enable the advanced menu options by choosing **Advanced** or **Both** in the **Amount of Menu Detail** field on the **VCS Settings > Perforce Settings** pane. To configure the display of advanced menu options:

1. Choose **Edit > Version Control Settings...** The **VCS Settings** dialog is displayed.
2. In the left pane, click **Perforce Settings**. The **Perforce Settings** pane is displayed.
3. In the **Amount of Menu Detail** field, choose the level of menu options you want displayed by default.

VCS menu commands

The VCS menu commands work as follows:

- A plain command (for example, **Sync**) performs the specified operation using defaults (for example, sync to the head revision).
- A command followed by an ellipsis (for example **Sync...** or **Sync Subtree...**) displays a dialog where you can specify command options.
- A command followed by "Subtree" performs the specified operation recursively (on the files in a selected folder plus its subfolders).

The following table describes the commands that the Perforce CodeWarrior Plug-In adds to CodeWarrior in the **VCS** menu and its submenus.

Menu Command	Description
About	Display information about your current Perforce configuration.
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.
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. To use the Perforce `*` and `?` wildcards, specify file names using depot syntax. The Perforce `...` wildcard works with both local and depot syntax. For details about wildcards and depot and local syntax, refer to the *Perforce User's Guide*. 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 support: • Perforce commands that use Perforce's global options. (See the *Perforce Command Reference* for a list of global option flags.) • 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.

Menu Command	Description
Connect and **Disconnect**	Irrelevant for plug-in users. The CodeWarrior VCS API requires use of this menu option, but Perforce has no corresponding feature.
Delete	Marks the selected files for deletion from the depot. Files are deleted when you submit the changelist in which they are marked. 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.
Diff	Diffs the revision in the client workspace against the revision last synced from the depot, using the CodeWarrior diff utility.
Edit	Opens the selected files for edit in the client workspace.
Filelog	Displays the revision history of the selected files.
Have	Lists the revision numbers of the selected files that were last synced to the client workspace.
Opened	Lists all open files in the current client workspace.
Opened -a	Lists all files opened in all client workspaces.
Properties	Displays Perforce information about the selected files.
Resolve	For files scheduled for resolve, displays a dialog enabling you to select and perform the desired type of resolve. (For interactive resolves, you must use a full-featured Perforce client application such as P4V or the `p4` command-line client.) Refer to the *Perforce User's Guide* for detailed information about resolving files using Perforce.
Revert	Reverts the selected files to the revision last synced from the depot.

Menu Command	Description
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.
Sync	Copies 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, use the advanced **Sync...** command.
Synchronize Status | Selection	Refreshes the version control status displayed on the icons for each file in the project or for selected files. Execute this command after you perform an operation outside of CodeWarrior that affects the status of files in a CodeWarrior project.

Basic SCM tasks

This section describes how to perform the following tasks:

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, CodeWarrior displays a message similar to the following:

   ```
   //depot/CW/cw-test100/cw-test100/hello.cpp#1 - opened for add
   ```

3. Choose **VCS>Submit**.

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

4. Enter your comments and click OK.

CodeWarrior displays messages similar to the following:.

```
Change 6426 created with 1 open file(s).
Submitting change 6426.
Locking 1 files ...
add //depot/CW/cw-test100/cw-test100/hello.cpp#1
Change 6426 submitted.
```

Include the project file under version control, especially if multiple developers are working on the same project. Although the project file does not appear in the CodeWarrior project window, it does reside in the file system. 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**, enter your comment on the submit form and click **OK.**
3. Choose **VCS>Project>Edit.**

Tip: export the project file in XML format and maintain the XML file under Perforce control, to enable you to diff versions of the file to see how it has changed.

To verify that source files are under Perforce control, edit a file that is not checked out. If the file is under Perforce control, CodeWarrior prompts you to check it out, make it writable, 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. (Note that the **Submit** menu option is enabled only when files are selected, but **Recursive>Submit** is always enabled.)
3. Check the files you want to check in.
4. Enter your comments in the **Comments** field and click **OK**.

CodeWarrior displays a message similar to the following:

```
Change 6429 created with 1 open file(s).
Submitting change 6429.
Locking 1 files ...
edit //depot/CW/cw-test100/cw-test100/hello.cpp#2
Change 6429 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 a file:

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 (indicating the file is writable) 	Most commands, including **Submit**, are enabled.
Files have been successfully submitted to Perforce	Pencil with slash (indicating the file is not writable) 	Check in

To display results for all Perforce commands, enable the **Show 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** menu item. To override the file's default Perforce type when adding the file to Perforce, hold down the Option key when choosing the **Add** menu item, 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 that otherwise might occur when the file is synced to a non-Macintosh computer.

Index