

**Team  
Coherence**

[www.teamcoherence.com](http://www.teamcoherence.com)

## **Version Manager 7.1.2.55 Rough Guide**

Copyright © Quality Software Components Ltd



<b>INTRODUCTION.....</b>	<b>4</b>
WHAT IS TEAM COHERENCE.....	4
VIEWS AND PARALLEL DEVELOPMENT.....	4
ARCHITECTURE.....	4
EFFICIENT NETWORKING.....	4
MULTI-PLATFORM.....	5
SIMPLE ADMINISTRATION.....	5
IDE SUPPORT.....	5
COMPREHENSIVE API.....	5
TRIGGER SUPPORT.....	5
EXTENSIBILITY API.....	5
<b>VISUAL ELEMENTS.....</b>	<b>6</b>
1) MAIN MENU / TOOLBAR.....	6
2) FOLDER PANE.....	6
3) FILES PANE.....	6
4) DETAIL PANE.....	7
5) INFORMATION PANE.....	7
<b>COMMON ACTIVITIES.....</b>	<b>8</b>
ADDING FILES.....	9
<i>Drag / Drop from Windows Explorer.....</i>	<i>9</i>
<i>Manually adding files to existing Folders.....</i>	<i>10</i>
GETTING FILES.....	11
CHECK-OUT FILES TO BE MODIFIED.....	13
CHECK-IN MODIFIED FILES.....	15
BASELINE A PROJECT OR FOLDER.....	18
CHECKING IN YOUR LOCKED FILES.....	19
FINDING FILES.....	20
USING VIEWS.....	22
1) <i>Project/Version based Views.....</i>	<i>23</i>
2) <i>Promotion Level based Views.....</i>	<i>24</i>
DEPLOYING FILES.....	25

The following sections will first give an overview of Team Coherence Version Manager and will follow up with a description of some common activities. Where Tracker integration applies to the development process, it will be mentioned, but the aim is to describe the Version Control functionality of Team Coherence.

Note that the description of the architecture applies both to Version Manager and Tracker.

# Introduction

## ***What is Team Coherence***

Team Coherence is a powerful, easy to use Software Configuration Management (SCM) System built on concepts that greatly simplify common tasks for small and large development teams. Team Coherence's client/server, TCP/IP based architecture is designed for heterogeneous, local- and wide-area networked environments. Team Coherence is easy to use, requires little administration, integrates with leading development environments and, most importantly, allows you to focus your energies on your product rather than ours!

## ***Views and Parallel Development***

Team Coherence's unique strength is the simple support for branching and parallel development provided by the simple concept of Views. A View is an image of selected projects at either a specified Version or Promotion Level. While working within a View, automatic branching makes sure that changes made to a particular version of a project do not affect the state of other versions of the project. In addition, each View can have its own workspace settings so that you can work on multiple versions of a project on the same machine.

Team Coherence supports either parallel or serial development modes, or a combination of the two. With parallel development, more than one user can work on the same revision of the same file and Team Coherence will automatically handle branching when the file is checked in. With serial development, only one user can have a file checked out at a time. Serial development can cause bottlenecks in the development process as developers have to wait for other users to finish with files before they can make their modifications. Team Coherence handles this by allowing individual, perhaps commonly required, files to be marked as supporting multiple locks.

## ***Architecture***

Team Coherence consists of server and client components. The Team Coherence server provides controlled access to the central file archive on the server's machine. Users access the Team Coherence server through client programs running on their local workstations anywhere on the network.

Team Coherence Server handles most of the Version Control logic and has an extremely small footprint. The server is capable of handling files of any size and its unique architecture allows extremely fast access to relevant information by the client applications. The repository format used by Team Coherence Server is our own proprietary format developed over more than seven years and is optimized specifically for the purpose of managing the version control of files.

Because Team Coherence is a client-server tool, unlike many of our competitors, users don't require rwd access to the repository. This completely eliminates the chances of accidental or malicious deletion of repository files and minimizes the chances of network problems causing corruption of the repository data.

## ***Efficient Networking***

Team Coherence server and clients communicate using a TCP/IP-based protocol designed for efficient network performance. Team Coherence does not use shared file access and works well in local- and wide-area networks. Enhanced batching and streaming is employed so that access to the repository is fast, even across high-latency connections typical with satellite. Once files are retrieved to a workspace, all accesses are local to the user's machine. Normal editing and compiling operations do not use the network.

## ***Multi-platform***

Team Coherence runs natively on Windows or Linux. Under Windows, users have access to the full GUI application and a command line tool. Under Linux, users currently have access to the command-line tool.

Under both Windows and Linux, the full Direct-API is accessible to developers to allow development of project-specific and additional client applications.

## ***Simple Administration***

Team Coherence largely manages itself. Routine administration consists of performing backups, which can be done without stopping the Team Coherence server.

## ***IDE Support***

Team Coherence supports Microsoft's Source Code Control (SCC) API, providing integration with the Developer Studio suite of products (Visual C++, Visual Basic, et al) and many other tools. The Team Coherence SCC API module uses the Team Coherence API to provide top performance and tight integration.

In addition to support for the SCC API, Team Coherence integrates directly with IDE's of Borland's Delphi and C++Builder products.

## ***Comprehensive API***

Team Coherence has a comprehensive API that allows developers to integrate Team Coherence with their own development tools and processes, if they are not already supported. The full API provides access to high-level functions that utilize the built-in dialogs and windows implemented by the GUI client, while a lower-level API (Direct API) gives you access to functions not requiring a user interface. The Direct API is supported under both Windows and Linux.

## ***Trigger Support***

Team Coherence supports both client- and server-side triggers for most common activities. Triggers can be used to implement logging functionality and to influence the progress of certain actions.

Full documentation and sample triggers (with source code) can be downloaded from our web site.

## ***Extensibility API***

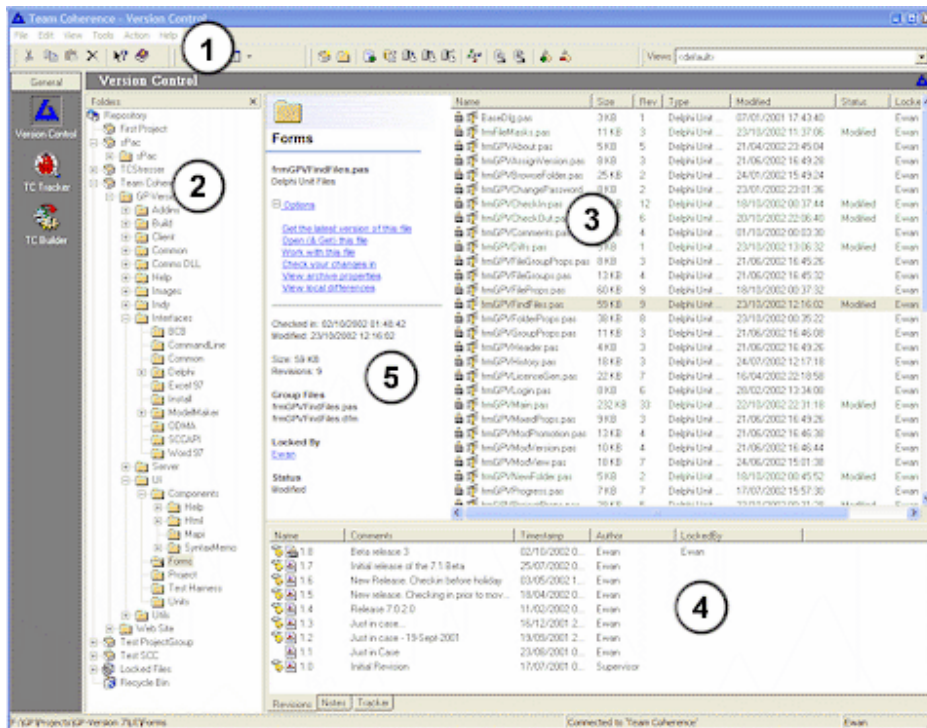
Although not yet documented, Team Coherence has been designed from the ground-up to be extensible. Using this API, you can extend many of the standard dialogs to add additional functionality and to add additional modules to the Team Coherence suite of tools.

The object model used by Team Coherence is extendible without server updates and most objects (Projects, Folders, Files, Revisions, Version Labels, etc) can be extended to associate additional information with them. An example of this extensibility is the way that Version Manager and Tracker are integrated with each other.

# Visual Elements

The main visual element in Team Coherence is the Version Manager. Version Manager allows you to get an overall view of the projects contained in a repository. It is also where most of the administrative tasks are carried out. If your development tool does not support integration of Team Coherence, Version Manager can be used directly to maintain archived files.

To access Version Manager, use the link under the Windows **Start** menu or, if your development tool supports it, select **Workgroups/Team Coherence** from the IDE menu:



The main Version Manager window is split into five areas. The actual contents of these areas may vary based on the current selection.

## 1) Main Menu / Toolbar

The main menu and toolbar give full access to the features of Team Coherence. Some menu options are dependent on the current selection, and the *Action* menu is also available as a context menu when right-clicking in some of the other windows.

## 2) Folder Pane

The Folder Pane is used to display the Projects and Folders contained in a Repository. In addition, it gives access to the Locked Files object and, if you are an Admin user, access to the Recycle Bin. Right-clicking on any object in this pane will display a context menu based on the current selection.

Selecting an object in this pane will cause the Files Pane to update based on this selection.

## 3) Files Pane

The Files pane displays the list of objects directly contained by the object selected in the Folder Pane. Depending on the current selection, this is normally a list of the files and folders contained within this object. Right-clicking on any object in this pane will display a context menu based on the current selection.

Selecting an object, or multiple objects, in this pane will cause the Detail and Information panes to update based on this selection.

#### **4) Detail Pane**

The detail pane displays details for the currently selected object in the Folder or File pane, depending on which is currently active. It contains a number of pages based on the current selection and may contain pages created by addin applications. Dependent on the selection, the pages include:

- ? **Notes** - The notes page allows HTML or notes to be attached to the selected object.
- ? **Revisions** - If the current selection is a File, the Revisions contained by that file will be listed.
- ? **New Files** - If the current selection is a Folder, this page will list any files or folders in the local directory associated with that folder that have not yet been archived.
- ? **Issues** - If you have installed the TC Tracker addin, any issues associated with the currently selected files will be displayed in this pane.

#### **5) Information Pane**

The contents of the Information Pane are based on the current selection in any of the other panes. It gives more details about the currently selected object(s) and also, dependent on the object, provides shortcut links to some of the most common Version Control actions.

## Common Activities

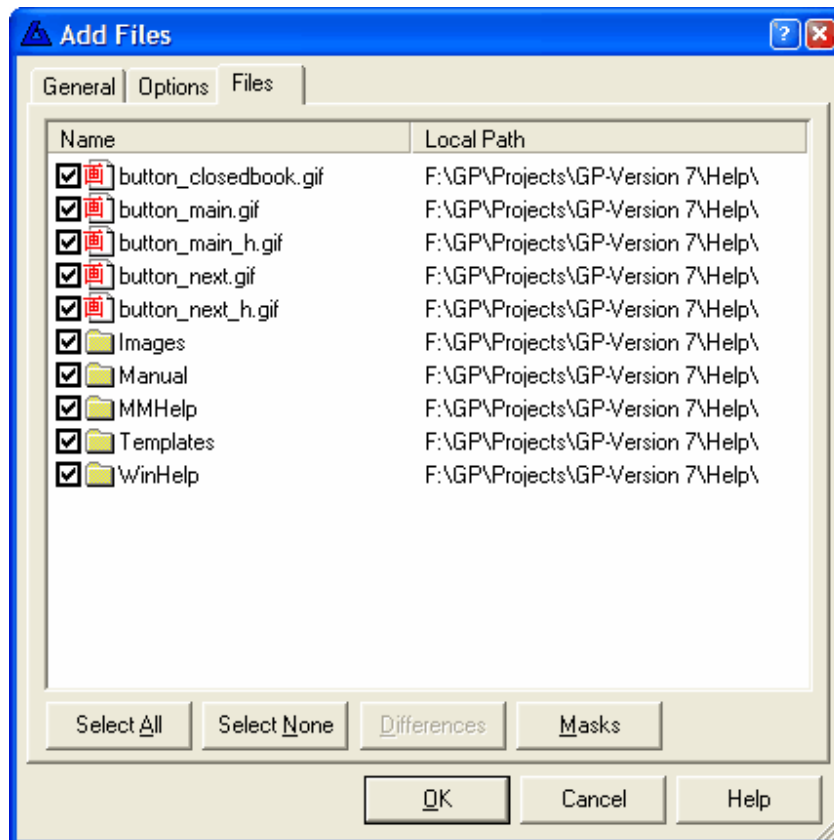
This section will describe how common activities are carried out. We will focus on Version Manager itself as IDE integration differs between products depending how the tool implements it.

## Adding Files

Before files can be managed by Team Coherence, they have to be added to the repository. By far the easiest way of doing this is to use the IDE integration functionality of the development tool you are using, however several other methods are available:

### Drag / Drop from Windows Explorer

Using this method, you can simply drag files and folders from Windows Explorer into existing Projects and Folders displayed in Team Coherence Version Manager. When you do this, the standard check-in dialog box will be displayed:



When you press the OK button, the selected files and folders will be added to the Team Coherence repository.

Team Coherence will automatically create any required folders based on the underlying disk structure and place the files in the relevant folders.

## Manually adding files to existing Folders

If you already have an existing project archived, it is sometimes necessary to add new files as they are created.

If you select a Folder in Version Manager, right clicking on the folder will allow you to add files by selecting **Add files...**

This will display the File-open dialog, with the Folders current local directory selected. Simply select the files you want to add and press the **OK** button.

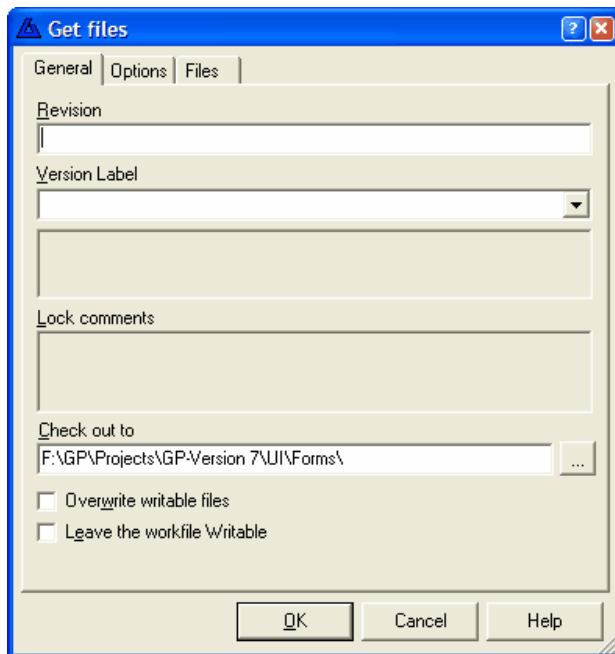
Note that a file cannot be archived more than once. In all of the above cases, if a file that you are adding is already archived, it will simply be checked-in if it is currently checked-out, or ignored.

Alternatively, when a Folder is selected in Version Manager, a page in the detail pane lists the files and folders for the currently selected directory that are not already archived. To add any of these files or folders: select them, right-click to display the context menu, and select **Add to Version Control**.

## Getting Files

Performing a Get on files is the process of getting a local read-only copy of the archived files. This could be to make sure that you are working with the most up to date versions of these files or to get a local copy of a past version of a project.

To perform a Get, select the relevant object(s) in Version Manager and select **Action/Get...** from the main menu. The Get Files dialog box will be displayed:



The Get Files dialog box is the same as the one used during the Check-out process, but only options relevant to doing a get will be enabled.

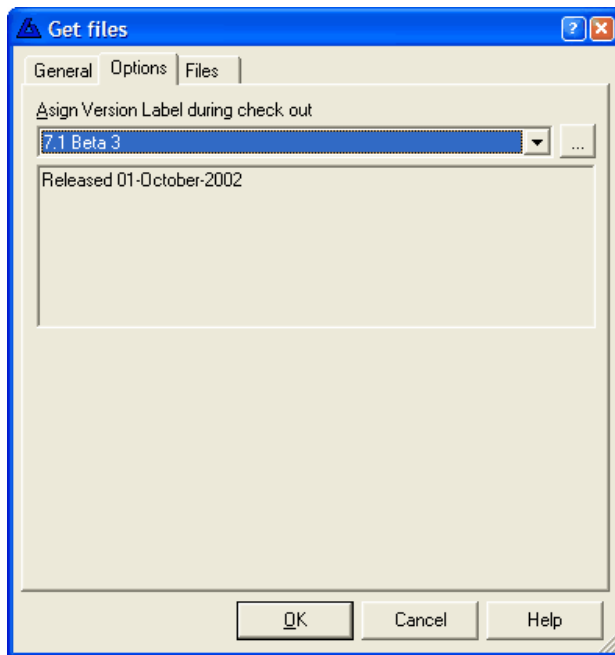
**Revision** – Enter a specific Revision of the files to refresh. Generally, unless you were getting a single file, it would not make sense to specify a revision.

**Version** – Select an existing Version to get that particular Version of the files. See also Views.

**Check out to** – Enter the root directory to get these files to. If this option is disabled, it normally means that the files you are getting do not have a common root directory.

**Overwrite writable files** – If this option is checked, any local files that are writable will be overwritten with the read-only copy being refreshed. If you check this option you should be aware that, when getting files, Team Coherence does not check the lock status of the files.

**Leave the workfile Writable** - If this option is checked, the local file will be made writable after the get has been performed. This option is not recommended since the local file can then be modified without checking the file out first.



**Assign Version during check out** – During the process of getting files, you have the option of assigning a new Version Label to the revision of each file that is refreshed.

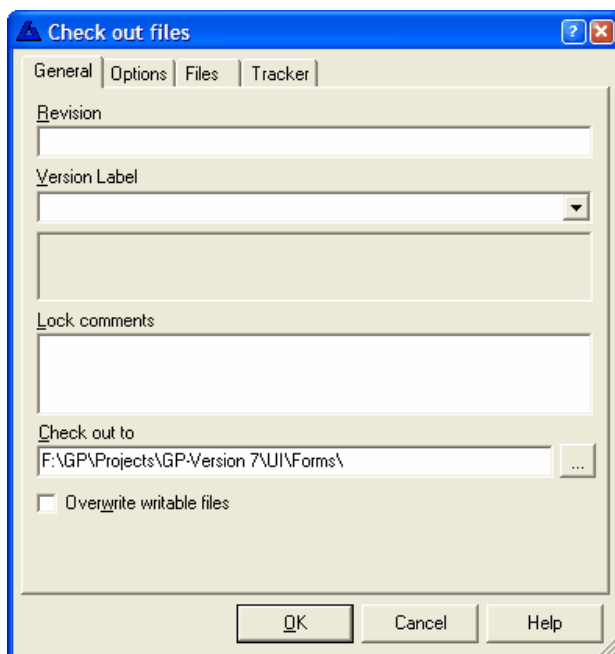
When you press the **OK** button, the selected files will be refreshed. The actual revision of the files that is refreshed will be based on the criteria you entered in this dialog.

## Check-out files to be modified

Before you can modify a file that has been archived by Team Coherence, you need to check out that file. Checking out a file makes the local copy of the file writable so that you can modify it and also places a lock on the archive. A locked archive does not necessarily stop other developers from modifying the file. Multiple locks can exist for a file if the server has been configured to allow this.

To check out a file from within your development tool (depending on the interface), make that file the current one and select **Workgroups / Checkout** from the main menu. To check out file(s), folder(s), or a project from within Version Manager, select the relevant objects and right-click, or select **Action**, and select **Checkout**.

You will be presented with the check out dialog box to allow you to select the revision or version of the files you are checking out and to specify a location for the files:



**Revision** – Enter a specific Revision of the files to check out. Generally, unless you were checking out a single file, it would not make sense to specify a revision.

**Version** – Select an existing Version Label to check out that particular Version of the files.

**Lock Comments** - Enter some text that describes what you are about to do with these files. This enables other users to see the reason why you have checked out these files. In addition, these comments can be used when checking in the files. Right-clicking in this box allows you to retrieve previously entered comments.

**Check out to** – Enter the root directory to check out these files to. If this option is disabled, it normally means that the files you are checking out do not have a common root directory.

**Overwrite writable files** – If this option is checked, any local files that are writable will be overwritten with the copy being checked out. If the check out process encounters any writable files, and you already have them locked, they will not be overwritten.

When you press the **OK** button, the selected files will be checked out. The actual revision of the files that is checked out will be based on the criteria you entered in this dialog.

## Check-in Modified Files

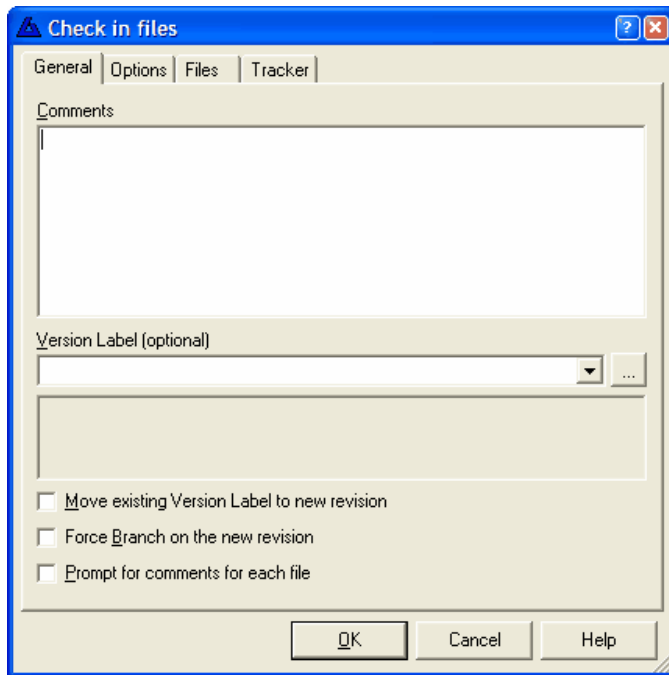
Once you have finished modifying files, it is important to check them back in to allow other users to access your changes.

Checking in a file creates a new revision in the archive if the file has been modified, and releases the lock on the file. By default it will also make the local copy of the file read-only so that it can't be modified.

- Because other developers may use the files you check in, it is important to make sure that they at least compile before checking them in.

To check in a file from within your development tool (depending on the interface), make that file the current one and select **Workgroups / Checkin** from the main menu. To check in file(s), folder(s), or a project from within Version Manager, select the relevant objects and right-click, or select **Action**, and select **Checkin**.

You will be presented with the check in dialog box to allow you to assign a Version Label to the files and to specify what to do once the files are checked in:



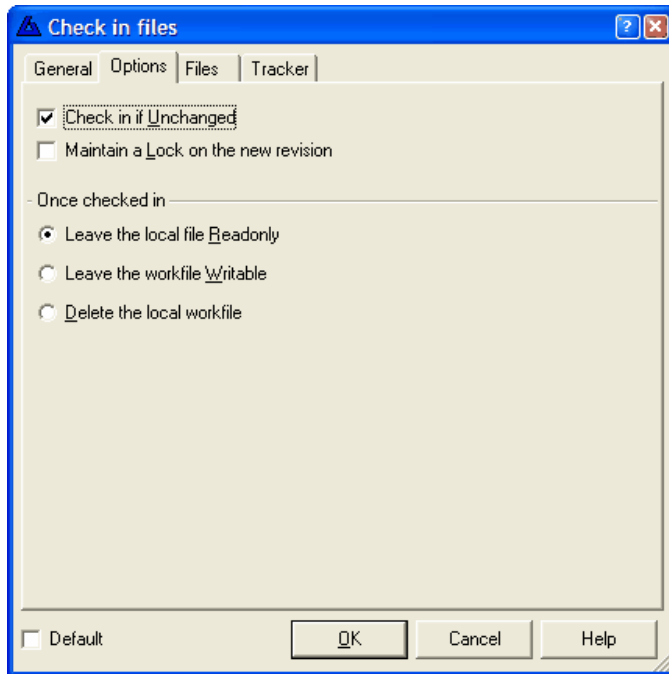
**Comments** - Enter comments to describe the changes made to the files being checked in. These comments will be applied to all files being checked in. Right-clicking in this box allows you to retrieve previously entered comments.

**Version Label** – If you want to attach a Version Label to the new revision of the file once it is checked in, select a Version Label using the ... button. The Version selection dialog box allows you to create a new version label.

**Move existing version label to new revision** – If you wish to move an existing Version Label to the new revision, check this box. If the Version Label you are assigning to the file already exists for a file, it will be removed from its existing revision and attached to the revision created when the file is checked in.

**Force Branch on the new revision** – If the file being checked in is not to become part of the main development stream, check this box. When the file is checked in, a branch revision will be created.

**Prompt for Comments for each file** - Checking this option, you can assign comments for each modified file being checked in. If this is left unchecked, the comment entered will be applied to all files being checked in.



**Checkin if Unchanged** - If this option is checked (default), all locked revisions will be unlocked whether or not there are any changes to the file. If this is unchecked, all unchanged files will remain locked.

**Maintain a Lock on the new revision** - If this option is checked, once the file is checked in, the new revision of the file will be locked again. This allows you to check in changes to allow others to get these changes while keeping a lock on the files to continue working on them.

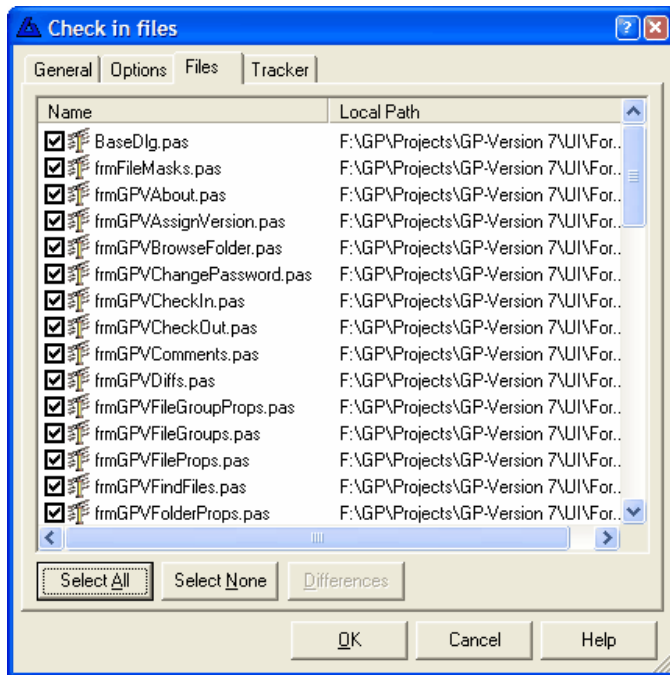
#### Once Checked in

**Leave the local file readonly** - Once the file is checked in, the local copy of the file will be made readonly to prevent accidental modification.

**Delete the workfile** – once checked in, the local workfile will be deleted.

**Make the file writable** – once checked in, the local workfile will be left writable, but the archive will *not* be locked.

**Default** – Check this box to make the current settings in the Options page the default settings for subsequent check in actions.



**Difference** - Switching to the Files page of this dialog allows you to check what changes were made to any of the files being checked in. Selecting a file and pressing the **Difference** button will execute the Difference Viewer against this file.

Pressing the **OK** button will check in the relevant files. The state of any local workfiles will be dependent on the options you select above.

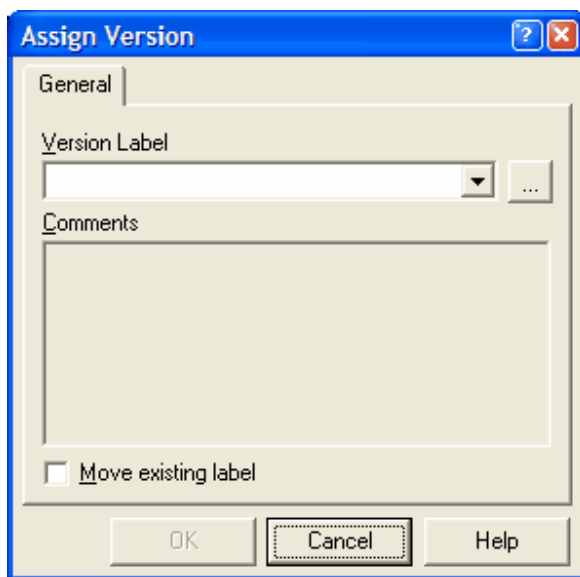
## Baseline a Project or Folder

As you develop your project you will reach certain stages where you want to 'remember' the state of the files involved in that project. For example, you would probably want to baseline a beta release of your software so that, if any bugs are reported, you know which revisions of each file were in that particular release.

Once all the files in your project have reached a stage where you want to do this make sure all files are checked in and select the project in Version Manager. Select **Action/Assign Version Label...** from the main menu, or right-click and select **Assign Version Label...**

You can also assign Version Labels during the Check-in, Check-out and Get processes.

Version Labels can be as descriptive as you like, but normally reflect the status of the project at that time. For example: Beta 1, RC1, etc. You can also add a description for the Version Label.



Drop down the **Version Label** list to select an existing label or press the ellipsis (...) button to create a new Version Label. The comments assigned to the selected label will be displayed in the **Comments** box.

If the label is already assigned to a revision in any of the files, and you want to move all instances of this label to the current tip revision, make sure and check the **Move existing label** checkbox.

When you press the OK button, the Version Label you entered will be added to the latest revision (i.e. the current revision) of all the selected Files based on the current **View**, if any.

It is important to remember that Version Labels are attached to a revision of a file as opposed to being related to the timestamp of the file at the time the version was created. This allows you to subsequently move the Version Label or add the same Version Label to a new file at a later date.

When checking a file out, or refreshing the local files, you can enter a Version Label in the check out dialog box to retrieve that particular version of the file. Version handling and Views will be discussed later in this discussion.

## Checking in your Locked files

If you check out files to modify them and don't check them in before going on to the next file, you can quickly lose track of just which files you have checked out. This is particularly important to other users when changes made to one file depend on changes to another file and only one of the files is checked in when the other users get the latest revision.

Obviously, Version Manager will display the locked files if you view the folder containing these files but, if the project has multiple folders, it can be tedious to locate and check-in individual files.

To help with remembering which files you and others have checked out, you can use the Locked Files view. This view lists, by user, all files that are currently checked out. This gives you a convenient method of checking in all your locked files.

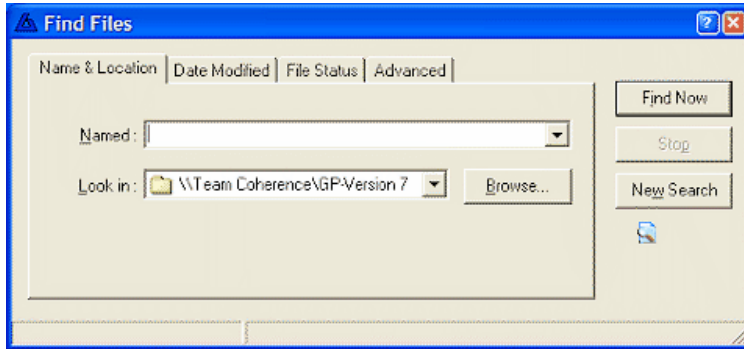


To check in all or some of the files you have checked out, select your username from the displayed list, select the files you want to check in, and select **Actions / Checkin...** from the main menu.

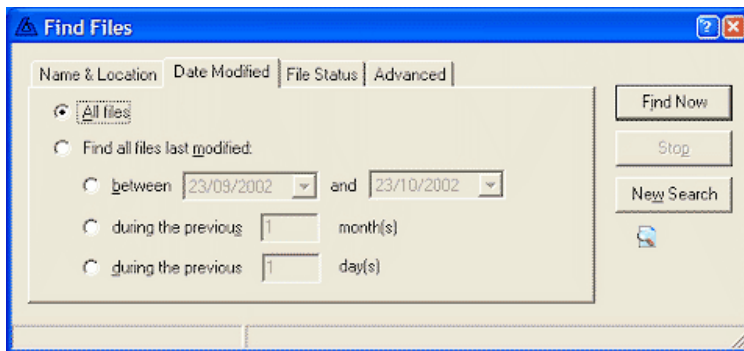
You can also use this view to check which files other users have currently got checked out.

## Finding Files

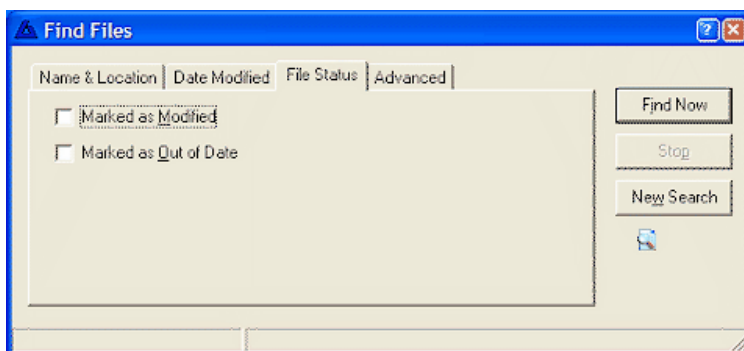
In large projects with deep folder hierarchies, it can be difficult to locate a file. For this reason, and to help with a number of tasks, Version Manager utilizes a complex search dialog. To display the find dialog, select **Edit/Search** from the main menu:



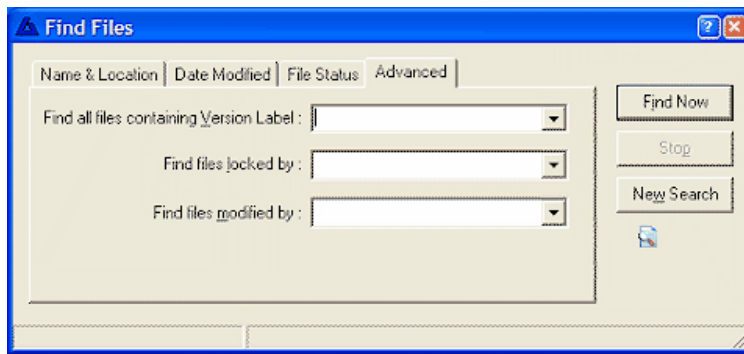
You can specify multiple criteria for your search and the results will be displayed in the search form. A typical search would be to locate all files with a particular extension, modified within a certain period:



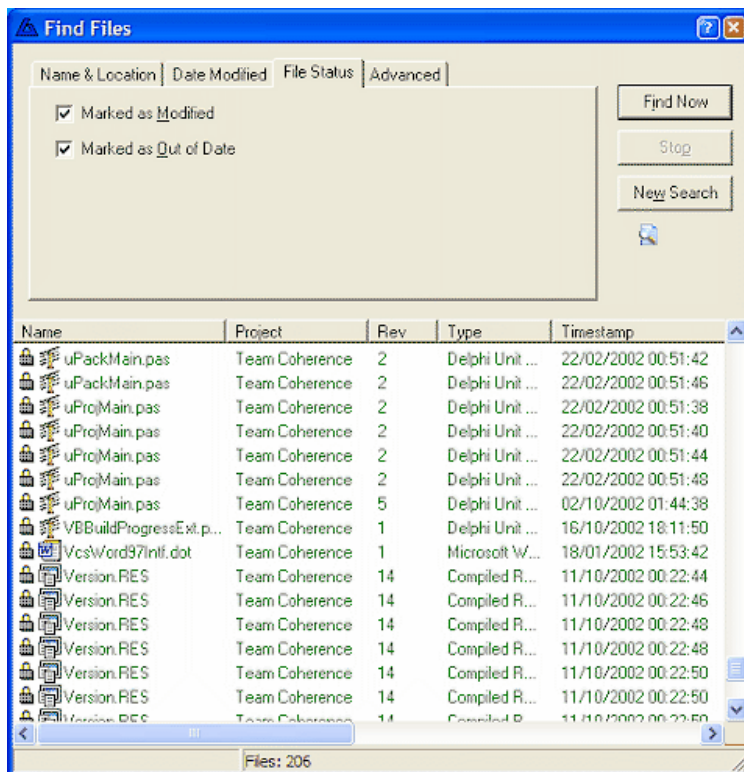
... currently modified on disk, or out of date files:



...and currently locked by a certain user:



When you press the **Find Now** button, the search results will be displayed in a pane below the criteria:



The result set can span multiple projects and folders and you have full access to the context menu for the selected files to allow you to check in and out files, view properties, add Version Labels and Promote.

## Using Views

A View is perhaps the most powerful tool available in Team Coherence when it comes to managing multiple versions of a project. At their simplest, Views allow you to filter which projects, files, and revisions are available in Version Manager.

Working with previous versions of a project then simply becomes a case of selecting a View. Once defined and selected, as far as the user is concerned they will be working on the latest version of the project. All branching, synchronizing, reassignment of Version Labels, etc are handled automatically by Team Coherence whether working through Version Manager, or your selected IDE interface.

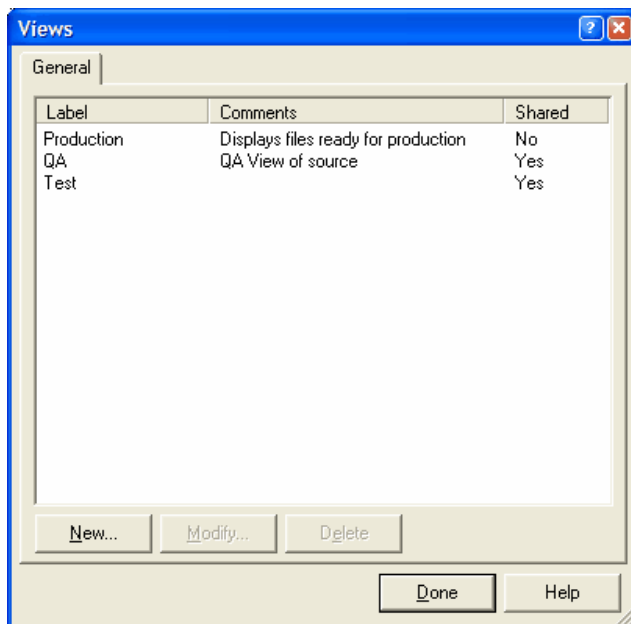
In addition, if required, the default working path for the files in any View can be different for each View, or can be based on the <default> View.

Views are accessible via the toolbar in Version Manager:



The drop-down list provides access to all user and shared views. It always contains at least one entry, which is the <default> view. The <default> view cannot be modified as it is the view that displays all projects and all files and represents the most current line of development.

To create or modify a view select the **Tools / Views...** option from the menu to display the View Manager dialog:



This dialog lists all the defined views for the current user. To modify a View select it in the list and press the **Modify...** button. To add a new View, press the **New** button.

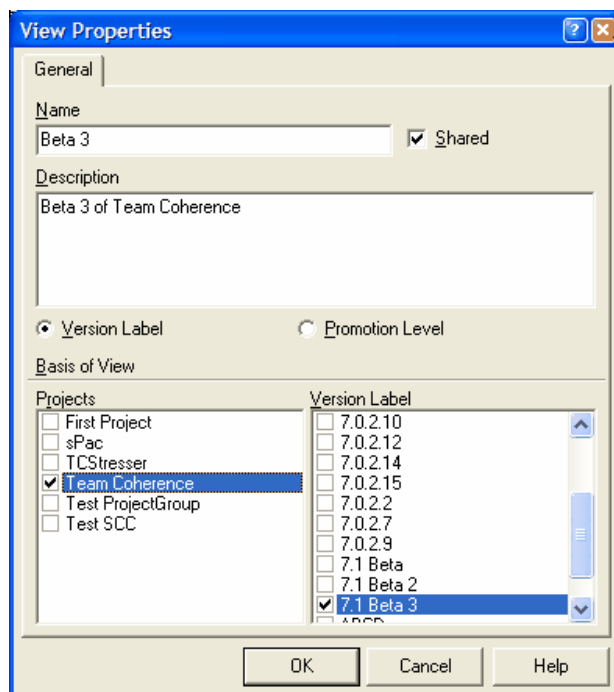
There are two types of View you can create:

1. **Version**
2. **Promotion**

Shared Views are available to all users connected to a repository, however shared views can only be created by Admin users.

## 1) Project/Version based Views

When defining a View based on a Project/Version filter, you are presented with a list of all Projects in the current repository and all Version Labels that have been assigned to these projects. Defining the filter is simply a case of checking the projects to be included and which version of these projects to view:



Note that, with Views based on a Version Label, there is no requirement to select a Version Label. In this case, the filter will default to the tip revision of files in the selected projects.

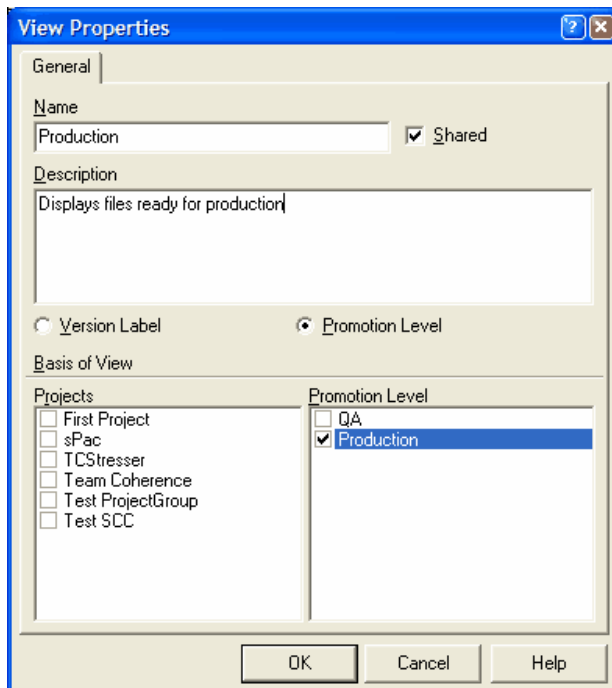
Making a Project/Version View the current view will force Team Coherence to carry out all actions on that particular version of the files in the selected projects. When viewing files in Version Manager, only files that match the View settings will be displayed, and only revisions up to and including that Version Label will be displayed.

Additionally, all folder working paths are specific to that View. Changing the default working path for a folder while a view is active changes it only for that View. In this way, you can work on different versions of a project simultaneously by keeping the working source for each version in a different location.

When a view is active, check out actions will automatically insert the Version Label that the view is based on in the Check Out dialog box. When checking in modified files, the Version Label will be added to the Check In dialog box, and the **Move existing version label to new revision** checkbox will be checked to make sure that the new revision is included in the view.

## 2) Promotion Level based Views

Views based on a Promotion Level are slightly different from that of a Project/Version view in that Promotion Level views are read-only. This basically means that, when these views are active, you cannot check out or check in files.



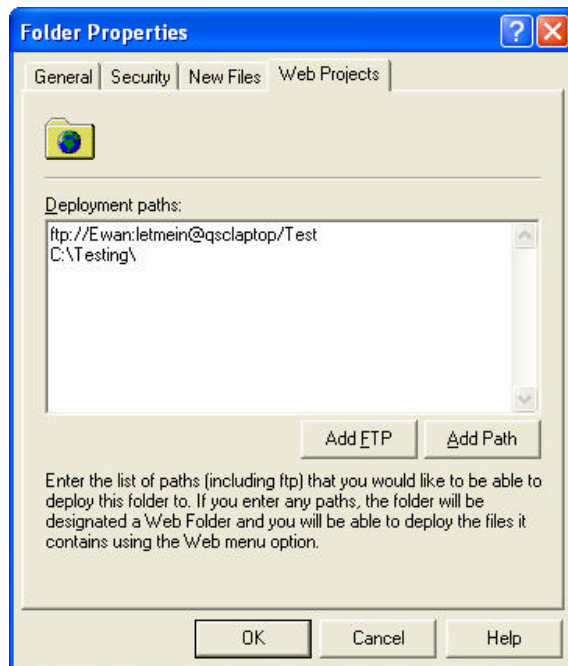
Promotional views are used to view all files at a particular level in the promotional hierarchy. Users can get a read-only copy of all the files at a particular promotional level but cannot modify them.

Note that not all promotion levels will be visible to all users. For example, developers normally do not require access to the QA or Production levels and similarly, QA do not normally require access to the Testing level.

## Deploying Files

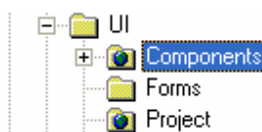
An addin developed for Team Coherence, installed as standard, allows the contents of Folders to be deployed via standard file-copy or FTP.

When you have completed development and testing of your Web pages, Admin users can use the **Web/Deploy** command to copy your project files to a live Web server. The location of this server is a folder or FTP site defined in the Folder Properties dialog box:



If you have defined a destination in the Web Projects page, you can deploy the files contained by the folder to as many sites as required. This particular page of the Folder Properties dialog is only visible to Admin users and only Admin users can physically deploy the files contained by the folder.

If a folder is designated as a Web Folder, it appears differently in Version Manager:



When selected, the Web/Deploy main menu action is enabled in Version Manager as is a shortcut in the Information Pane:

## Components

---


This object is a **Web Folder**. Click on the link below to Deploy the contents of the folder:

[Deploy this Folder](#)

Select an item to view its description.

Selecting Web/Deploy from the main menu will cause the latest version of the files to be extracted from Team Coherence which are then copied to the specified locations. The copy process will only update the server files if they are deemed as changed.

If any of the files are currently locked (i.e. the repository version is not necessarily the latest), the user will be prompted to confirm that they are sure they want to continue.

-  Rollback is not currently supported (7.1.2.55) and only files with a later timestamp are deployed, but this will be resolved in the next release.