

# **ComponentSoftware RCS Version 2.0**

## **User's Guide**



# Table of Contents

<b>Welcome to ComponentSoftware RCS.....</b>	
Why Use an RCS?.....	
ComponentSoftware RCS is Better than the Best.....	
How ComponentSoftware RCS Helps You.....	
<b>Quick-Start Tutorial.....</b>	
Basic Concepts.....	
Set Yourself Up.....	
Add a File to the ComponentSoftware RCS Repository.....	
View the Changes to a File.....	
Check-out the Latest Revision of a File.....	
Check-In a New Revision.....	
View the Revision History of a File.....	
View the Global Picture.....	
Thank You.....	
<b>Efficiency Tips.....</b>	
Invoking the Context Menu.....	
Working with Multiple Files.....	
ComponentSoftware RCS Properties.....	
CSDiff File Comparison Utility.....	
ComponentSoftware RCS Add-ons.....	
<b>Working with Projects.....</b>	
Introduction.....	
The Project Library.....	
Create a New Project.....	
Project-level Operations.....	

Project Reports.....	
Project Milestones.....	
Introduction.....	
The Milestone Library.....	
Creating a New Milestone.....	
Milestone Reports.....	
Milestone Retrieval.....	
Advanced Topics.....	
Local Workstation Settings.....	
Delete and Rename Commands.....	

<b>Workgroup Environment Setup.....</b>	
The RCS Repository Server.....	
The RCS Workstation.....	
The Network.....	
Workgroup Environment Tutorial.....	

<b>Command-line Interface.....</b>	
Basic Command-line Interface.....	
Advanced Command-line Interface.....	
Create command.....	
CheckOut command.....	
Update command.....	
CheckIn command.....	
Mark command.....	
Retrieve command.....	
Scan command.....	
Status command.....	
Report command.....	
History command.....	
Help command.....	

# Welcome to ComponentSoftware RCS

ComponentSoftware RCS is a powerful and robust document revision control solution. ComponentSoftware RCS monitors changes made in files that are accessed by standalone or networked workstations. Based on the widely used GNU RCS, it is fully integrated with Windows 95 and Windows NT. ComponentSoftware RCS supports multi-platform workgroups, making it the ideal solution for sites that share common files on UNIX and Windows platforms.

## ***Why Use an RCS?***

A Revision Control System (RCS) helps you manage multiple revisions of files by keeping a complete history of the changes performed to the files. This allows you to see how and when a file was changed, or to quickly return to a previous revision of a file. This can be crucial for files that are edited frequently such as programs, word-processor documents, and HTML documents.

## ***ComponentSoftware RCS is Better than the Best***

ComponentSoftware RCS is based on revision 5.7 of GNU RCS. GNU RCS was developed by the GNU project and is currently used by thousands of users, mainly in the UNIX environment. GNU RCS has an old-fashion command-line user interface, but ComponentSoftware RCS has a modern user-friendly graphical user interface that is fully integrated with the Windows Explorer and shell. This combination of tested power and ease-of-use brings full RCS mastery to your fingertips.

## **How ComponentSoftware RCS Helps You**

ComponentSoftware RCS allows you to:

- ***Retrieve any revision by any criteria anytime.*** Each revision is marked with a unique revision number, revision date, author name, optional symbolic name, and descriptive comment.
- ***Conveniently see what was changed between any two revisions.*** You can study the evolution of a function throughout its lifetime.
- ***Never lose any work because you can always backtrack.*** This is critical when there is a deadline and the last revision worked better.
- ***Work with binary files.*** You can maintain revisions of binary files such as MS-Word documents, pictures, drawings, etc..
- ***Save disk space.*** Since only the differences between revisions are saved, the archive file size is kept to the minimum size.

In a workgroup environment, ComponentSoftware RCS additionally allows you to:

- ***Know who does what and when.*** This simplifies communication between users, and smoothes out the whole development process.
- ***Avoid two users modifying the same file at the same time.*** ComponentSoftware RCS maintains a “lock” for each file.
  - ***Know when there is an updated revision of a shared document.*** ComponentSoftware RCS notifies you whenever a coworker checks-in a new revision.
- ***Use variety of servers and networks.*** Any file server can be used to keep the RCS archive files. Network connection can be LAN, corporate wide-area network, dial-up connection and the **Internet**.

ComponentSoftware RCS was designed to handle UNIX machines and Windows machines using the same RCS archive repository. In a multi-platform environment, ComponentSoftware RCS additionally allows:

- ***Users on UNIX and Windows machines to share common files.*** This option increases reusability and stability of products that target both UNIX and Windows platforms.
  - ***Transparently convert text files UNIX-to-DOS and DOS-to-UNIX.*** ComponentSoftware RCS takes care of the line-break sequence that is different for UNIX and Windows/DOS files.
-

# Quick-Start Tutorial

This brief tutorial shows you how to perform the basic tasks you need to use ComponentSoftware RCS. We recommend you follow this tutorial through from start to finish. The tutorial takes about 10 minutes.

## ***Basic Concepts***

The ComponentSoftware RCS Repository keeps track of the various revisions of your files. Each file you want to keep track of must be added to the Repository. Whenever a file receives a significant revision, it should be added again to the Repository so that anytime in the future you can go back to that revision of the file. Adding a file again to the repository is called *checking-in* a file.

When you want to edit a file, you should take the latest revision of the file out of the Repository. This way you can make sure you have the file in its correct form. This is called *checking-out* the file.

## ***Set Yourself Up***

Before we begin, prepare a sample file:

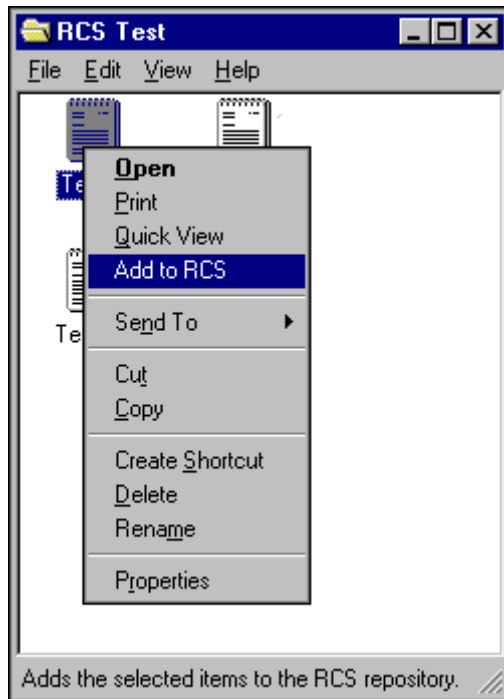
1. Create a new text file named TEST.
2. Open Notepad and type a few lines.
3. Save the file.

## **Add a File to the ComponentSoftware RCS Repository**

To have ComponentSoftware RCS keep track of your file, you must add the file to the Repository. Add TEST.TXT to the Repository:

1. View TEST.TXT in Windows Explorer.
2. Click on the file with the right mouse button to open the context menu, and choose **Add to RCS**.

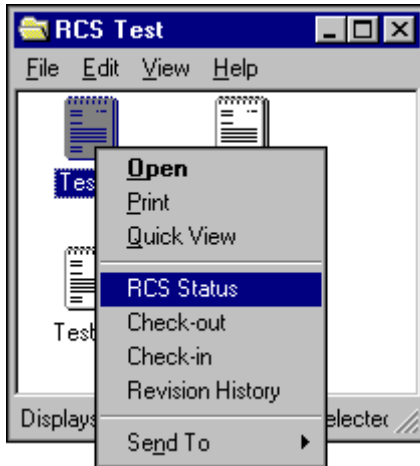
The current revision of TEST.TXT is added to the Repository.



## View the Changes to a File

You can check whether a file has been changed since it was last added to the ComponentSoftware RCS Repository. If the file has been changed, you can see the changes. Check whether TEST.TXT has changed since you last added it to the Repository:

3. View TEST.TXT in Windows Explorer.
4. Click on the file with the right mouse button to open the context menu, and choose **RCS Status**.



The following message appears:



The message indicates that TEST.TXT has not been changed since you last added it to the Repository.

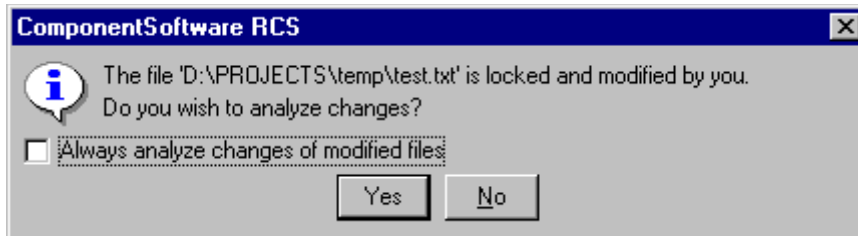
**Note:** As long as TEST.TXT is unchanged, the *Check-out* and *Check-in* menu entries are disabled. (If you are in a workgroup environment, the *Check-out* menu entry is enabled for unlocking the document.)



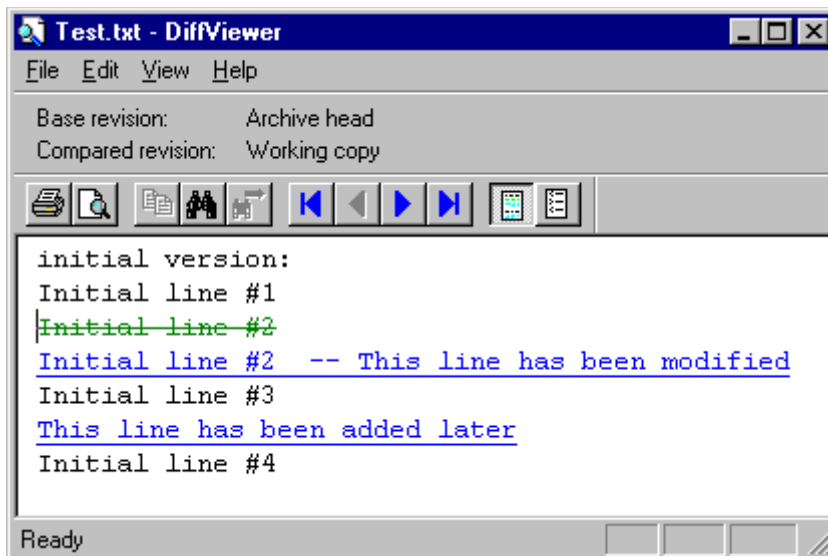
Change TEST.TXT and then check again whether TEST.TXT has changed since you last added it to the Repository:

5. Using Notepad, add a few lines to TEST.TXT.
6. Invoke **RCS Status** again.

The following message appears:



The message indicates that TEST.TXT has been changed since you last added it to the Repository. If you click **Yes**, a list of the differences between the last version you added to the Repository and the current file appears.

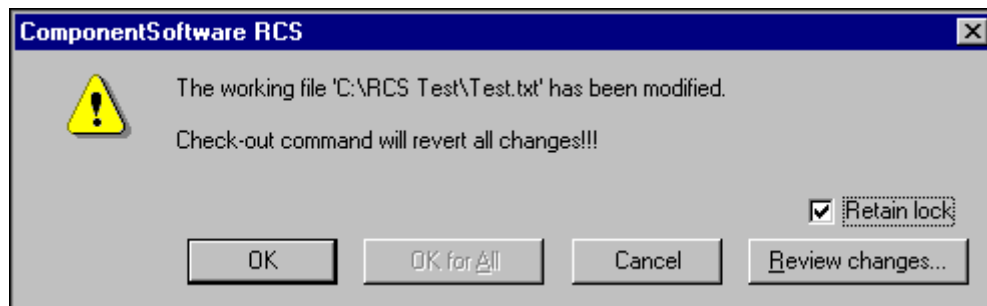


## Check-out the Latest Revision of a File

At any time, you can undo all changes you have made since you last added a file to the Repository. This is called *checking-out* a file, and effectively returns the file to how it was when it was last added to the repository. Check-out TEST.TXT:

7. View TEST.TXT in Windows Explorer.
8. Click on the file with the right mouse button to open the context menu, and choose Check-out.
9. Select OK.

TEST.TXT returns to its original revision.

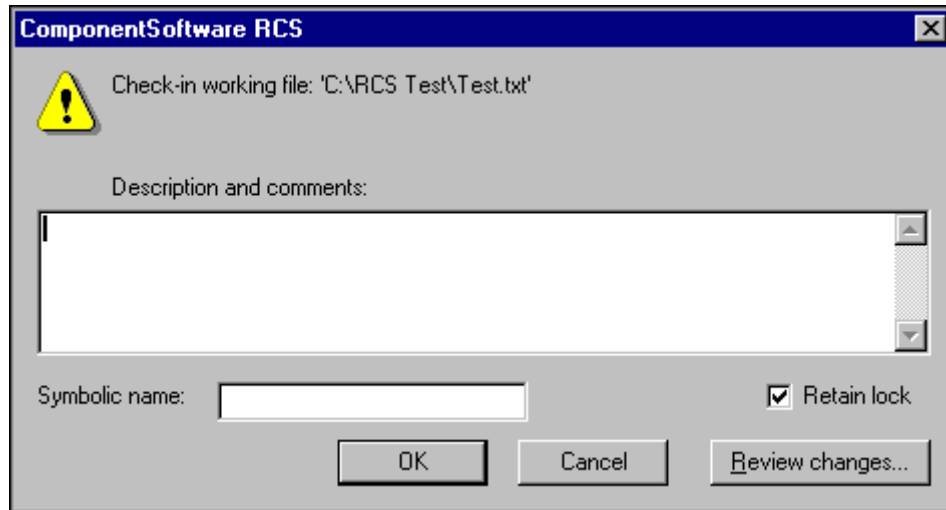


When the Check-out dialog box is open, you can see what was changed since the last revision by pressing **Review Changes**. If you are in a workgroup environment, when you check-in or check-out a file, nobody else can check-in or check-out the same file. If you want other users to be able to check-in or check-out the file, uncheck the **Retain Lock** option.

## Check-In a New Revision

As mentioned above, whenever a file has been significantly edited, it should be reentered into the Repository, or *checked-in*. Edit TEST.TXT and check it in:

10. Use Notepad to edit TEST.TXT.
11. View TEST.TXT in Windows Explorer.
12. Click on the file with the right mouse button to open the context menu, and choose Check-in.
13. Fill in the Description and comments. Use a symbolic name to emphasize major revisions.
14. Press OK.



When the Check-in dialog box is open, you can see what was changed since the last revision by pressing **Review Changes**. If you are in a workgroup environment, when you check-in or check-out a file, nobody else can check-in or check-out the same file. If you want other users to be able to check-in or check-out the file, uncheck the **Retain Lock** option.

***Tip:** When checking in a new revision, press Review Changes and copy and paste sections into the Description and Comments field. This will help you keep track of the changes in that version.*

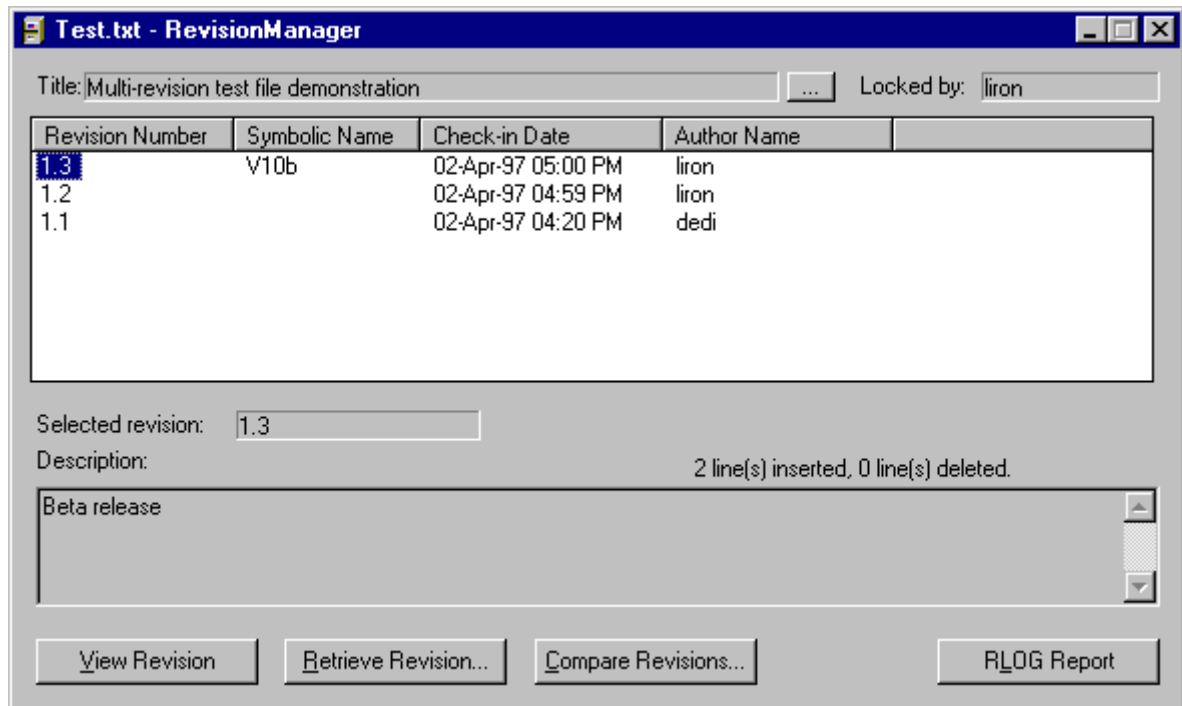
## View the Revision History of a File

At any point, you can view a list of the revisions of a file. You can then compare the revisions to one another, and revert to an earlier revision. View the revisions of TEST.TXT:

15. View TEST.TXT in Windows Explorer.

16. Click on the file with the right mouse button to open the context menu, and choose **Revision History**.

The various revisions of TEST.TXT are displayed. You can sort the list as you wish by clicking on the desired column header. Click again on the same column header to reverse the sort order.

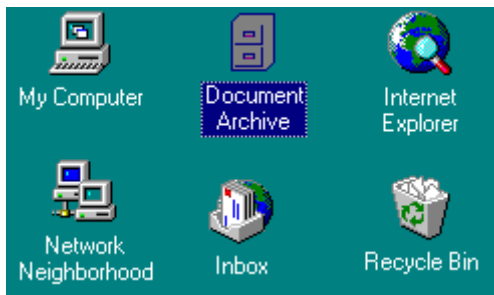


This dialog box gives you many options:

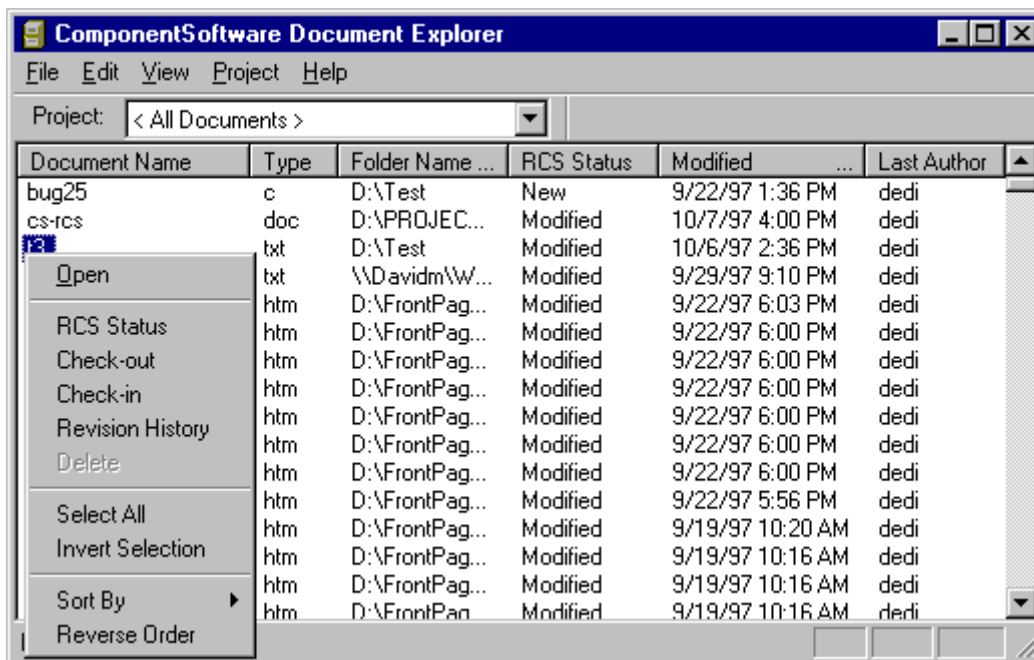
- Select any revision on the list to view the revision's description and comments.
- Click on any revision with the right mouse button to open the context menu and apply revision related commands.
- Press **View Revision** to view the selected revision.
- Press **Retrieve Revision** to save the selected revision to disk.
- Press **Compare Revisions** to view the changes done between the selected revision to any other revision.
- Press **RLOG Report** to view a detailed change report for the file.
- Press '...' to set the document title and description.

## View the Global Picture

ComponentSoftware RCS adds a special icon to your desktop called the *Document Archive*.



At any point, you may activate the *Document Explorer* by double-click on the *Document Archive* icon. Choose *Project* and then *Scan New Files* to scan the RCS repository for new files.



You can see the major properties of your document:

- Document name, type and Folder
- Document archive status.
- The last time the document has been checked-in or modified
  - The last author of the document.

You can sort the list as you wish by clicking on the desired column header. Click again on the same column header to reverse the sort order. You may click on a document with the right mouse button to open the context menu and apply any RCS command.

The Document Status property gives you the exact state of the document:

- *Locked* – You may change the document. (In a workgroup environment, nobody else may change it.)
- *Modified* – You have changed the document. It is recommended to check-in the document whenever it receives a significant revision.

In a workgroup environment, a document may have the following states as well:

- *Archived* – The document is archived. Nobody may change it.
- *Blocked By...* – A partner had locked the document. Nobody else may change it till the lock is released.
- *New* – The document exists in the archive but not in your working directory. You may check-out the document in order to get the current revision.
  - *Updated* – The document has been updated in the archive. You should check-out the document in order to get the current revision.

**Note:** Project-level commands are discussed later on this manual.

## ***Thank You***

Now you have the basic skills necessary to take advantage of the power of ComponentSoftware RCS. Thank you for participating in this tutorial.

# Efficiency Tips

## ***Invoking the Context Menu***

The context menu can be invoked in several ways:

- Click on the file with the right mouse button.
- Select the file, and then open the Explorer *File* menu.
- If you have a Windows 95 keyboard, select the file and then press the key near the right Ctrl key.

*Tip:* You can invoke the context menu from the Windows *File Open* dialog box.

## ***Working with Multiple Files***

Sometimes, it is convenient to perform an operation on several files at once. For instance, if you have edited several files related to the same product release, they can all be checked-in at one time. This gives all the files the same check-in information.

To perform a multiple file operation:

17. Select all the related files or folders.

18. Invoke the context menu and choose the desired option.

19. *Examples:*

20. Right-click on a working folder and choose *RCS Status* to evaluate the status of all files within a working folder tree.

21. Use the Windows Explorer *Find* command to get a list of files that match certain criteria. (For instance, all include files in a project.) Select the desired files within the list and apply any RCS command.

## ***ComponentSoftware RCS Properties***

Although ComponentSoftware RCS has been designed for quick start operation, many of the ComponentSoftware RCS properties can be customized. To access the ComponentSoftware RCS Properties setup program either:

- Right click on the *Document Archive* icon on the desktop and select *Properties*.
- Invoke the Explorer *Start* menu, and select *Programs | CS-RCS | Properties*.

For full details, click *Help* to get a context sensitive on-line help.

## ***CSDiff File Comparison Utility***

CSDiff is a stand-alone folder and file comparison utility included with ComponentSoftware RCS. To activate CSDiff, create a desktop shortcut to the CSDiff.EXE program (located on the CS-RCS “System” directory).

- Double-click on the CSDiff icon to activate it with default parameters.
- Drag two files or a single file to the CSDiff icon to activate it with your selected files.
- To activate CSDiff from any external tool or DOS window, type:

<CSDiff folder>\CSDiff.exe "FilePath1" "FilePath2"

*Note:* CSDiff is available as a free stand-alone product. For full details, visit the CSDiff home page at: <http://www.ComponentSoftware.com/csdiff/>

## ***ComponentSoftware RCS Add-ons***

From time to time, ComponentSoftware and other companies will provide add-on packages for ComponentSoftware RCS. For instance, an add-on package named D3RCS enables the integration of ComponentSoftware RCS with Borland’s Delphi 2 and Delphi 3. Another add-on package, enables integration with Symantec Visual Café. For full details, visit ComponentSoftware RCS web site at: <http://www.ComponentSoftware.com/csrs/addons.htm>



# Working with Projects

## *Introduction*

ComponentSoftware RCS can be used to manage any number of documents. When working with a large number (over 50) of documents, it is recommended to group these files into projects. Working with projects you have the following benefits:

- It is convenient to inspect and perform RCS operations on a pre-defined set of related files.
- Users manipulating one project can not accidentally modify the state of documents assigned to another project.
- Projects can be defined as a sub-tree of the central repository or as an alternate repository tree. With alternate repository tree, you can use ComponentSoftware RCS to manage distributed RCS repositories in the corporate network or over the Internet.
- Projects can be assigned a per-workstation working directory root.

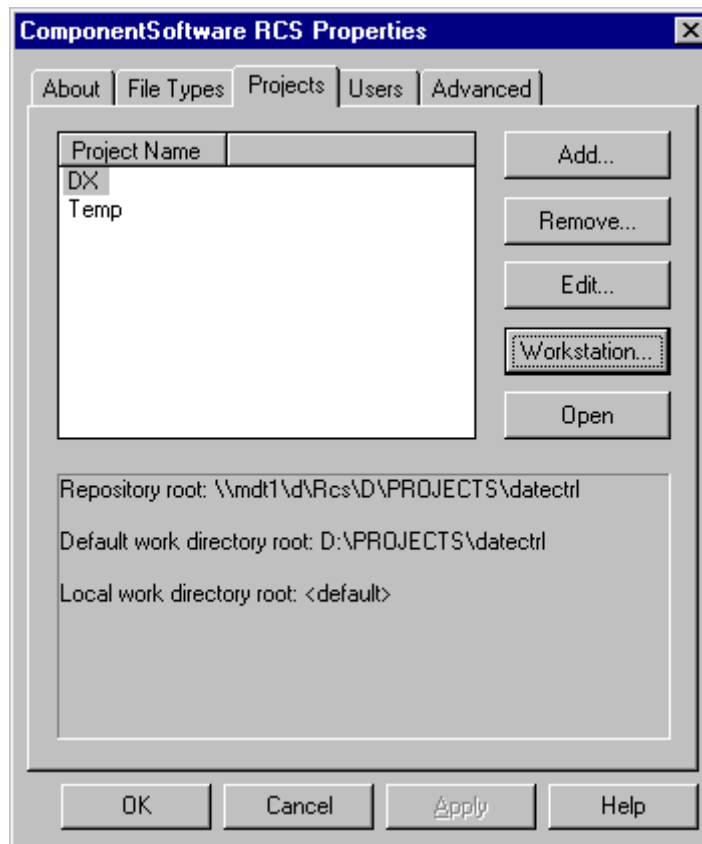
*Note:* ComponentSoftware RCS must be configured to use **Central Repository** in order to support projects. (Central repository is the setup program default.)

## The Project Library

To create a new project or edit an existing one, activate the ComponentSoftware RCS *Project Library* by either:

- Invoke the Explorer *Start* menu, and select *Programs | CS-RCS | Properties*.
- Right-click on the *Document Archive* icon on the desktop and select *Properties*.
  - Double-click on the *Document Archive* icon on the desktop to activate the *Document Explorer*. Choose *Project* and then choose *Settings*.

The *Projects* tab is used to create and edit projects.



This dialog box gives you the following options:

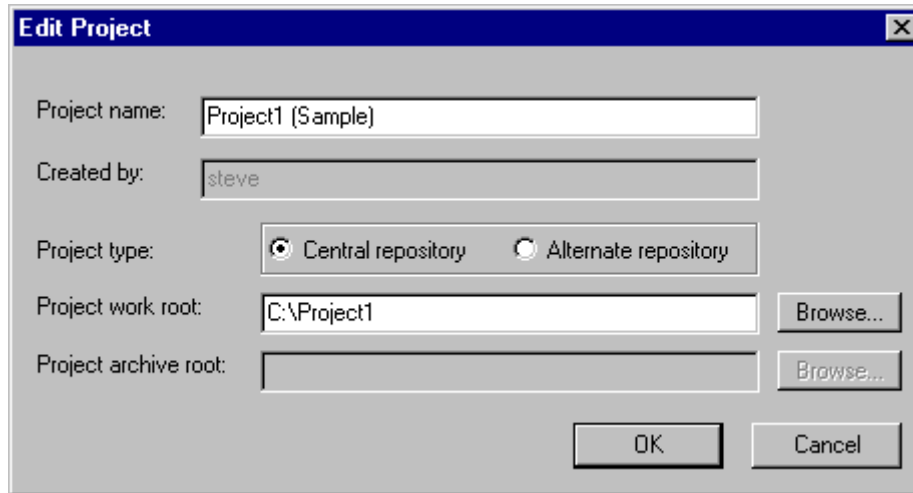
- Press **Add...** to define a new project.
- Press **Remove...** to remove the selected project.
- Press **Edit...** to edit the selected project. (Click on the project's label to rename a project.)
- Press **Workstation...** to define the local workstation working directory of the selected project and to create a desktop shortcut to the project.
- Press **Open...** to open the selected project.

## Create a New Project

To create a new project, please prepare few sample files:

1. Create a new folder named *C:\PROJECT1*.
2. Copy few sample files into the folder.
3. Add all sample files to the RCS repository. (Select all files and apply *Add to RCS*.)

Now, activate the *Project Library*. Press **Add...** to define a new project.



This dialog box includes the following properties:

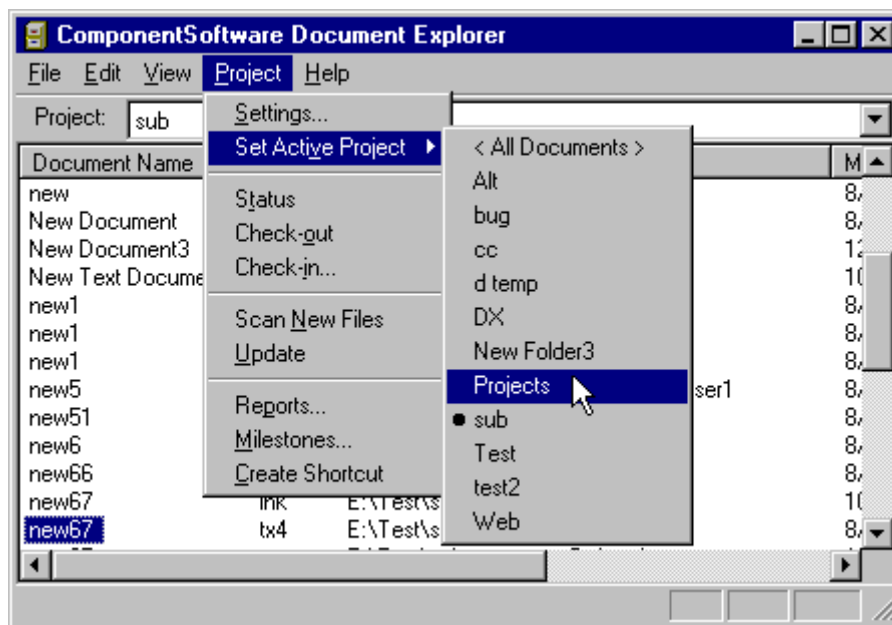
- *Project name* – Specify the desired project name.
- *Created by (read-only)* – Project creator name.
- *Project type* – Select *Central repository* to specify a sub-tree of the central repository. (*Alternate repository* is used to define an external RCS repository tree. This feature is usually used to access documents of other department or other company.)
- *Project work root* – Specify the work root of the project. Press the **Browse...** button. Select *C* and then *PROJECT1*.
  - *Project archive root* – This field is applicable to alternate repositories only. (*Central repository* projects are mapped to the central repository using the normal ComponentSoftware RCS mapping conventions.)

Press **OK** to return to the *Project Library*. Press **OK** again to save the new settings.

*Note:* Advanced users can define sub-projects using the same method. As an example, create a sub-folder named *C:\Project1\Doc* and map it to a new project named *PROJECT1 Documents*. Operations on the *PROJECT1 Documents* project will affect files within the *Doc* sub-folder. Operations on the *PROJECT1* project will affect all files within the *C:\PROJECT1* tree.

## Project-level Operations

Double-click on the *Document Archive* icon on the desktop to activate the *Document Explorer*.



- Choose *Settings...* to activate the ComponentSoftware RCS *Project Library Manager*.
  - Switch to any project by selecting it from the *Project* list box or from the *Set Active Project* menu. (**Tip:** Select *<All Documents>* to perform operations on all documents within the repository.)

Once you have selected a project, you can perform the following project-level commands:

- Choose *Status* to evaluate the status of documents of the current project.
- Choose *Check-out* to check-out all documents within the current project. Use this command to check-out the latest revision of the RCS repository into your working directory. On workgroup environment, this command is used to lock and unlock files as well. (**Note:** See the *Update* command below for an alternative way to update the working directory.)
- Choose *Check-in* to check-in all modified documents within the current project.
- Choose *Scan New Files* to scan the RCS repository for new documents of the current project.
- Choose *Update* to update your working directory to the latest revision of the RCS repository. (**Note:** Unlike the *Check-out* command, the *Update* command does not change the *lock* state of files and does not suggest to revert modified files.)
  - Choose *Create Shortcut* to create a *Windows Desktop* shortcut for the current project and its sub-projects.

*Note:* The *Reports* and *Milestones* commands are elaborated on the next sections.

## Project Reports

Choose the *Reports* command to generate a report of the current status of the active project.

This dialog box includes the following options:

- Specify the desired fields and order on the *Reports fields* table. (To add a field, select on the *Exclude fields* table and click *Add* or double-click on the field name.)
- Select the sort key on the *Sort by* table. Choose *Ascending* or *Descending* as desired.
- To report a range, specify an alphanumeric string on the *From* and/or *To* fields. The specified filter is used to determine the report range according to the report sort key.
- Choose *Text* for text reports. Text reports can be imported by any Windows spreadsheet, database or report-generator. Choose *HTML* for HTML reports. HTML reports can be published on the Web. Then, using any browser, they can be examined and printed.
- Choose *Preview* to preview your report.
- Choose *Report...* to generate a report to a file.
  - Choose *Close* to close this dialog.

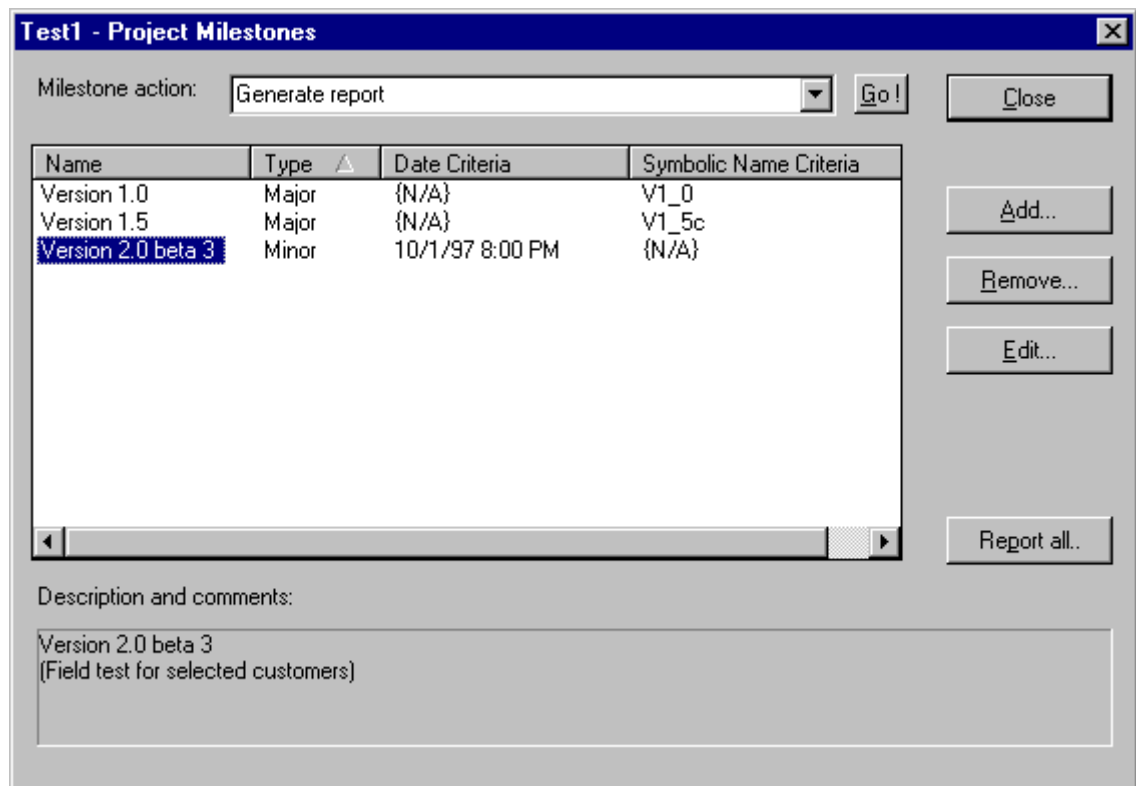
## Project Milestones

### Introduction

Whenever a project reaches a significant milestone (such as product release) it is recommended to mark the current status. This mark enables you to “freeze” the current state and to backtrack later if needed.

### The Milestone Library

Choose the *Milestones* command to get the project’s *Milestone Library*.



This dialog box gives you the following options:

- Press **Add...** to define a new milestone.
- Press **Remove...** to remove the selected milestone.
- Press **Edit...** to edit the selected milestone. (Click on the milestone’s label to rename it.)
- Press **Report all...** to generate a report for all defined milestones. (Refer to the *Project Reports* section for full details on the CS-RCS report generator.)
- Select an action and Press **Go...** to perform an action on the selected milestone.

## Creating a New Milestone

To define a new milestone, activate the *Milestone Library* and press **Add...**

This dialog box includes the following properties:

- Milestone name – Specify the desired Milestone name.
- Description and comments – Describe the reason for this milestone.
- Milestone criteria – Specify the time and date of this milestone. Alternatively, provide an RCS symbolic name that defines the milestone. (*Note:* The second method is to be used mainly with legacy RCS files.)
  - Milestone type – Select *Major* to indicate major releases. Select *Minor* to indicate intermediate releases. Select *Temporary* to indicate internal releases.

*Notes:*

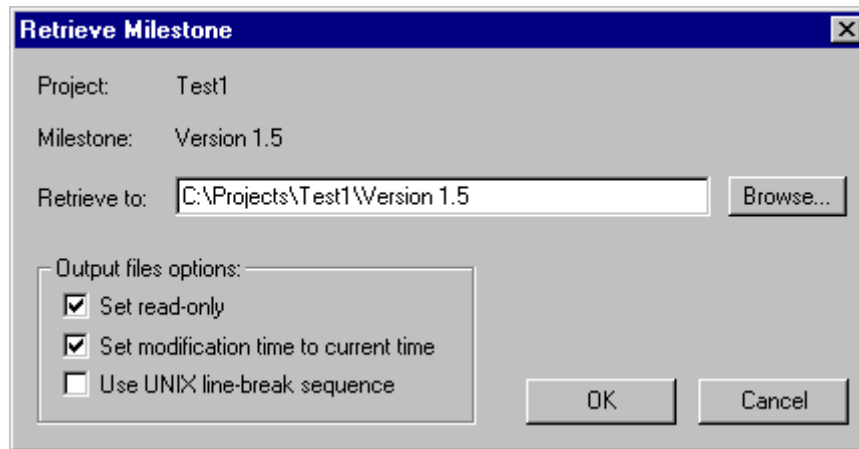
- To “freeze” the project’s current state, be sure none of the files is currently modified by any user. (That is, all files in the project are either *Archived* or *Locked*.)
- Advanced users might consider to mark each RCS file with an RCS symbolic name as well. For full details, refer to the *Mark* command of the CS-RCS command-line interface.

## Milestone Reports

To generate reports for a milestone, select the *Generate report* action and click on the *Go* button. This will activate the CS-RCS report generator. (Refer to the *Project Reports* section for full details on the CS-RCS report generator.)

## Milestone Retrieval

You may anytime retrieve a project revision specifying a milestone name. To do so, select the *Retrieve revision* action and click on the *Go* button.



This dialog box includes the following properties:

- Retrieve to – Specify the actual working tree root for this retrieve operation. (For instance, you may use this option to retrieve any revision to a production server.) Click the *Browse...* button to browse for the desired folder.
- Set read-only – Clear this option to produce output files with a read-write permission.
- Set modification time to current time – Clear this option to produce output files with modification time identical to the files' original check-in time. (This feature is useful is you wish to perform folder comparison. However, it should be used with care since it might confuse *make* utilities.)
- Use UNIX line-break sequence – By default, CS-RCS produce working files using the DOS/Windows line-break sequence conventions. Clear this option to produce output files with the UNIX line-break sequence conventions. (This feature is useful is you wish to use UNIX tools to process the output files.)

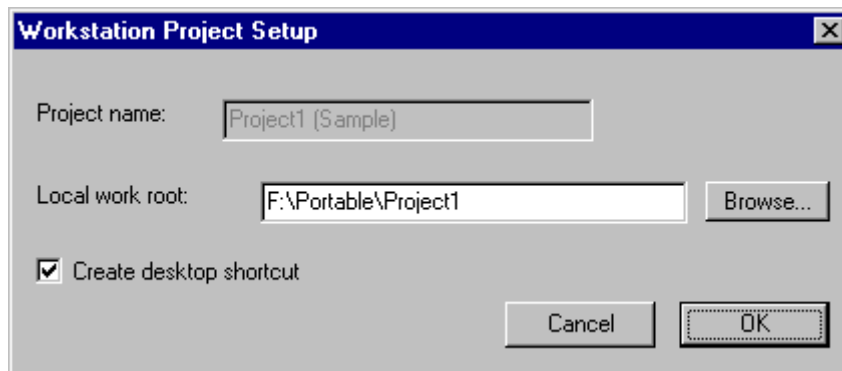


## Advanced Topics

### Local Workstation Settings

Sometimes, it is convenient to tune projects mapping for a specific workstation. To do so, activate the *Project Library* and select the desired project.

Press **Workstation...** to customize the local workstation properties.



This dialog box includes the following properties:

- Local work root – By default, all workstations have the same working directory structure as derived from the repository structure. Whenever desired, you may specify a local working directory root for the selected project. (For instance, you may want to map the working folders to a portable media.)
- Create desktop shortcut – Check this option to create a *Document Explorer* desktop shortcut for this project.

*Note:* Working-file to repository-file mapping is done using a best-fit method.

## **Delete and Rename Commands**

Sometimes, a new file added to the repository obsolete an entire old file. To delete RCS files, select the desired files in the *Document Explorer*, invoke the context menu and choose *Delete*. (Obsolete files are labeled with the *Deleted* status.)

Using the *Document Explorer*, you may perform the following operations on *Deleted* files:

- Trace the history and retrieve any revision of the file.
- Restore the file into a normal state.
- Permanently remove the file from the RCS repository.

**Tip:** Choose *View* and then *Deleted Documents* to toggle *Deleted* documents display.

Another common scenario is when working file's name and/or folder changes. To rename an RCS file, select the desired file in the *Document Explorer*, invoke the context menu and choose *Rename*. The *Rename* command marks the old RCS name as *Deleted* and creates a new RCS file with the desired name and location.

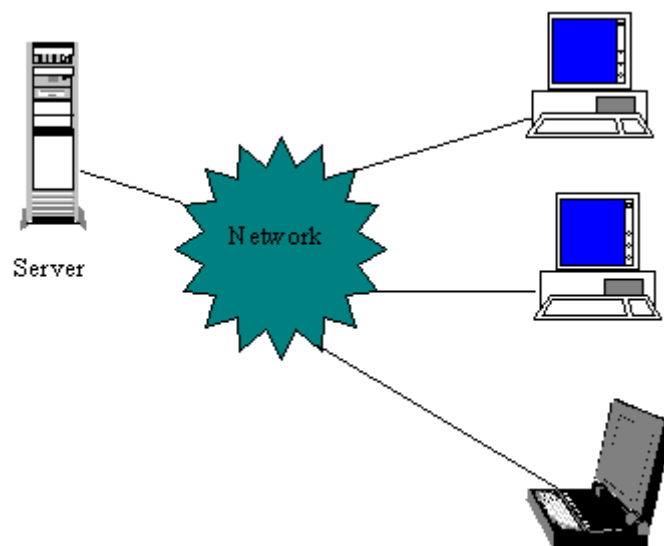
The implementation of the *Delete* and *Rename* commands enable to retrieve old project milestones containing deleted or renamed files. Since this method is very important for project-level revision management, you should be careful with permanent deletion of RCS files.

**Note:** You may not delete or rename files locked by other users.

# Workgroup Environment Setup

Workgroup environment consists of three components:

- The RCS repository server
- The workstations
- The network



## ***The RCS Repository Server***

The requirements from the RCS repository server are:

- All workstation must have read and write access to the server's drive.
  - The server must support long file names.

Popular servers that meet these requirements are:

- Windows 95 or Windows NT shared (i.e., “exported”) folder
- Windows NT Advanced Server
- NetWare Novell (*Note: long file names support should be installed.*)
- SMB server on a UNIX machine (such as public-domain SAMBA)
  - NFS server on a UNIX machine.

### ***Notes:***

- You may use the same machine as both server and workstation.
  - You may refer the shared drive as a normal drive letter (F:, for example) or as an UNC path (\\SERVER1\RCS-TREE\\, for example). The later is simpler and guaranties unique identification of the shared drive.

## **The RCS Workstation**

Any machine running Windows 95 or Windows NT can be used as an RCS workstation.

The following example illustrates a typical environment:

The RCS administrator creates the following document tree and adds the documents to the RCS repository.

```
C:\Project1\...  
  C:\Project1\Src  
  C:\Project1\Doc  
  C:\Project1\FAQ  
C:\Project2\...  
  C:\Project2\Src  
  C:\Project2\WWW
```

From now on, any workstations can perform the following operations:

- Retrieve a local copy (called *working copy*) of the current revision. A workstation may retrieve either Project1 or Project2 (or both).
- Retrieve a read-only copy (an *Archived* copy) or a read-and-write copy (a *Locked* copy). ComponentSoftware RCS grants one user only may change a document at a given time.
- Check-in a new revision. All other users are notified a new revision of the document is available.
- Add a new document to the repository. All other users are notified a new document is available.

### **Notes:**

- By default, the working folders have the same structure on all workstations. You can change this by customizing the *Workstation* properties of any project.
- If desired, the working folders may reside on a server drive that is private to each workstation. As an example: the working folder may be G:\Project1 while G: is mapped to \SERVER\User1 on one workstation and mapped to \SERVER\User2 on another workstation.

## ***The Network***

You may select the network that best meets your needs:

- **Local Area Network (LAN)** – Teams that are located in the same building should use a LAN to access the RCS repository.
- **Corporate network** – Coworkers that are located on several sites of the same organization should use the corporate network (LAN/WAN) to access the RCS repository.
- **Dial-up connection (RAS)** – Mobile workers can use dial-up lines to access the RCS repository server.
- **The Internet** – Coworkers located anywhere in the country or around the globe can use the Internet to access the RCS repository. While the workstations may have local dial-up connection to the Internet, the server must have a permanent connection to the Internet. (Virtual hosting service and connect-on-demand are cost-effective solutions.) The server must be able to export the shared RCS repository tree to the Internet (password protected, of course). This feature is build-in on NT servers and may be easily installed on UNIX servers using a public-domain product named Samba (NetBIOS over TCP/IP).

## **Workgroup Environment Tutorial**

To setup ComponentSoftware RCS in a workgroup environment, follow these ten simple steps:

1. Activate the setup program on your workstation and click *Master Setup*. Specify a server drive or Windows shared folder for the repository root.
2. Follow the quick-start tutorial above to get familiar with ComponentSoftware RCS and to verify full access to the repository.
3. Create several additional test files; Add these files to the RCS repository as well. (*Note:* The working copy of your test files should be located on your local drive and **not** on a shared drive.)
4. Select one or two files and apply *Check-out* with the *Lock* option cleared. That will release the lock from these files. (*Note:* These files are now read-only and should not be modified.)
5. Activate the *Document Explorer*, choose *Project* and then *Scan New Files*. Files' status should be either *Archived* or *Locked*.
6. On another workstation, activate the setup program and click *Workstation Setup*. (*Note:* You must use another user name for this workstation.)
7. Activate the *Document Explorer* and choose *Project Scan New Files*. You should see all the archived files marked as *New*. (Means there is no local working copy of these file on this workstation.)
8. Choose *Project* and then *Update*. ComponentSoftware RCS will create a local working copy on the workstation PC. File status is *Blocked by...* for files locked by the administrator, *Locked* for files locked by the workstation user and *Archived* for files without any lock.
9. On the workstation, check-out (locked), modify and check-in one file.
10. Click *F5* on the administrator's *Document Explorer* window. The file that has been modified on the workstation have an *Updated* status. That indicates that your local copy is outdated and you should check-out the updated revision whenever convenient.

***Notes:***

- Follow steps 6-8 to install ComponentSoftware RCS on as many workstations as desired.
- It is **very** important to keep the workstations' time synchronized with your file-server time. (You can use the Windows *net time \\YourServer /set* command for that purpose.)
- By default, the working folders have the same structure on all workstations. You may customize this using ComponentSoftware RCS projects.
- To avoid mutual interference problems in software projects, it is recommended to check-out all *Updated* files and to re-test your modifications. Then you can safely commit a new revision to the RCS repository,

**Thank You**

Now you have the skills necessary to take advantage of the power of ComponentSoftware RCS in a workgroup environment. Thank you for participating in this tutorial.



# Command-line Interface

Sometimes, it is convenient to perform an operation using command-line interface. For instance, you might wish to invoke ComponentSoftware RCS from development tool such as Visual Basic or Delphi or from batch file. ComponentSoftware RCS supports basic command-line interface and an advanced command-line interface.

## ***Basic Command-line Interface***

The basic command-line interface has been designed to enable the integration of ComponentSoftware RCS with development tools such as Visual Basic, Delphi, Visual C++ and programmer's editors.

The basic command-line interface syntax is:

`CSRCS command "file-path"`

### ***Basic command-line interface commands:***

*Create* – Add a new file to the RCS repository

*CheckOut* – Check-out a working copy.

*CheckIn* – Check-in a modified file.

*Status* – Inspect the RCS status of a file

*History* – Trace the revisions of a file.

*Help* – Get on-line help.

### ***Example:***

To check-in a modified file to the RCS repository, type in the Explorer **Run...** window:

`CSRCS CheckIn "C:\Project1\Src\test.c"`

## Advanced Command-line Interface

The advanced command-line interface has been designed to deliver the full power of ComponentSoftware RCS for the Explorer *Run* window, DOS command windows and batch files. (*Note:* The advanced command-line interface is a superset of the basic command-line interface.)

The command-line interface syntax is:

`CSRCS command [command-switches] "file1" ["file2" ...]`

The *file* argument(s) can be either:

- Full path
- Relative path to the current working directory.
- Wildcard argument

### ***Examples:***

To get the RCS status of all files in the current directory tree type:

`CSRCS Status /s *.*`

To check-in all .c and .h files in the current directory type:

`CSRCS ci *.c *.h`

To retrieve all files of project *Test* as they were on July 15, type:

`CSRCS retrieve /d"15-Jul" /p"Test"`

The next sections elaborate the various commands and command options.

### *Create command*

Purpose: Add new files to the RCS repository

Syntax: CSRCS Create [*command-switches*] "*file1*" [*file2*" ...]

Abbreviation: *Add*

Switches:

/s – Handle files in specified directory and all sub-directories.

Exit code: 0 – success; 1 – otherwise.

## *CheckOut command*

Purpose: Check-out working copy of files.

Syntax: CSRCS CheckOut [*command-switches*] "*file1*" ["*file2*" ...]

Abbreviation: *co*

Switches:

*/s* – Handle files in specified directory and all sub-directories.

*/p* "*ProjectName*" – Perform operation on all files in project *ProjectName*.

*/l* – Lock the files.

*/u* – Unlock the files.

*/q* – quiet mode. (Do not display dialog box and status messages)

Exit code: 0 – success; 1 – otherwise.

## *Update command*

Purpose: Update the local working copy of files with a new archive revision.

Syntax: CSRCS Update [*command-switches*] "*file1*" [*file2*" ...]

Abbreviation: *upd*

Switches:

*/s* – Handle files in specified directory and all sub-directories.

*/p* "*ProjectName*" – Perform operation on all files in project *ProjectName*.

*/q* – quiet mode. (Do not display dialog box and status messages)

Exit code: 0 – success; 1 – otherwise.

*Notes:*

Unlike the *Check-out* command, the *Update* command does not change the *lock* state of files and does not suggest to revert modified files.

## *CheckIn command*

Purpose: Check-in modified files. Assign symbolic name to revision.

Syntax: CSRCS CheckIn [*command-switches*] "*file1*" [*file2*" ...]

Abbreviation: *ci*

Switches:

*/s* – Handle files in specified directory and all sub-directories.

*/p* "*ProjectName*" – Perform operation on all files in project *ProjectName*.

*/l* – Lock the files.

*/u* – Unlock the files.

*/m* "*message*" – Use *message* to describe the checked-in revision.

*/m@filename* – Read message from *filename*, use to describe the checked-in revision.

*/nname* – Use *name* as the checked-in revision symbolic name.

*/q* – quiet mode. (Do not display dialog box and status messages)

Exit code: 0 – success; 1 – otherwise.

## Mark command

Purpose: Mark current revision of files. Use this command to “freeze” the current status whenever your project reaches a milestone.

Syntax: CS RCS Mark [*command-switches*] “*file1*” [“*file2*” ...]

Abbreviation: *mr*

Switches:

*/s* – Handle files in specified directory and all sub-directories.

*/p* “*ProjectName*” – Perform operation on all files in project *ProjectName*.

*/nname* – Mark current revision with a symbolic name.

*/m* “*message*” – Use *message* to describe checked-in modified files.

*/tstate* – Assign *state* to the current revision. (See note below)

*/rrev* – Change the current revision number to *rev*. (See note below)

*/q* – quiet mode. (Do not display dialog box and status messages)

Exit code: 0 – success; 1 – otherwise.

Notes:

1. Before activating this command, verify none of the files is locked by other users.
2. A state is an identifier used to categorize the revision. A useful set of states is Exp (for experimental; this is the default), Stab (for stable), and Rel (for released). The revision state is listed in the RLOG report and expanded by the \$State\$, \$Id\$ and \$Header\$ keywords.
3. You may optionally assign a new revision number (e.g., 2.0) to register the milestone. Since symbolic names provide the same benefits, this feature is redundant and provided for compatibility reasons only. If you use this feature, you should force the new version number for new files added to the archive. See the *Advanced* properties tab for more details.

## *Retrieve command*

Purpose: Retrieve archived revision of files.

Syntax: CSRCS Retrieve [*command-switches*] "*file1*" [*file2*" ...]

Abbreviation: *ret*

Switches:

*/s* – Handle files in specified directory and all sub-directories.

*/p*"*ProjectName*" – Perform operation on all files in project *ProjectName*.

*/d*"*date*" – Retrieve the last revision created before a specific date. (if any)

*/nname* – Retrieve a revision marked with a symbolic name. (if any)

*/a*"*AlternateWorkRoot*" – Retrieve to an alternate working directory.

*/q* – quiet mode. (Do not display dialog box and status messages)

Exit code: 0 – success; 1 – otherwise.

*Notes:*

1. */d* and */n* are mutually exclusive. If non of them is specified, the head revision is retrieved.
2. Valid date formats are: DD-MMM, DD-MMM-YY, DD-MMM-YY HH:MM or HH:MM. For example: 15-Jun-97 10:45 ; 16-Jun ; 20:00 (today)



## *Scan command*

Purpose: Scan new files of project.

Syntax: CSRCS Scan *command-switches*

Abbreviation: *sc*

Switches:

*/p "ProjectName"* – Perform operation on all folders of project *ProjectName*.

*/q* – quiet mode. (Do not display dialog box and status messages; this is the default)

Exit code: 0 – success; 1 – otherwise.

*Notes:*

All project operations are performed on project listed files. It is recommended to use the *Scan* command from time to time to update the project file list.

## *Status command*

Purpose: Display the files' RCS status

Syntax: CSRCS Status [*command-switches*] "*file1*" ["*file2*" ...]

Abbreviation: *st*

Switches:

*/s* – Handle files in specified directory and all sub-directories.

*/p* "*ProjectName*" – Perform operation on all files in project *ProjectName*.

Exit code: 0 – success; 1 – otherwise.

## Report command

Purpose: Generate project reports.

Syntax: CSRCS Report *command-switches*

Abbreviation: *None*

Switches:

*/p "ProjectName"* – Generate report on project *ProjectName*.

*/r "ReportDescription"* – Report definition string. The report definition string consists on the following fields: (*Note*: All fields are optional. Fields are concatenated and may be specified in any order.)

- **O:** *"OutputFileName"*

Output is generated into *OutputFileName*. When this field is not specified, the *Project Report Dialog* is invoked. (*Tip*: Initially, omit this parameter to debug all other parameters' setting.)

- **R:**A|H

Specify the report type. (A - Text ; H - HTML)

- **C:***ColNum*[...]

Specify one or more columns to report. (0 - Document name ; 1 - Document type ; 2 - Folder name ; 3 - RCS Status ; 4 - Time modified ; 5 - Head revision number ; 6 - Last author/locker)

- **S:***ColNum*[A|D]

Column number used as the sort key (column numbers as above). Optionally, Specify the sort order (A - Ascending ; D - Descending)

- **F:***From*[,To]

Specify alphanumeric range to report. (The filter is applied on the selected sort field.)

Exit code: 0 – success; 1 – otherwise.

*Example:* CSRCS report */pTest /rR:HO:"C:\Temp\rcsreport.htm"C:3201S:0A*

This command generates an HTML report of project *Test* into file C:\Temp\rcsreport.htm. The report consists on the first 4 columns and is sorted by file name in ascending order.

### *History command*

Purpose: Trace the revisions of a file.

Syntax: CSRCS History "*file*"

Abbreviation: *his*

Switches:

None

Exit code: 0 – success; 1 – otherwise.

*Note:* The history command is valid for single file only.

*Help command*

Purpose: Get on-line help.

Syntax: CSRCS Help